Evasion Attacks

A. M. Sadeghzadeh, Ph.D.

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

October 24, 2025

Today's Agenda

Intriguing Properties of neural networks

Intriguing properties of neural networks

Intriguing properties of neural networks

Christian Szegedy
Google Inc.

Wojciech Zaremba
New York University

Ilya Sutskever
Google Inc.

Joan Bruna
New York University

Dumitru Erhan
Google Inc.

Ian Goodfellow
University of Montreal

Rob Fergus
New York University
Facebook Inc.

Abstract

- Deep neural networks are powerful learning models that achieve excellent performance on visual and speech recognition problems.
- They can be difficult to interpret and can have **counter-intuitive properties**.

- Two counter-intuitive properties of deep neural networks:

1. There is no distinction between individual high level units and random linear combinations of high level units.
   - This suggests that **it is the space**, rather than the individual units, that **contains the semantic information** in the high layers of neural networks.
2. By applying an **imperceptible non-random perturbation** to a test image, it is possible to arbitrarily **change the network's prediction**.
   - Such perturbed examples are called **Adversarial Examples**.
   - Adversarial examples are relatively robust, and are **shared by neural networks** with varied number of layers, activations, or trained on different subsets of the training data (**Transferability**).

Notation

- Denote by $x \in R^m$ an input image, and $\phi(x)$ activation values of some layer, where m is the input dimension.
- We denote by $f : \mathbb{R}^m \rightarrow \{1...k\}$ a classifier mapping image pixel value vectors to a discrete label set.
- We assume that f has an associated continuous loss function denoted by $loss_f : \mathbb{R}^m \times \{1...k\} \rightarrow \mathbb{R}^+$.

Units of $\phi(x)$

- Traditional computer vision systems rely on feature extraction: often a single feature is easily interpretable, e.g. a histogram of colors.
- Some works interpret an activation of a hidden unit as a meaningful feature. They look for input images which maximize the activation value of this single feature.
- The aforementioned technique can be formally stated as visual inspection of images x', which satisfy (or are close to maximum attainable value):

$$x' = \underset{x \in \mathcal{I}}{argmax} \ \langle \phi(x), e_i \rangle \tag{1}$$

where $\mathcal{I}$ is a hold-out set of images from the data distribution that the network was not trained on and e_i is the natural basis vector assocated with the i-th hidden unit.

Units of $\phi(x)$

(a) Unit sensitive to lower round stroke.

(b) Unit sensitive to upper round stroke, or lower straight stroke.

(c) Unit senstive to left, upper round stroke.

(d) Unit senstive to diagonal straight stroke.

Units of $\phi(x)$

- The experiments show that any random direction $v \in R^n$ gives rise to similarly interpretable semantic properties.
- More formally, They find that images x' are semantically related to each other, for many x' such that

$$x' = \underset{x \in \mathcal{I}}{argmax} \ \langle \phi(x), v \rangle \tag{2}$$

- This suggests that the natural basis is not better than a random basis for inspecting the properties of $\phi(x)$.
- This puts into question the notion that neural networks disentangle variation factors across coordinates.

Units of $\phi(x)$

(a) Direction sensitive to upper straight stroke, or lower round stroke.

(b) Direction sensitive to lower left loop.

(c) Direction senstive to round top stroke.

(d) Direction sensitive to right, upper round stroke.

Network Level Inspection

- So far, unit-level inspection methods had relatively little utility beyond confirming certain intuitions regarding the complexity of the representations learned by a deep neural network
- **Network level inspection** methods can be useful in the context of explaining classification decisions made by a model
  - For instance, identify the parts of the input which led to a correct classification of a given visual input instance
- Such global analyses are useful in that they can make us understand better the input-to-output mapping represented by the trained network.

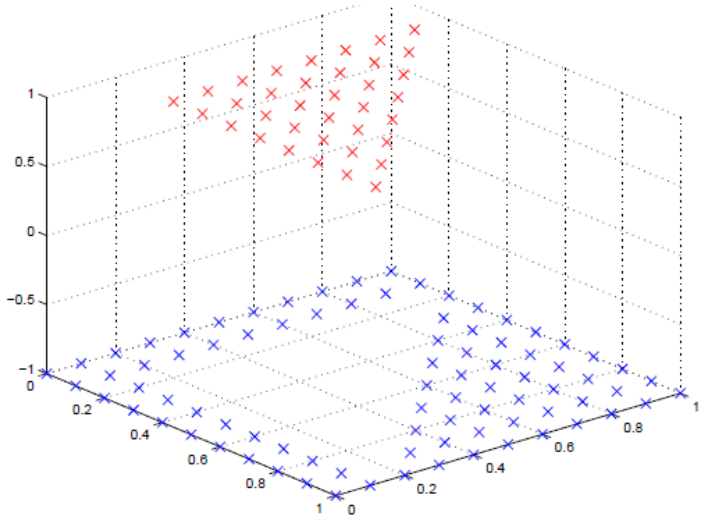
How to Explain Individual Classification Decisions

A probability function $P : \mathbb{R}^d \rightarrow [0, 1]$ of a classification model learned from examples $\{(x_1, y_1), ..., (x_n, y_n)\} \in \mathbb{R}^d \times \{-1, +1\}$ (binary classification) the explanation vector for a classified test point x_0 is the local gradient of p at x_0:

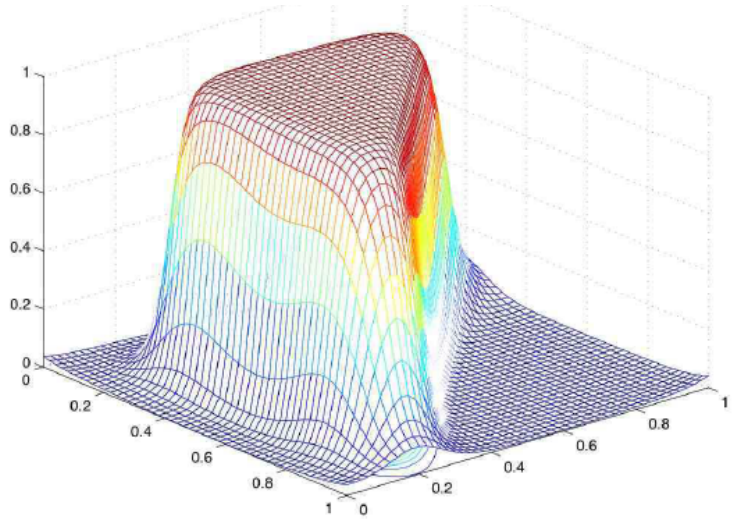
$$\eta_p(x_0) = \nabla_{x_0} P(x_0)$$

- By this definition the explanation η is again a d-dimensional vector just like the test point x_0 is.
- The sign of each of its individual entries indicates whether the prediction would increase or decrease when the corresponding feature of x_0 is increased locally and each entry's absolute value give the amount of influence in the change in prediction.
- As a vector η gives the direction of the steepest ascent from the test point to higher probabilities for the positive class.

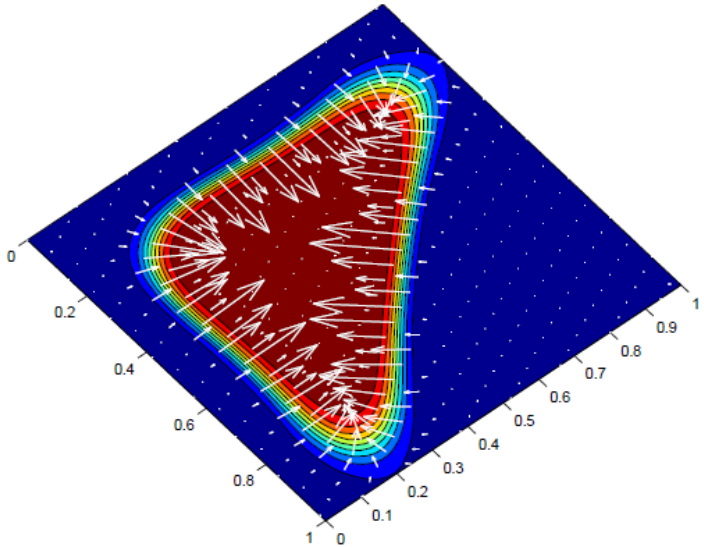
How to Explain Individual Classification Decisions



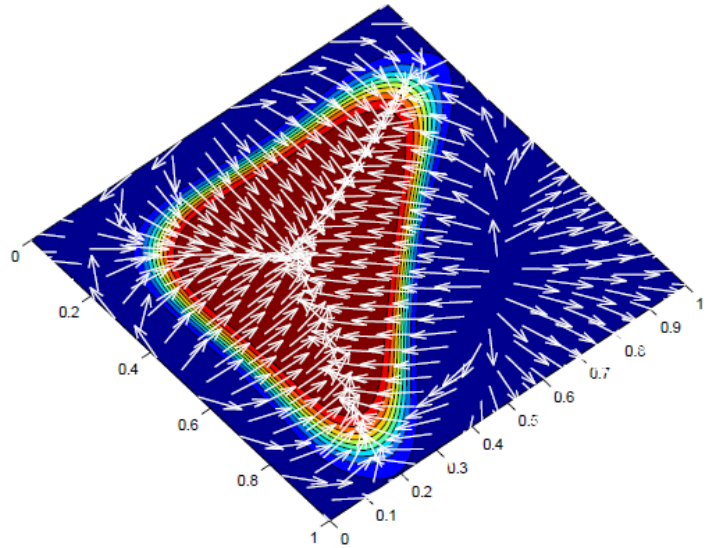
(a) Object



(b) Model



(c) Local explanation vectors



(d) Direction of explanation vectors

How to Explain Individual Classification Decisions

Figure 5: USPS digits (test set bottom part): 'twos' (left) and 'eights' (right) with correct classification. For each digit from left to right: (i) explanation vector (with black being negative, white being positive), (ii) the original digit, (iii-end) artificial digits along the explanation vector towards the other class.

Blind Spots in Neural Networks

- Generally speaking, the output layer unit of a neural network is a **highly nonlinear function** of its input.
- It has been argued that the deep stack of non-linear layers in between the input and the output unit of a neural network are a way for the model to encode a **non-local generalization prior** over the input space.
  - In other words, it is assumed that is possible for the output unit to assign nonsignificant (and, presumably, non-epsilon) probabilities to regions of the input space that contain no training examples in their vicinity.
    - Such regions can represent, for instance, the same objects from different viewpoints, which are relatively far (in pixel space), but which share nonetheless both the label and the statistical structure of the original inputs.

Smoothness Prior

Smoothness Prior

For a small enough radius $\epsilon \geq 0$ in the vicinity of a given training input x, an $x + r$ satisfying $\|r\| \leq \epsilon$ will get assigned a high probability of the correct class by the model.

- This kind of smoothness prior is typically valid for computer vision problems.
- In general, imperceptibly tiny perturbations of a given image do not normally change the underlying class.

However, for deep neural networks this smoothness assumption can be violated: **imperceptibly small perturbations can lead to completely different predictions**. These perturbed samples are known as **adversarial examples**.

Adversarial Pockets in the Manifold

- Adversarial examples can be seen as a way to **traverse the manifold** represented by the network efficiently (through optimization).
- These examples correspond to **low-probability, high-dimensional pockets** in the input space, which are difficult to locate by simply sampling randomly around an example.

Adversarial Example (L-BFGS)

Formal description

For a given $x \in \mathbb{R}^m$ image and target label $l \in \{1...k\}$, we aim to solve the following box-constrained optimization problem:

$$\text{Minimize } \|r\|_2 \text{ subject to:}$$
$$f(x+r) = l$$
$$x + r \in [0,1]^m$$

- Informally, $x' = x + r$ is the closest image to x classified as l by f.

Formal description

For a given $x \in \mathbb{R}^m$ image and target label $l \in \{1...k\}$, we aim to solve the following box-constrained optimization problem:

$$\text{Minimize } \|r\|_2 \text{ subject to:}$$
$$f(x+r) = l$$
$$x + r \in [0,1]^m$$

- Informally, $x' = x + r$ is the closest image to x classified as l by f.
- The minimizer r might not be unique.
- This task is non-trivial only if $f(x) \neq l$.
- In general, the exact computation of x' is a hard problem, so we approximate it by using a box-constrained L-BFGS.

Formal description

Recall: Generalized Lagrange Function (Karush–Kuhn–Tucker (KKT))

Suppose we wish to maximize $f(x)$ subject to $g_j(x) = 0$ for $j = 1, \cdots, J$, and $h_k(x) \geq 0$ for $k = 1, \cdots, K$.

$$\begin{aligned} \text{Minimize} \quad & f(x) \\ \text{subject to} \quad & g_j(x) = 0 \quad \text{for} \quad j = 1, \cdots, J \\ & h_k(x) \geq 0 \quad \text{for} \quad k = 1, \cdots, K \end{aligned}$$

We introduce Lagrange multipliers $\{\lambda_j\}$ and $\{\mu_k\}$, and then optimize the Lagrangian function given by

$$L(x, \{\lambda_j\}, \{\mu_k\}) = f(x) + \sum_{j=1}^{J} \lambda_j g_j(x) + \sum_{k=1}^{K} \mu_k h_k(x)$$

subject to $\mu_k \geq 0$ and $\mu_k h_k(x) = 0$ for $k = 1, \cdots, K$.

The optimal point $x^\star$ of the above constrained optimization on $f(x)$ is the same as the optimal point of the unconstrained optimization L.

(See this playlist for more information about Lagrange multipliers)

Formal description

For a given $x \in \mathbb{R}^m$ image and target label $l \in \{1...k\}$, we aim to solve the following box-constrained optimization problem:

$$\text{Minimize } \|r\|_2 \text{ subject to:}$$
$$f(x+r) = l$$
$$x + r \in [0,1]^m$$

- Informally, $x' = x + r$ is the closest image to x classified as l by f.
- The minimizer r might not be unique.
- This task is non-trivial only if $f(x) \neq l$.
- In general, the exact computation of x' is a hard problem, so we approximate it by using a box-constrained L-BFGS.

Concretely, we find an approximation of x' by performing line-search to find the minimum $c > 0$ for which the minimizer r of the following problem satisfies $f(x+r) = l$.

$$\text{Minimize } c\|r\|_2 + loss_f(x+r, l) \text{ subject to } x + r \in [0,1]^m$$

- Since neural networks are non-convex in general, so we end up with an approximation to find solution.

Adversarial Examples

Figure: Adversarial examples generated for AlexNet.(Left) is a correctly predicted sample, (center) difference between correct image, and image predicted incorrectly magnified by 10x (values shifted by 128 and clamped), (right) adversarial example. All images in the right column are predicted to be an *"ostrich, Struthio camelus"*. Average distortion based on 64 examples is 0.006508.

Experimental results

Intriguing properties

- **100%** success rate
  - For all the networks we studied (MNIST, AlexNet (ImageNet)), for each sample, we have always managed to generate very close, visually hard to distinguish, adversarial examples that are misclassified by the original network.
- **Cross model generalization**
  - A relatively large fraction of examples will be misclassified by networks trained from scratch with **different hyper-parameters** (number of layers, regularization or initial weights).
- **Cross training-set generalization**
  - A relatively large fraction of examples will be misclassified by networks trained from scratch on a **disjoint training set**.

The above observations suggest that adversarial examples are somewhat **universal** and not just the results of overfitting to a particular model or to the specific selection of the training set.

Experimental results

Model Name	Description	Training error	Test error	Av. min. distortion
FC10(10^{-4})	Softmax with $\lambda = 10^{-4}$	6.7%	7.4%	0.062
FC10(10^{-2})	Softmax with $\lambda = 10^{-2}$	10%	9.4%	0.1
FC10(1)	Softmax with $\lambda = 1$	21.2%	20%	0.14
FC100-100-10	Sigmoid network $\lambda = 10^{-5}, 10^{-5}, 10^{-6}$	0%	1.64%	0.058
FC200-200-10	Sigmoid network $\lambda = 10^{-5}, 10^{-5}, 10^{-6}$	0%	1.54%	0.065
AE400-10	Autoencoder with Softmax $\lambda = 10^{-6}$	0.57%	1.9%	0.086

Table 1: Tests of the generalization of adversarial instances on MNIST.

- The last column measures the minimum average pixel level distortion necessary to reach 0% accuracy on the training set.
- The distortion is measure by $\sqrt{\frac{\sum_i (x'_i - x_i)^2}{n}}$ between the original x and distorted x' images, where $n = 28 \times 28 = 784$ is the number of image pixels. The pixel intensities are scaled to be in the range $[0, 1]$.

Cross-model Generalization

	FC10(10^{-4})	FC10(10^{-2})	FC10(1)	FC100-100-10	FC200-200-10	AE400-10	Av. distortion
FC10(10^{-4})	100%	11.7%	22.7%	2%	3.9%	2.7%	0.062
FC10(10^{-2})	87.1%	100%	35.2%	35.9%	27.3%	9.8%	0.1
FC10(1)	71.9%	76.2%	100%	48.1%	47%	34.4%	0.14
FC100-100-10	28.9%	13.7%	21.1%	100%	6.6%	2%	0.058
FC200-200-10	38.2%	14%	23.8%	20.3%	100%	2.7%	0.065
AE400-10	23.4%	16%	24.8%	9.4%	6.6%	100%	0.086
Gaussian noise, stddev=0.1	5.0%	10.1%	18.3%	0%	0%	0.8%	0.1
Gaussian noise, stddev=0.3	15.6%	11.3%	22.7%	5%	4.3%	3.1%	0.3

Table 2: Cross-model generalization of adversarial examples. The columns of the Tables show the error induced by distorted examples fed to the given model. The last column shows average distortion wrt. original training set.

Cross-training-set Generalization

- To study cross-training-set generalization, we have partitioned the 60000 MNIST training images into two parts P_1 and P_2 of size 30000 each and trained three non-convolutional networks with sigmoid activations on them.
- Table 3 summarizes the basic facts about these models. After we generate adversarial examples with 100% error rates with minimum distortion for the test set, we feed these examples to the each of the models.

Model	Error on P_1	Error on P_2	Error on Test	Min Av. Distortion
FC100-100-10: 100-100-10 trained on P_1	0%	2.4%	2%	0.062
FC123-456-10: 123-456-10 trained on P_1	0%	2.5%	2.1%	0.059
FC100-100-10' trained on P_2	2.3%	0%	2.1%	0.058

Table 3: Models trained to study cross-training-set generalization of the generated adversarial examples. Errors presented in Table correpond to original not-distorted data, to provide a baseline.

Cross-model-and-training-set Generalization

- In the last experiment, we magnify the effect of our distortion by using the examples $x + 0.1\frac{r}{\|r\|_2}$ rather than x'.
- This magnifies the distortion on average by 40%, from stddev 0.06 to 0.1.
- The intriguing conclusion is that the adversarial examples remain hard for models trained even on a disjoint training set, although their effectiveness decreases considerably.

	FC100-100-10	FC123-456-10	FC100-100-10'
Distorted for FC100-100-10 (av. stddev=0.062)	100%	26.2%	5.9%
Distorted for FC123-456-10 (av. stddev=0.059)	6.25%	100%	5.1%
Distorted for FC100-100-10' (av. stddev=0.058)	8.2%	8.2%	100%
Gaussian noise with stddev=0.06	2.2%	2.6%	2.4%
Distorted for FC100-100-10 amplified to stddev=0.1	100%	98%	43%
Distorted for FC123-456-10 amplified to stddev=0.1	96%	100%	22%
Distorted for FC100-100-10' amplified to stddev=0.1	27%	50%	100%
Gaussian noise with stddev=0.1	2.6%	2.8%	2.7%

Table 4: Cross-training-set generalization error rate for the set of adversarial examples generated for different models. The error induced by a random distortion to the same examples is displayed in the last row.

Spectral Analysis of Unstability

- The adversarial examples show that there exist **small additive perturbations** of the input (in Euclidean sense) that produce **large perturbations at the output** of the last layer.
- Mathematically, if $\phi(x)$ denotes the output of a network of K layers corresponding to input x and trained parameters W, we write

$$\phi(x) = \phi_K(\phi_{K-1}(...\phi_1(x; W_1)...; W_{K-1})W_K)$$

  where ϕ_K denotes the operator mapping layer $k-1$ to layer k.
- The unstability of $\phi(x)$ can be explained by inspecting the upper **Lipschitz constant** of each layer.

Lipschitz continuity

A function $f : I \to R$ over some set $I \subseteq \mathbb{R}^d$ is called Lipschitz continuous if there exists a positive real constant L such that, for all $x, y \in I$,

$$|f(y) - f(x)| \leq L\|y - x\|_2$$

or

$$f(x) - L\|y - x\|_2 \leq f(y) \leq f(x) + L\|y - x\|_2$$

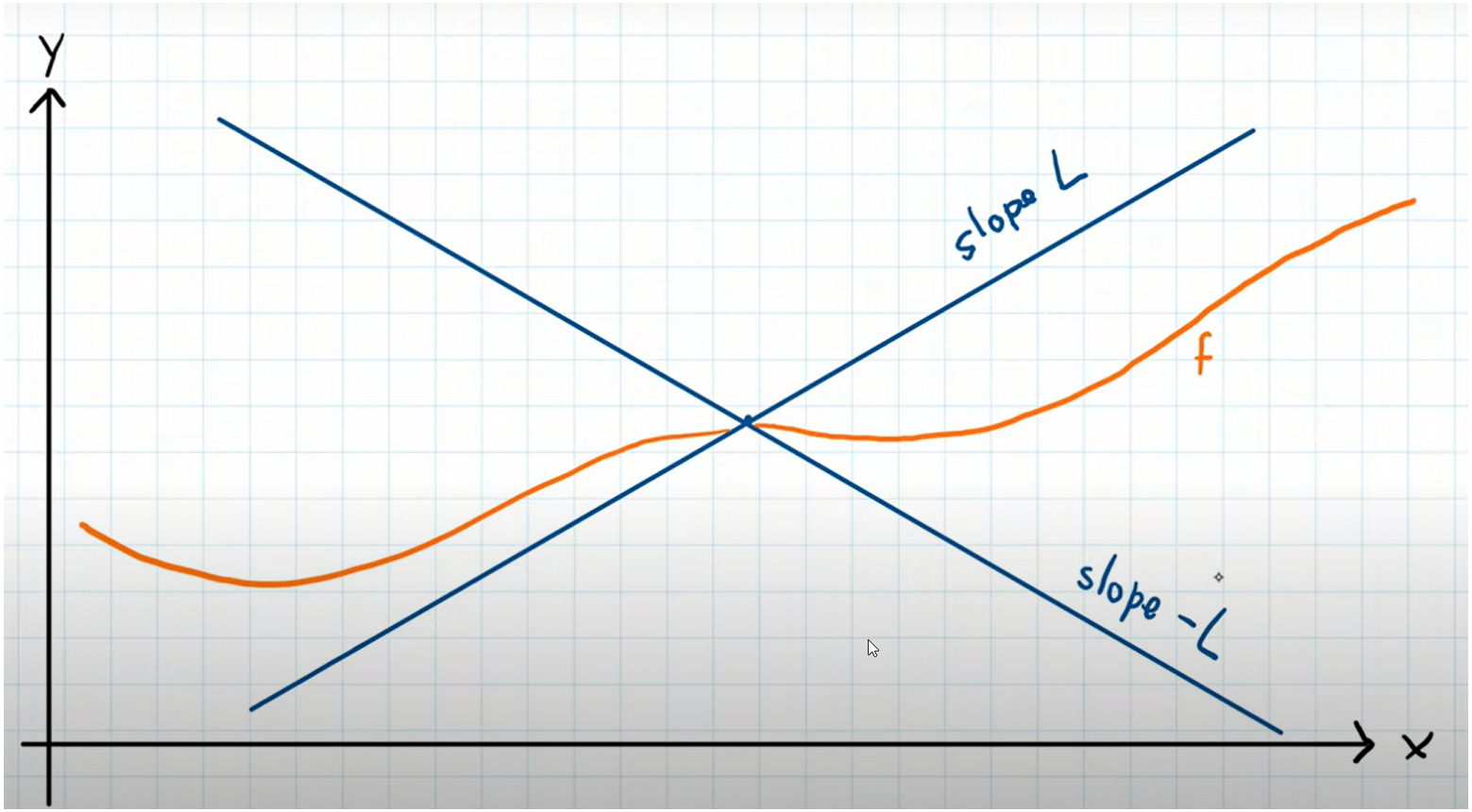
We call L the Lipschitz constant of f over I.

Let functions f_1 and f_2 be both Lipschitz continuous with constants L_1 and L_2, the upper Lipschitz constant of their composition $f_1 \circ f_2$ is $L_1 L_2$.

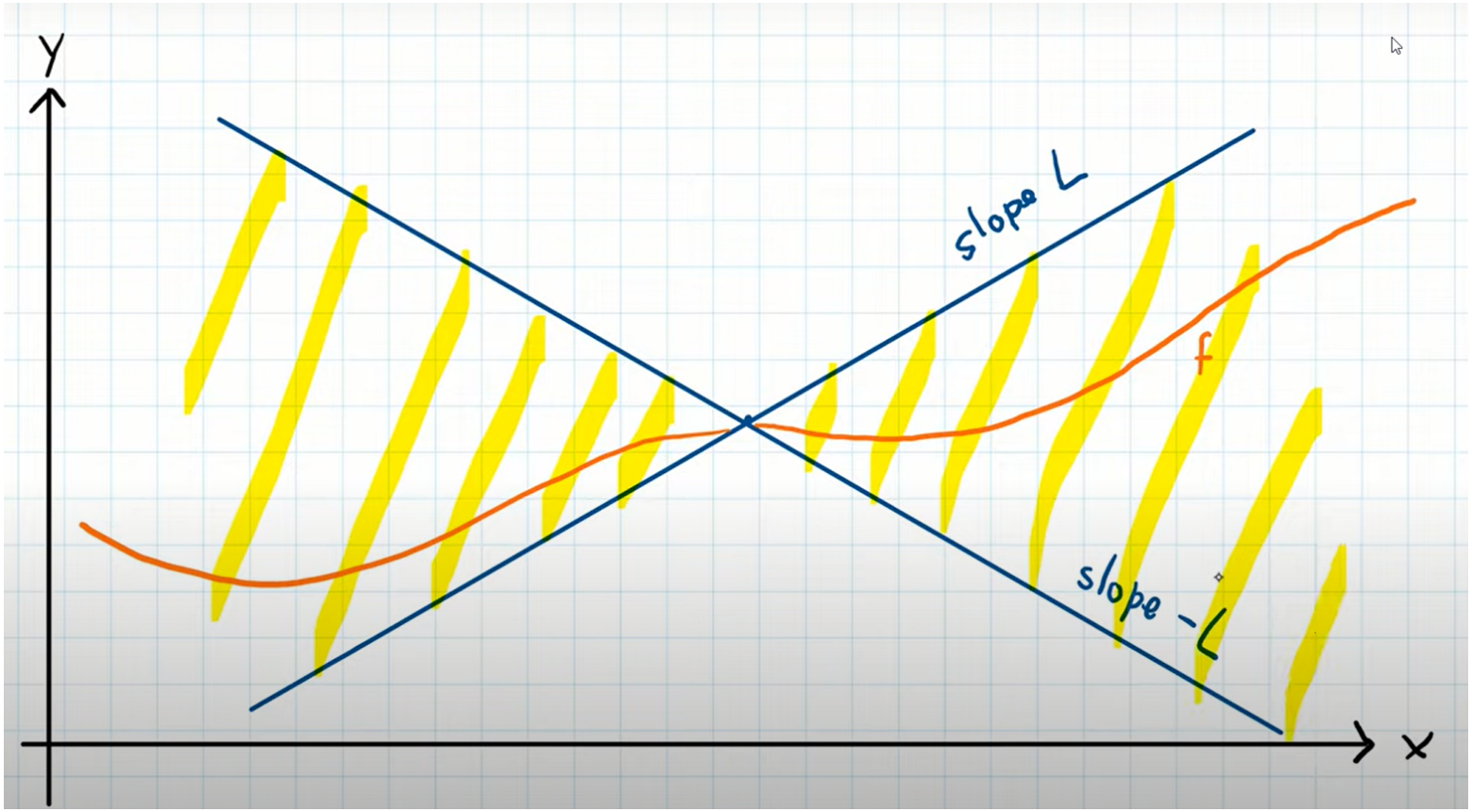
$$|f_1(f_2(y)) - f_1(f_2(x))| \leq L_1|f_2(y) - f_2(x)| \leq L_1 L_2\|y - x\|_2$$

Generally, Let $f = f_1 \circ f_2 \circ .... \circ f_K$ and the Lipschitz constant of f_i be L_i for all $i \in \{1, 2, ..., K\}$, then the Lipschitz constant of f is $L \leq \prod_{k=1}^{K} L_k$.

Lipschitz continuity



Lipschitz continuity



Spectral Analysis of Unstability

- Mathematically, if $\phi(x)$ denotes the output of a network of K layers corresponding to input x and trained parameters W, we write

$$\phi(x) = \phi_K(\phi_{K-1}(...\phi_1(x; W_1)...; W_{K-1})W_K)$$

  where ϕ_K denotes the operator mapping layer $k-1$ to layer k.
- The unstability of $\phi(x)$ can be explained by inspecting the *upper Lipschitz constant* of each layer, defined as the constant $L_k > 0$ such that

$$\forall x, r, \ \|\phi_k(x; W_k) - \phi_k(x + r; W_k)\| \leq L_k \|r\|$$

- The resulting network thus satsifies $\|\phi(x + r) - \phi(x)\| \leq L\|r\|$, with $L \leq \prod_{k=1}^{K} L_k$.

Lipschitz continuity

Let $f : I \to R$ be a continuous and differntiable function over some set $I \subseteq \mathbb{R}^d$, if we have $\|f'(x)\|_2 \leq m$ for all $x \in I$, then m is the upper Lipschitz constant of f ($L \leq m$).

Proof sketch:
Mean value theorem: Let $f : I \to R$ be a continuous and differntiable function over some set $I \subseteq \mathbb{R}^d$, For all $a, b \in I$ $(b > a)$, there exists some $c \in (a, b)$ such that:

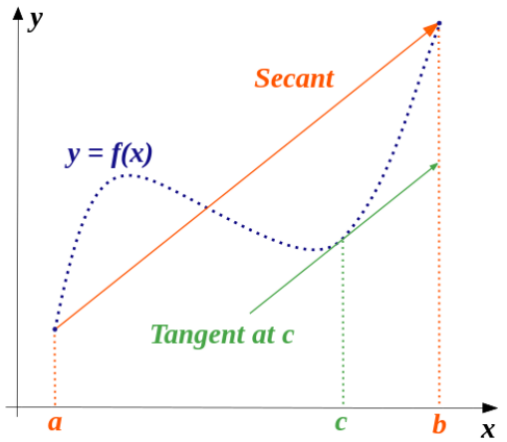
$$f'(c) = \frac{f(b) - f(a)}{b - a}$$

For all $a, b \in I$, there exist $c \in (a, b)$, such that:

$$|f(b) - f(a)| = \|f'(c).b - a\|_2 \leq \|f'(c)\|_2 \|b - a\|_2.$$

Since we know that $\|f'(c)\|_2 \leq m$, we have

$$|f(b) - f(a)| \leq m\|b - a\|_2.$$



Spectral Analysis of Unstability

- ReLU and max pooling layers have a Lipschitz constant of 1.
  - The upper bound of derivative is 1.
- Batch normalization layer has a Lipschitz constant of $\frac{\gamma}{\sqrt{\sigma^2+\epsilon}}$
  - $\nabla_x BN(x) = \nabla_x \gamma \frac{x-\mu}{\sqrt{\sigma^2+\epsilon}} + \beta = \frac{\gamma}{\sqrt{\sigma^2+\epsilon}}$
- Linear layers (Wx) have the Lipschitz constant of $\sigma(W)$, where σ is the spectral norm (largest singular value).
  - Lipschits constant of linear layers

$$\|Wy - Wx\|_2 \leq L\|y-x\|_2 \Rightarrow \|W(y-x)\|_2 \leq L\|y-x\|_2$$

$$\underset{z=y-x}{\Rightarrow} \|Wz\|_2 \leq L\|z\|_2 \Rightarrow L \geq \frac{\|Wz\|_2}{\|z\|_2} \Rightarrow L = \sigma(W)$$

  - The spectral norm of a matrix $A \in \mathbb{R}^{m\times n}$ is defined as

$$\sigma(A) = \max_{x\in\mathbb{R}^n, x\neq 0} \frac{\|Ax\|_2}{\|x\|_2}$$

    which corresponds to the largest singular value of A

Spectral norm definition source.

Spectral Analysis of Unstability

Layer	Size	Stride	Upper bound
Conv. 1	$3 \times 11 \times 11 \times 96$	4	2.75
Conv. 2	$96 \times 5 \times 5 \times 256$	1	10
Conv. 3	$256 \times 3 \times 3 \times 384$	1	7
Conv. 4	$384 \times 3 \times 3 \times 384$	1	7.5
Conv. 5	$384 \times 3 \times 3 \times 256$	1	11
FC. 1	9216×4096	N/A	3.12
FC. 2	4096×4096	N/A	4
FC. 3	4096×1000	N/A	4

$$2.75 \times 10 \times 7 \times 7.5 \times 11 \times 3.12 \times 4 \times 4 \approx 793000$$

- Notice that we compute upper bounds: large bounds do not automatically translate into existence of adversarial examples; however, small bounds guarantee that no such examples can appear.

Type of Adversarial Attacks

Threat Models

White-Box

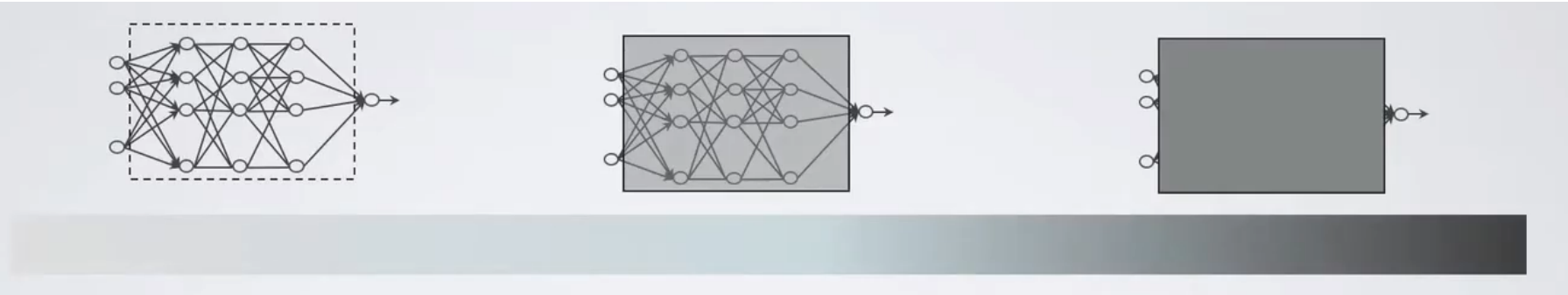
- The adversary has complete access to the algorithm, architecture, parameters, hyper-parameters, and input-output type of the target model.
- Introduced L-BFGS attack has white-box threat model.

Gray-Box

- The adversary has partial access to the algorithm, architecture, parameters, hyper-parameters, and input-output type of the target model.

Black-Box

- The adversary only has API access to the target model.



Targeted and Untargeted Adversarial Examples

Untargetted attack

- The adversary wants to change the predication of the classifier to a wrong class.
  - Example: an image of a “cat” classified as “dog” or “car” (any wrong label).

Targeted attack

- The adversary wants to change the predication of the classifier to a given target class.
  - Example: an image of a “cat” deliberately modified to be classified as “airplane.”

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.
- D. Baehrens, T. Schroeter, S. Harmeling, et al., "How to explain individual classification decisions." The Journal of Machine Learning Research 11 (2010): 1803-1831.