

Adversarial Examples

Authors:
Daniel Donoghue
Arnaldo Pereira
Nicholas Lines

EN.625.638 Neural
Networks
Professor Mark
Fleischer



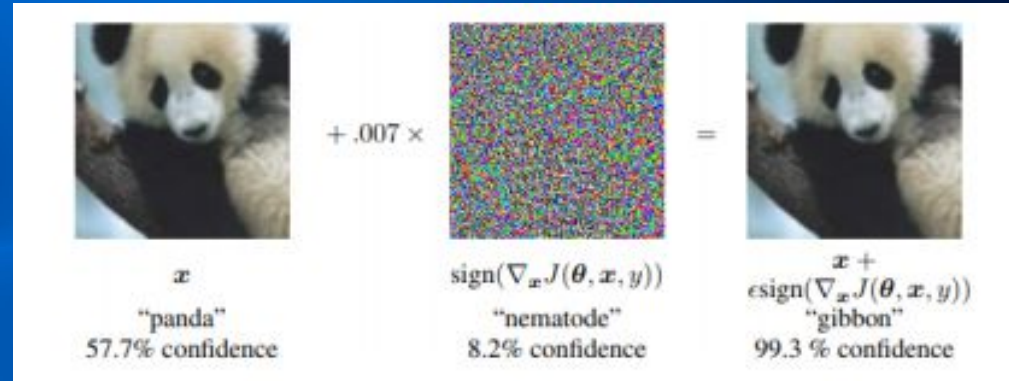
JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Presentation Outline

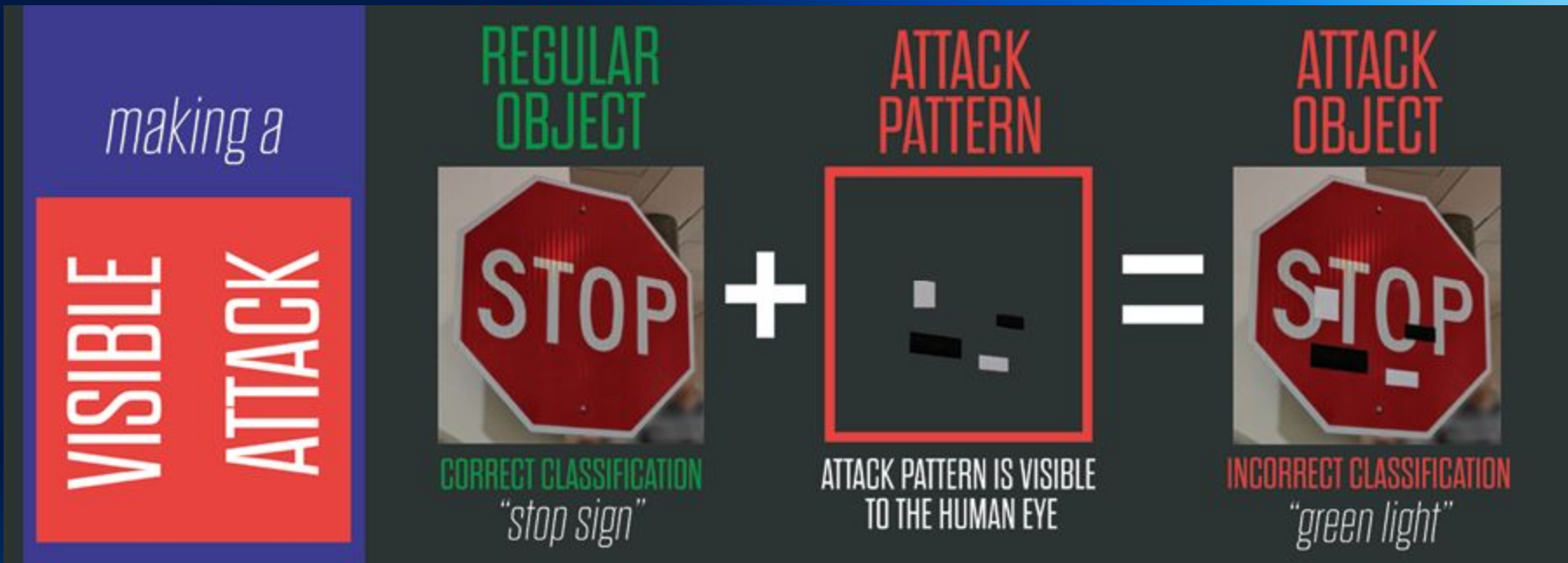
- What are adversarial examples?
- Why are networks vulnerable to adversarial examples?
- Introduction to our project
- Adversarial Attacks
- Adversarial Defenses
- Conclusions
- References

Adversarial Examples

An **Adversarial Example** is: "An input to a machine learning model that is intentionally designed to cause the model to make a mistake in its predictions, despite resembling a valid input to a human." [6]



Threats of Adversarial Attacks



Different Types of Adversarial Attacks



Misuse of data, models, and algorithms with the intent to defeat or exploit these. [2]

Usually it falls into one of these categories:

Extraction methods: (Possibly without training data or algorithm details) use interactions with the model to recover training details or the model itself.

Example: Stealing an image classifier via queries.

Evasion methods: (Possibly without training data or algorithm details) make data examples that are misclassified or misunderstood by the algorithm.

Example: Inverted videos on YouTube.

Data poisoning: (Possibly without algorithm details) Introduce data used to create or refine a model that will produce misclassifications or bad model decisions.

Example: Self-driving car sign misinterpretation.





[1],[3]

Why are networks vulnerable to adversarial examples?

- Each input (feature) has a precision limit, call it ϵ .
- Consider inputs \mathbf{x} . Add $\mathbf{X}+\boldsymbol{\eta}$, where $\boldsymbol{\eta}$ is a vector of constants smaller than ϵ .
- To humans, \mathbf{x} and $\mathbf{X}+\boldsymbol{\eta}$ look the same.
- But the effects of many network layers are approximately linear, so the activation is approximately

$$\mathbf{w}^T (\mathbf{X}+\boldsymbol{\eta}) = \mathbf{w}^T \mathbf{X} + \mathbf{w}^T \boldsymbol{\eta}$$



BIG!

- *I.e.* very small changes to inputs can make BIG output changes if \mathbf{w} is ill-conditioned!

Project Overview

1. Used Imagnette dataset to train Neural Networks to classify images.
2. Implemented Fast Gradient Sign Method and Directed Attacks to Create our own Adversarial Examples.
3. Attempted to defend our network against our attacks.



Imagenette Dataset

[4],[5]

- ImageNet is a standard computer vision dataset. The full version has 14 million images, 20k classes, 320 pixels or smaller.
- Imagenette is a subset of this with only 10 classes that are notoriously easy to predict: {tench (a fish), English springer, cassette player, chain saw, church, French horn, garbage truck, gas pump, golf ball, parachute}
- We (re)trained several neural networks to classify these images, and give a confidence score on those classifications.
- Our tool suite: Github, Python3/Jupyter, Google Colaboratory, PyTorch.



Baseline Performance: 99%+ accuracy



English terrier: 0.995133



gas pump: 0.999048



chainsaw: 0.956269



church: 0.999427



golf ball: 0.999928



church: 0.999813



parachute: 0.996126



French horn: 0.996256



English terrier: 0.995468



chainsaw: 0.998758



gas pump: 0.999271



tench: 0.999652



tench: 0.985282



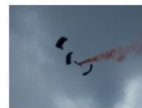
cassette player: 0.900653



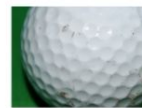
church: 0.999630



gas pump: 0.998723



parachute: 0.999928



golf ball: 1.000000



chainsaw: 0.992427



garbage truck: 0.998693



church: 0.999485



cassette player: 0.991328



gas pump: 0.999990



French horn: 0.998577



parachute: 0.999504



English terrier: 0.999871



chainsaw: 0.458943



golf ball: 0.999540



garbage truck: 0.918213



gas pump: 0.998266



golf ball: 0.999169



gas pump: 0.999765



gas pump: 0.999649



cassette player: 0.996599



English terrier: 0.999295



church: 0.999886



chainsaw: 0.962802



tench: 0.999857



church: 0.999881



French horn: 0.396092



[3]

Generating Adversarial Examples: Fast Gradient Sign (FGS) Method

- Goal: get vector η that matches cost function gradient sign for a given input.

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

Diagram illustrating the components of the Fast Gradient Sign (FGS) method equation:

- η : Perturbation vector
- ϵ : Allowable level of perturbation
- J : Cost function
- $\nabla_{\mathbf{x}}$: Vector of model parameters
- $\boldsymbol{\theta}$: Inputs
- y : True label

- $\mathbf{x}' = \mathbf{x} + \eta$
- Getting the gradient is easy via backpropagation!

FGSM attack samples



[7]



English terrier: 0.342782



church: 0.860821



church: 0.762751



church: 0.723839



golf ball: 0.552850



church: 0.501100



church: 0.978104



cassette player: 0.674960



gas pump: 0.412536



gas pump: 0.455732



church: 0.878713



cassette player: 0.422767



garbage truck: 0.424218

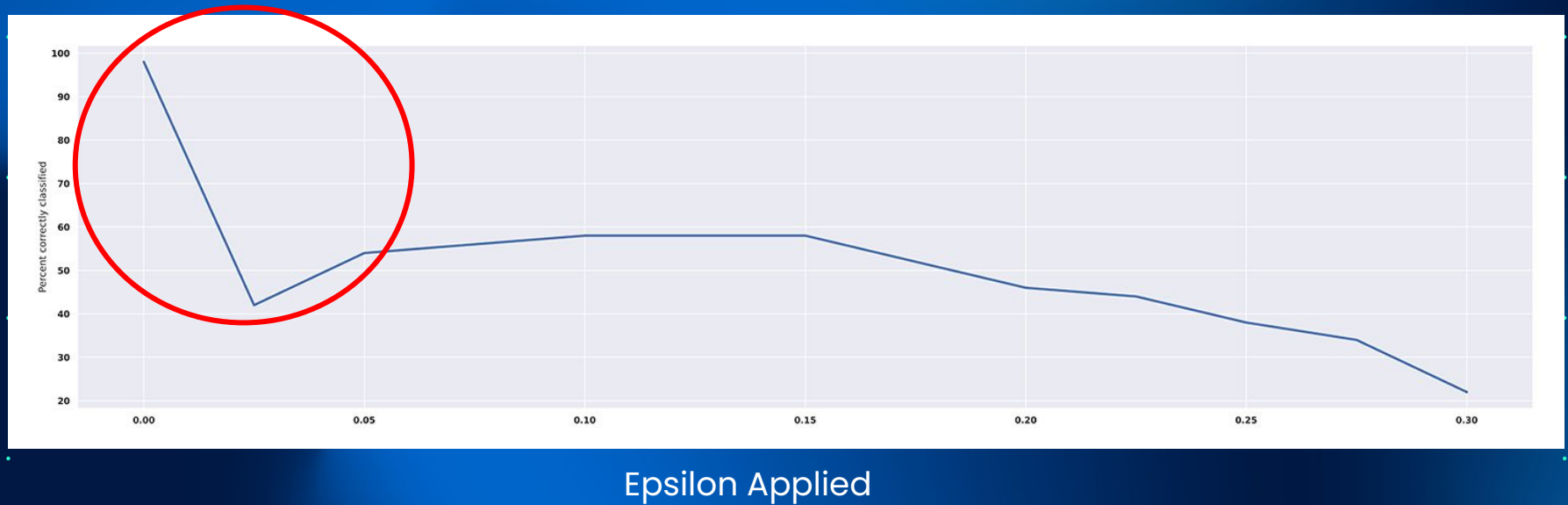


church: 0.852802



parachute: 0.605359

FGSM Attack vs Baseline Network





[7]

Directed Attack

Used to 'direct' the Neural Network to classify all incoming images as a particular class.

For this example, we applied noise to direct every image to be classified as an 'English Terrier'.

In this form of attack, we are looking to not only maximize the loss of the true label, but to also minimize the loss for the target label.



English terrier: 0.995133



English terrier: 1.000000



English terrier: 1.000000



English terrier: 1.000000



English terrier: 1.000000



English terrier: 1.000000

Directed Attack vs Baseline Network

% classified as English terrier



Epsilon Applied



[7]

Directed Attack: Generating Perturbations

Similar to FGSM: Easy to calculate from network gradients!

Differences:

- Use loss function to **direct** toward a specific desired label.
- Iterate using gradient descent for a fixed number of iterations.
- Project gradient onto ℓ_∞ -norm ϵ -sphere (technically a hypercube).
- Fancy way of saying don't change a pixel's color by more than ϵ . We don't want people to notice!

$$\min_{\delta} \{L_{adv}(x + \delta) = L(x + \delta, y_{desired}) - L(x + \delta, y_{true})\} \text{ subject to } \|\delta\|_{\infty} \leq \epsilon.$$

$$\delta_t = \delta_{t-1} - \alpha * \text{sign}(\nabla L_{adv}(x + \delta_{t-1})),$$

$$\delta_t \leftarrow \text{clip}(\delta_t, -\epsilon, \epsilon)$$

$$\text{with } \delta_0 = \mathbf{0}.$$

Visualization of Directed Attack



church: 0.999485



epsilon = 0.02



English terrier: 1.000000

Defense 1: Adversarial Training

The Neural Network is re-trained using a new dataset that includes Adversarial Examples. These Adversarial Examples were generated using directed method, although they were targeting random classes, not just 'English Terrier' as in the previous example.



church: 0.992980

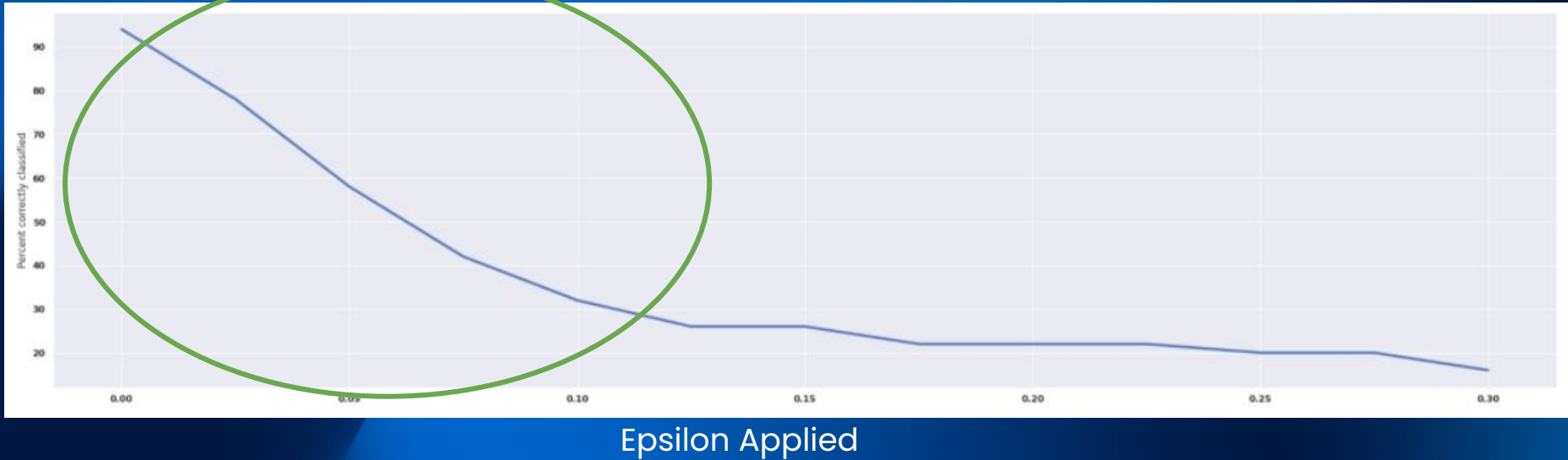


epsilon = 0.02



church: 0.816696

FGSM Attack vs Robust Network



Directed Attack vs Robust Network



English terrier: 0.881161



church: 0.856597



chainsaw: 0.424758



English terrier: 0.919079



golf ball: 0.961225



cassette player: 0.610140



Graph above shows % of images classified as 'English Terrier' vs. Epsilon

Defense 2: Perturbed Prediction Averaging

Use case: We don't have resources or access to the network to retrain it, but need **some** level of defense at inference time against adversarial attacks.

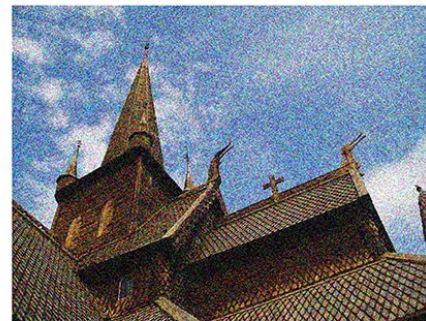
Solution: Add random noise in excess of what we expect from attacks (e.g., enough to be visible). Average over many predictions to “wash out” perturbation of attack.



Original image

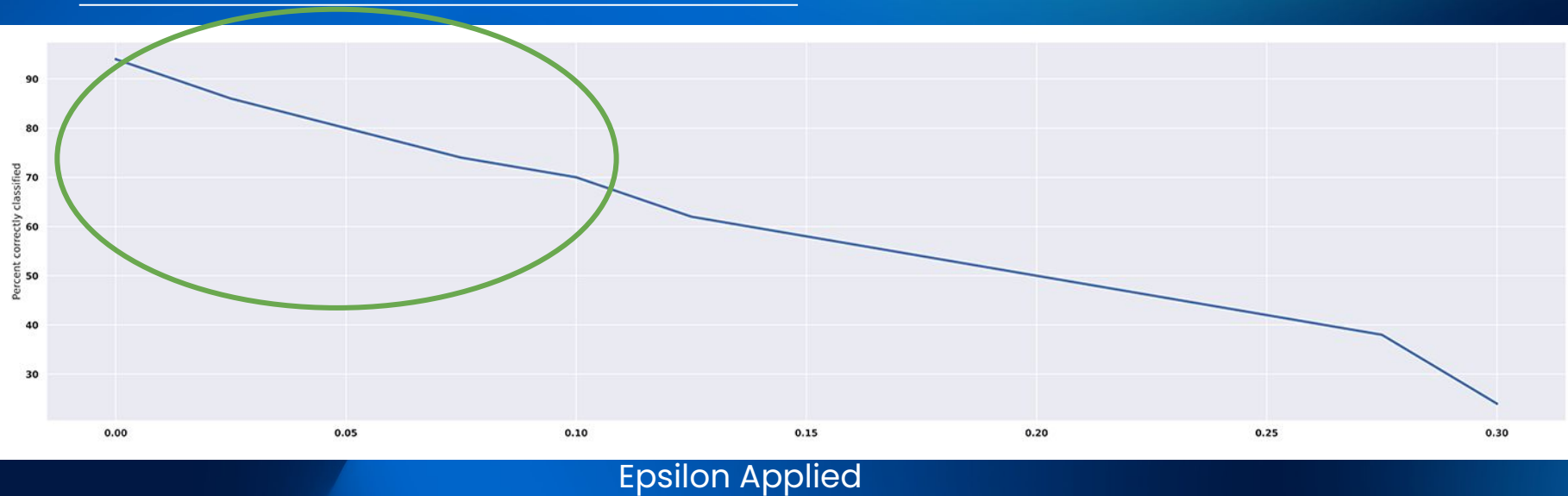


epsilon = 0.3

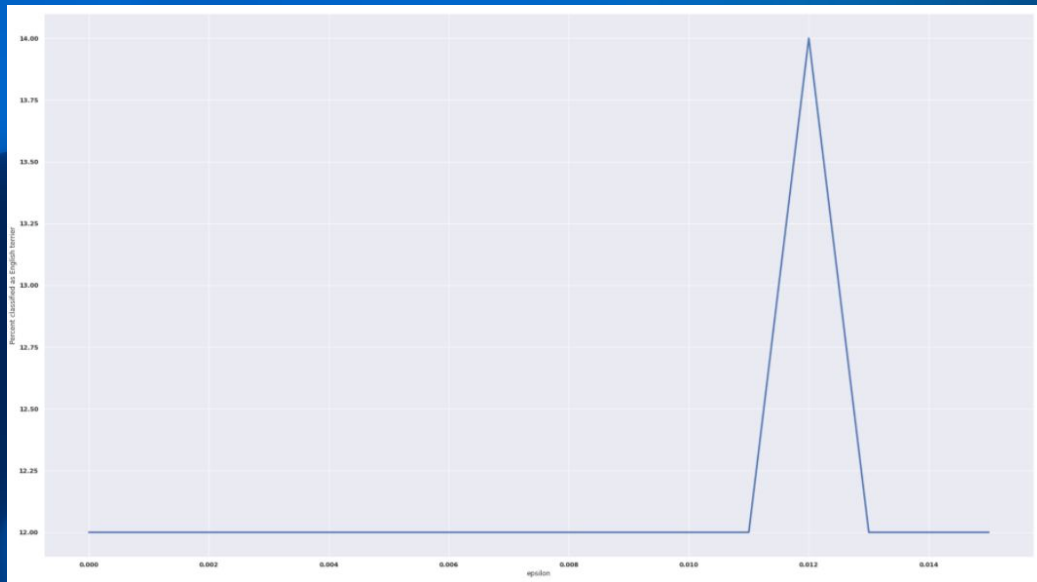


Perturbed image

FGSM Attack Against Prediction Averaging Defense



Directed Attack Against Prediction Averaging Defense



What Are the Defenses Doing?

Recall this from earlier...



church: 0.992980



epsilon = 0.02

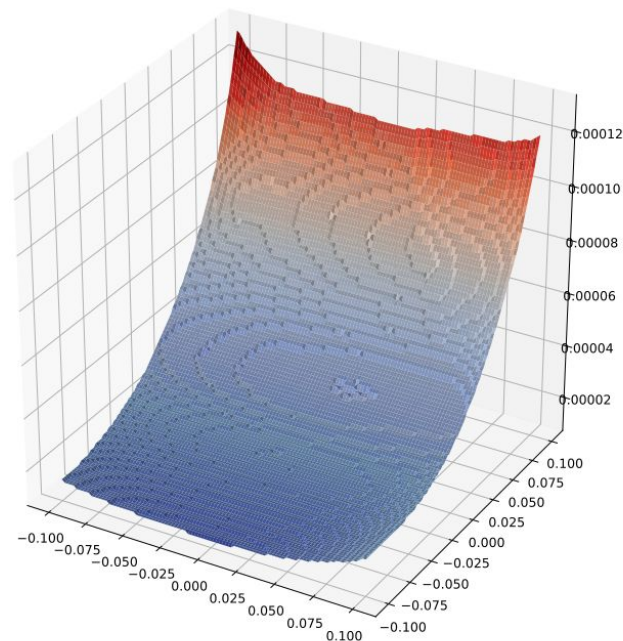
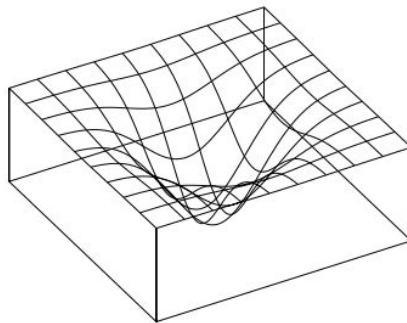
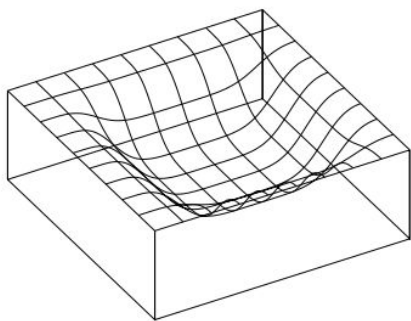


church: 0.816696

What Are the Defenses Doing?

Perturbing an image is the same as predicting on the **original image** with **perturbed weights**.

Flatter parameter = more robustness



Questions?

Thanks for viewing this!

Authors:

Daniel Donoghue: ddonogh1@jhu.edu

Arnaldo Perera: aepereira@gmail.com

Nicholas Lines: nicholasalines@gmail.com

CREDITS: This presentation template was
created by Slidesgo, including icons by Flaticon,
infographics & images by Freepik

References

- [1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2014.
- [2] Wikipedia contributors, "Adversarial machine learning - Wikipedia, the free encyclopedia," https://en.wikipedia.org/w/index.php?title=Adversarial_machine_learning&oldid=987074175, 2020, [Online; accessed 11-November-2020].
- [3] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," arXiv preprint arXiv:1412.6572, 2014.
- [4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in CVPR09, 2009.
- [5] Imagenette. <https://github.com/fastai/imagenette>.
- [6] Wiyatno, R. R., Xu, A., Dia, O., & Berker, A. D. (2019, November 15). Adversarial Examples in Modern Machine Learning: A Review. Retrieved 2020, from <https://arxiv.org/pdf/1911.05268.pdf>.
- [7] Madry, Z. (2020). Chapter 3 - Adversarial examples, solving the inner maximization. Retrieved December, 2020, from https://adversarial-ml-tutorial.org/adversarial_examples/