Adversarial Examples

Authors:
Daniel Donoghue
Arnaldo Pereira
Nicholas Lines

EN.625.638 Neural Networks
Professor Mark Fleischer

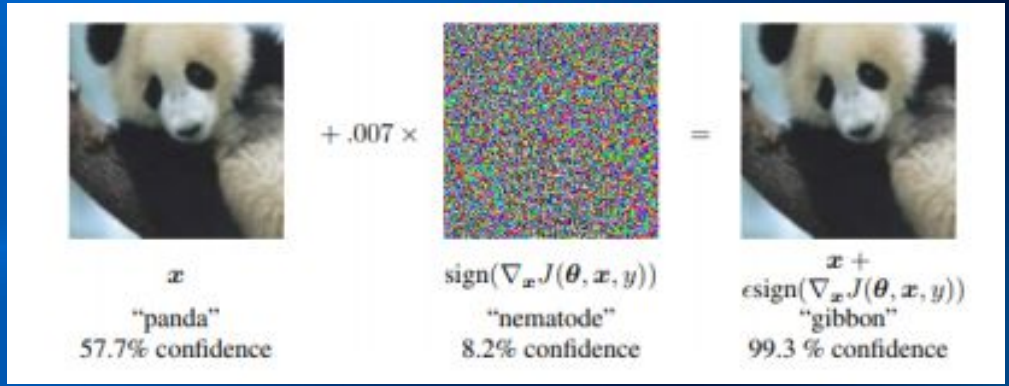
JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

Presentation Outline

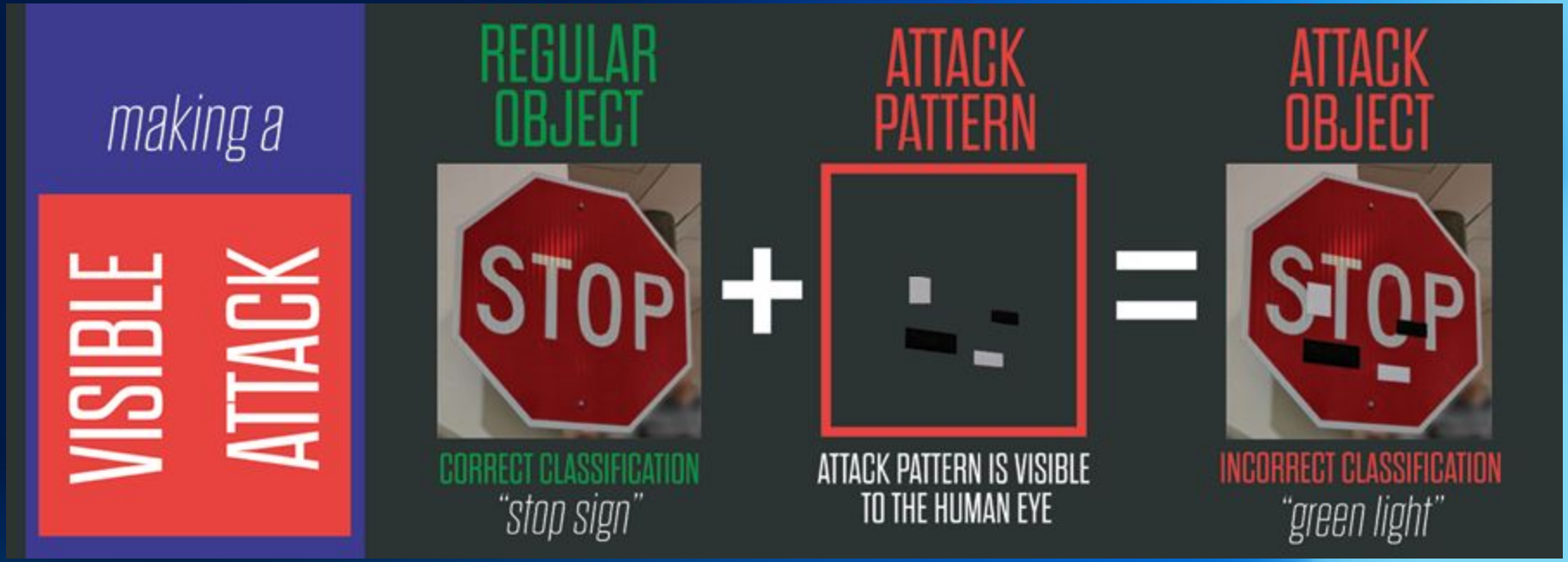
- What are adversarial examples?
- Why are networks vulnerable to adversarial examples?
- Introduction to our project
- Adversarial Attacks
- Adversarial Defenses
- Conclusions
- References

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.

Why are networks vulnerable to adversarial examples?

[1],[3]

- 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

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

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

Project Overview

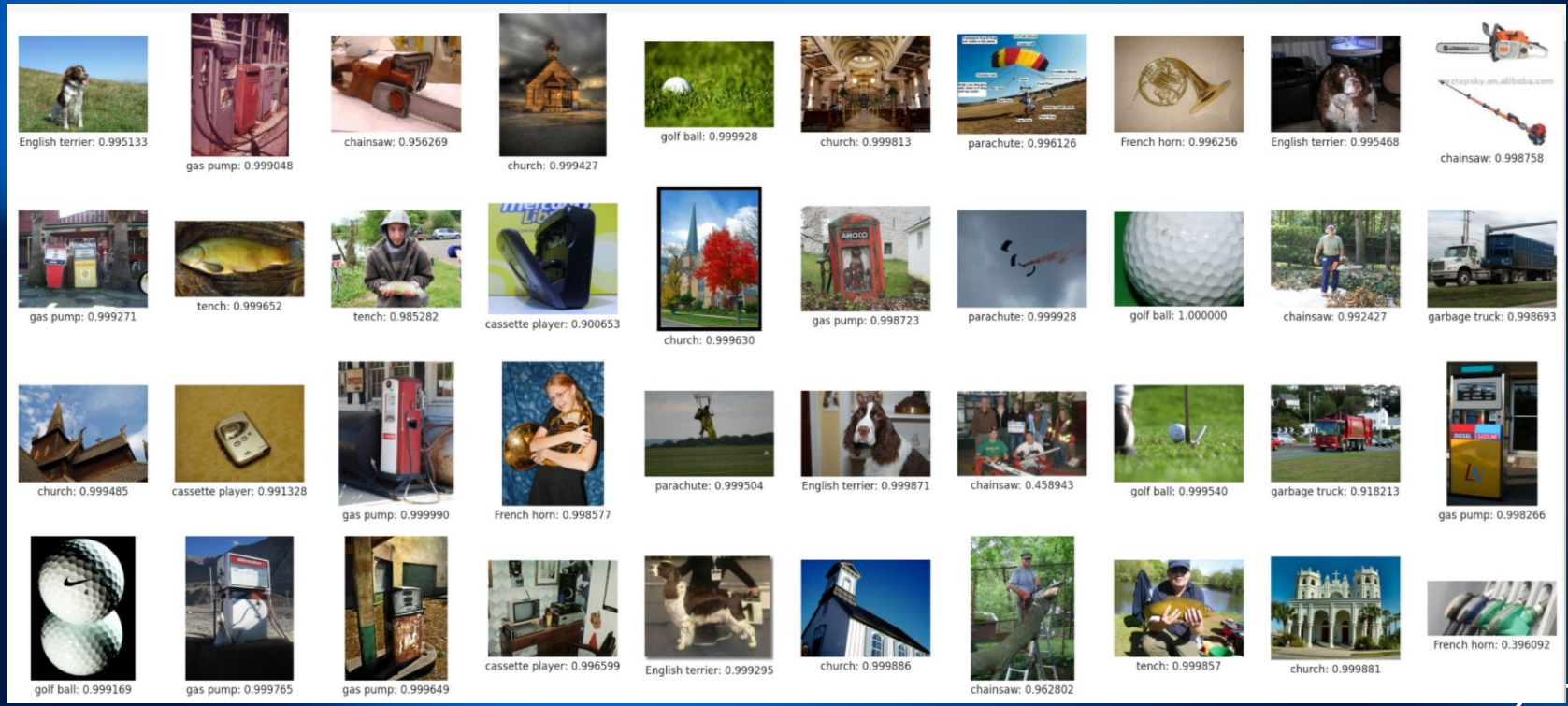
1. Used Imagnette dataset to train Neural Networks to classify images.
2. Implemented Fast Gradient Sign Method and Directed Attacks to Create our own Adversarial Examples.
3. Attempted to defend our network against our attacks.

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



Generating Adversarial Examples: Fast Gradient Sign (FGS) Method

[3]

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

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

Perturbation vector

Allowable level of perturbation

Cost function

Vector of model parameters

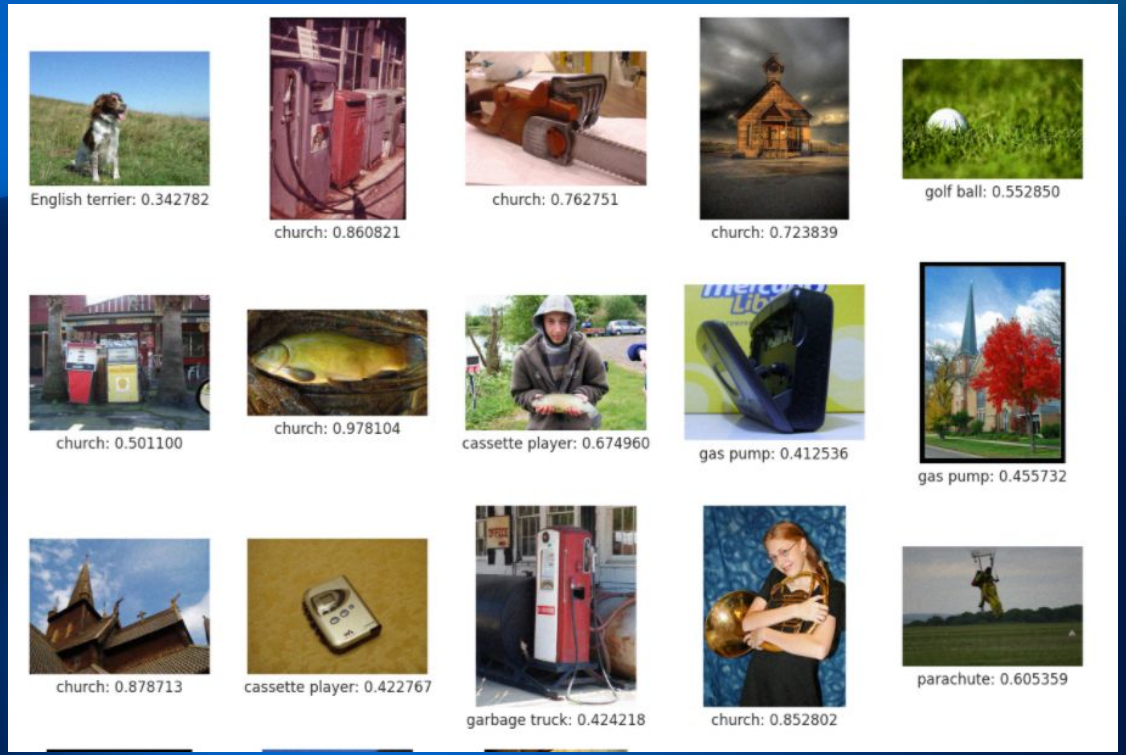
Inputs

True label

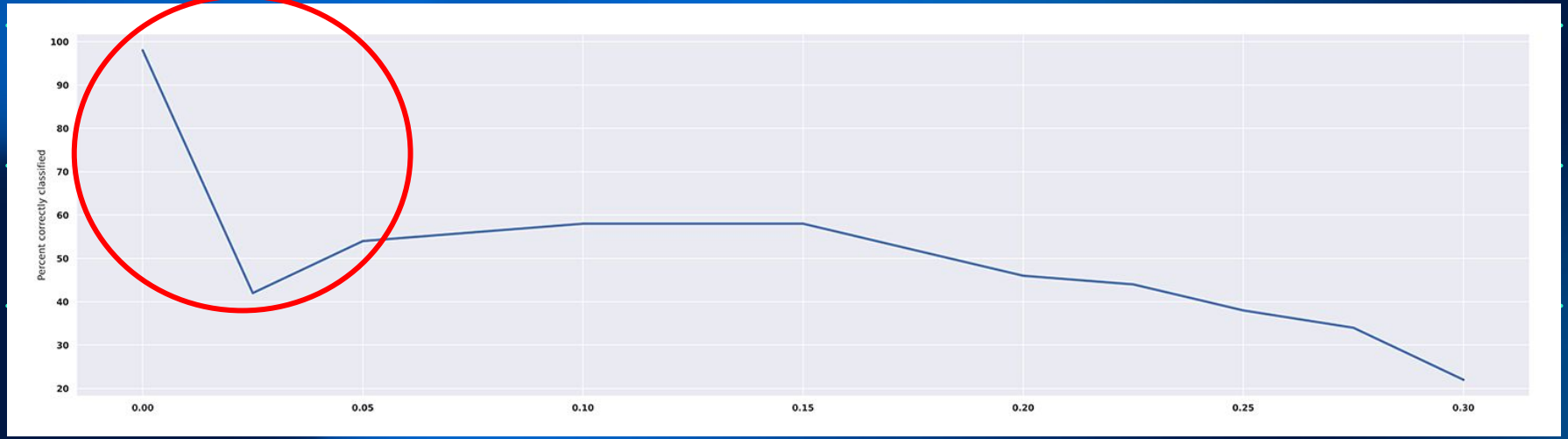
- $x' = x + \eta$
- Getting the gradient is easy via backpropagation!

FGSM attack samples

[7]



FGSM Attack vs Baseline Network



Epsilon Applied

Directed Attack

[7]

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.

Directed Attack vs Baseline Network

% classified as English terrier

Epsilon Applied

Directed Attack: Generating Perturbations

Similar to FGSM: Easy to calculate from network gradients!

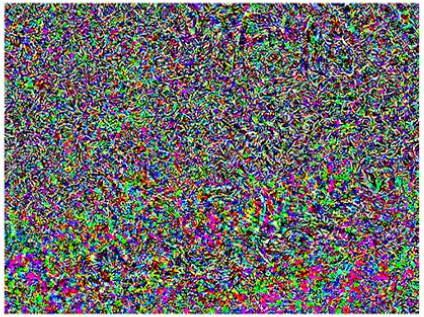
Differences:

- 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!

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

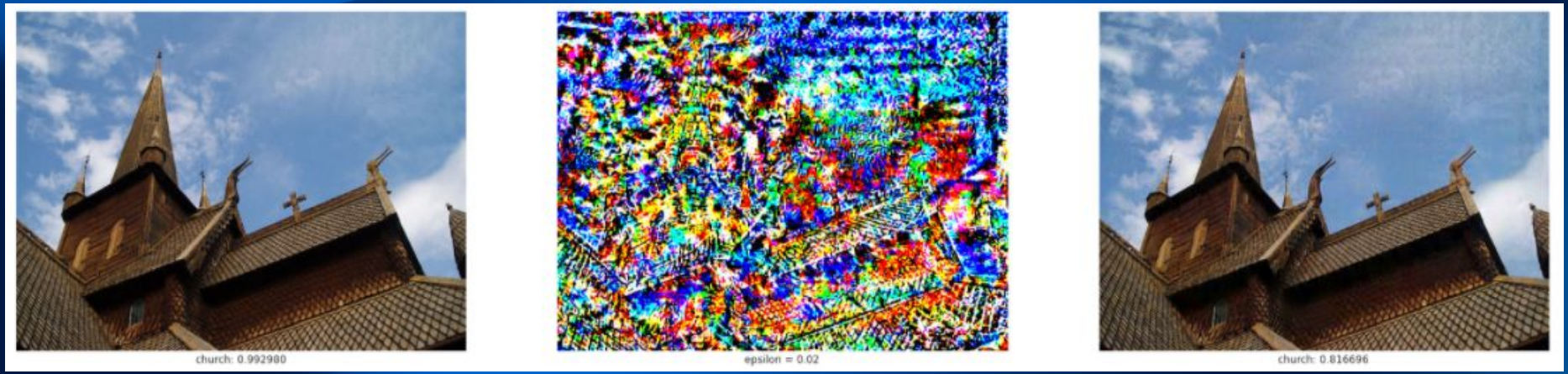
$$\delta_t = \delta_{t-1} - \alpha * sign(\nabla L_{adv}(x+\delta_{t-1})),$$
$$\delta_t \leftarrow clip(\delta_t, -\epsilon, \epsilon)$$
$$\text{with } \delta_0 = \mathbf{0}.$$

Visualization of Directed Attack

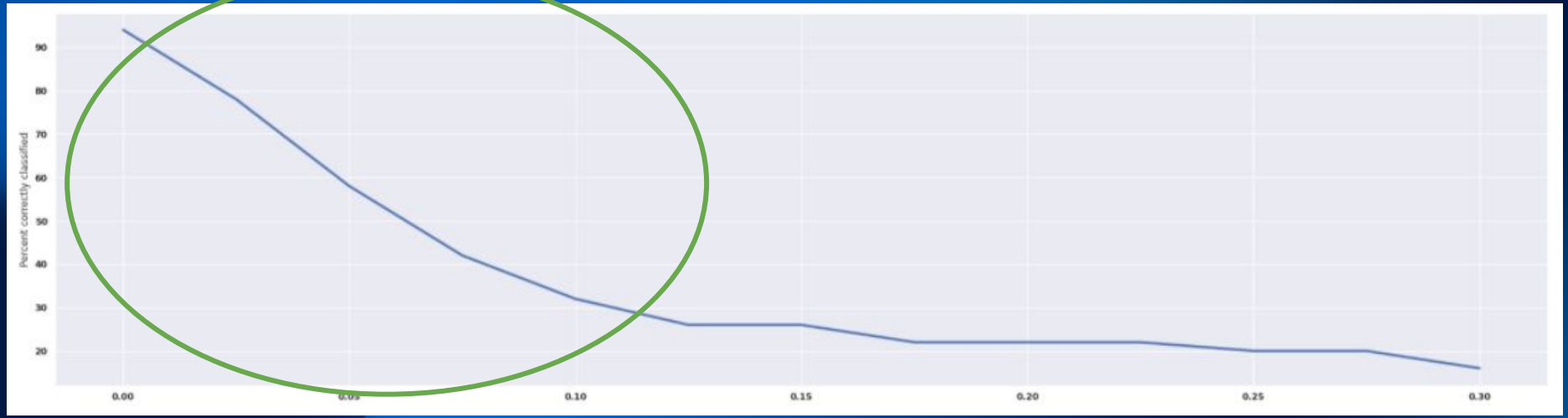


Defense 1: Adversarial Training

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.

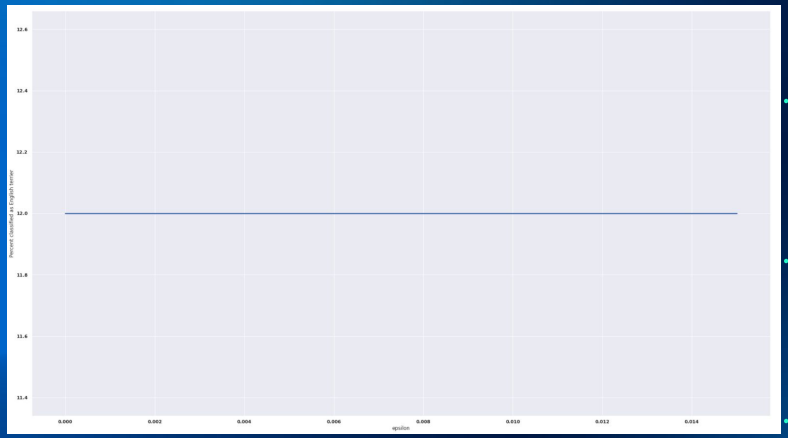


FGSM Attack vs Robust Network



Epsilon Applied

Directed Attack vs Robust Network

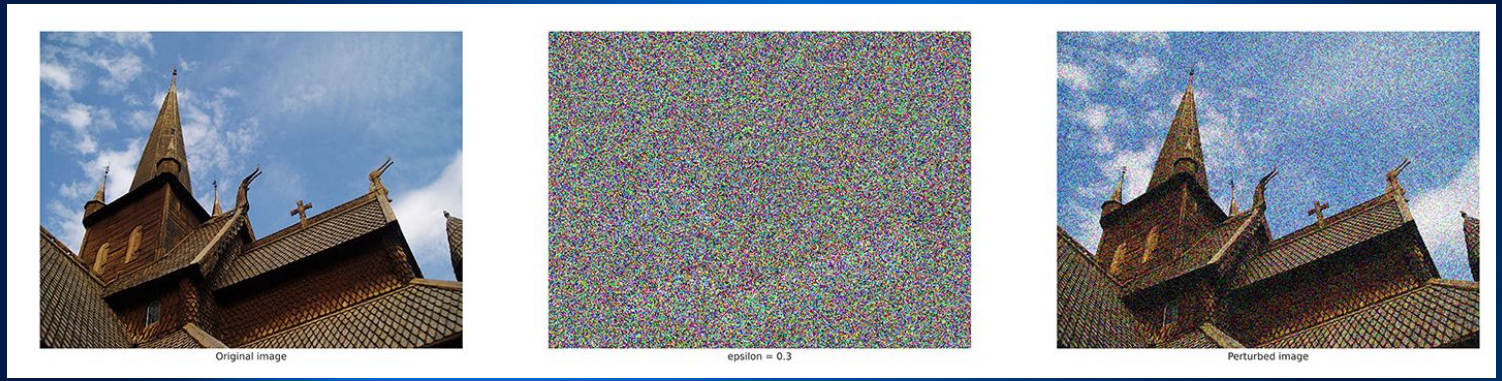


Graph above shows % of images classified as ‘English Terrier’ vs. Epsilon

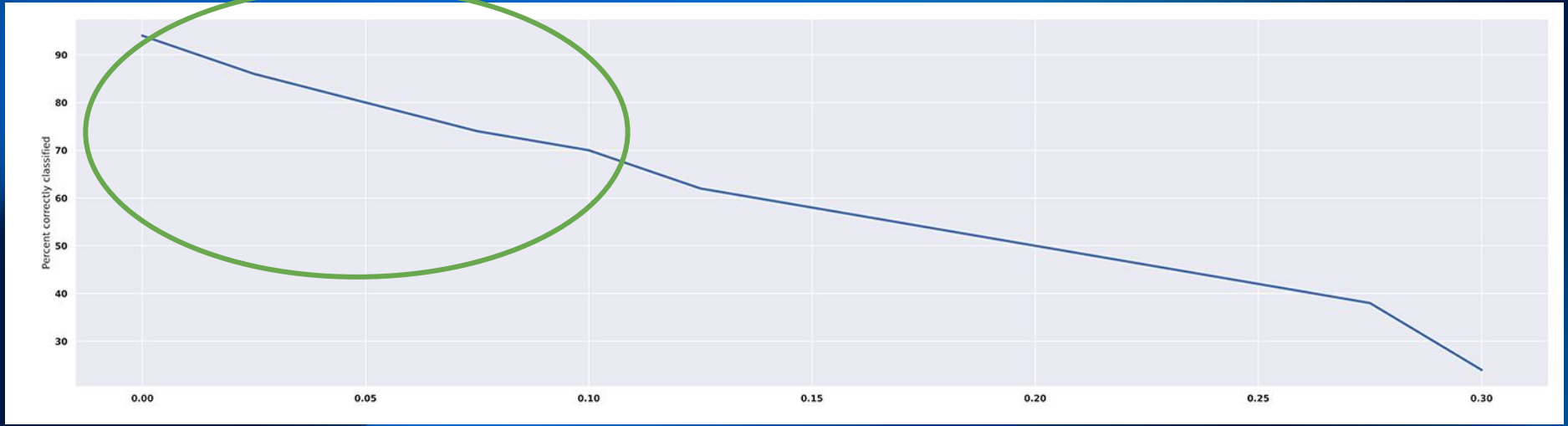
Defense 2: Perturbed Prediction Averaging

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.

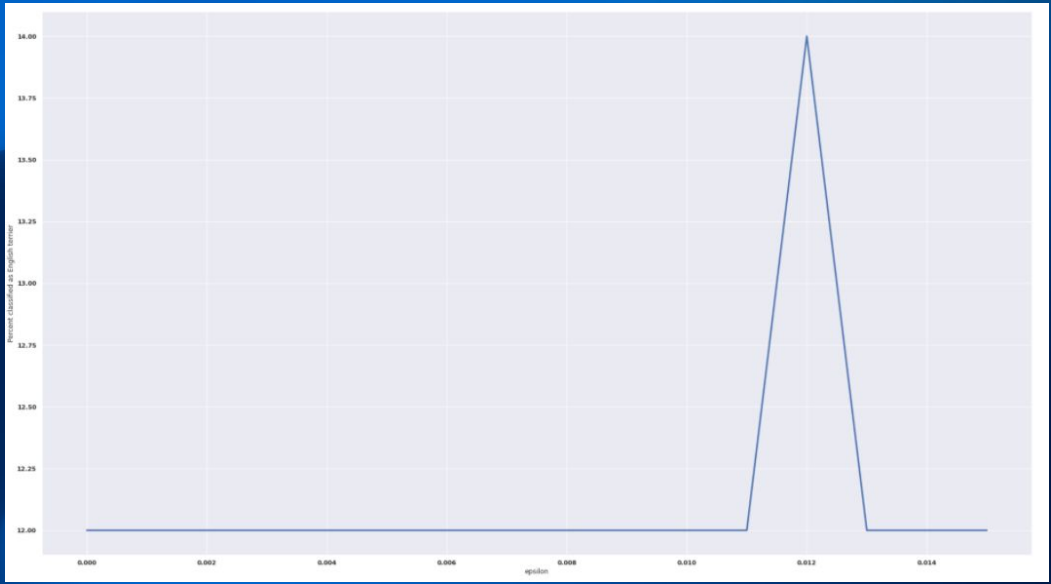
Solution: Add random noise in excess of what we expect from attacks (e.g., enough to be visible). Average over many predictions to "wash out" perturbation of attack.



FGSM Attack Against Prediction Averaging Defense

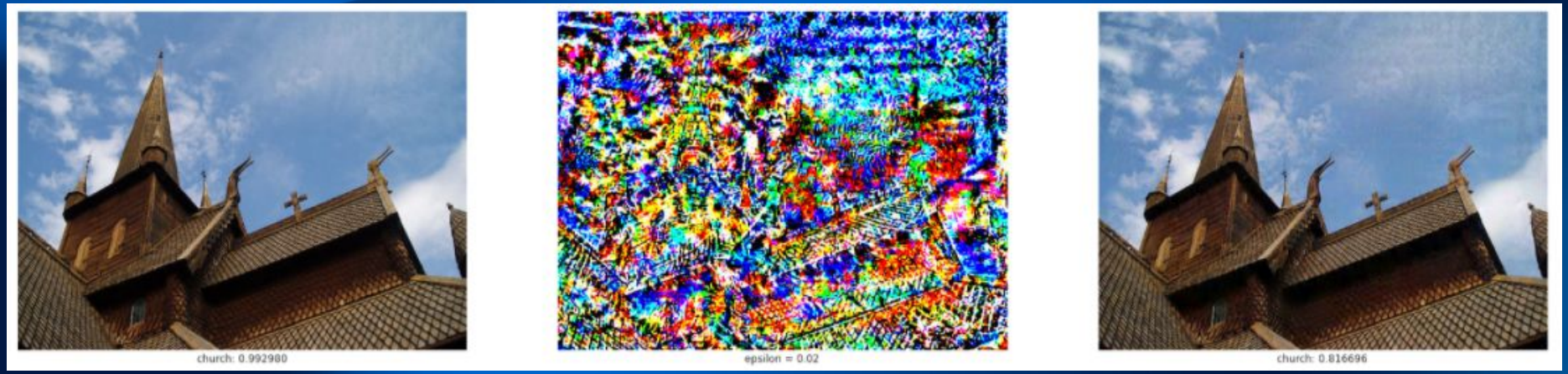


Directed Attack Against Prediction Averaging Defense



What Are the Defenses Doing?

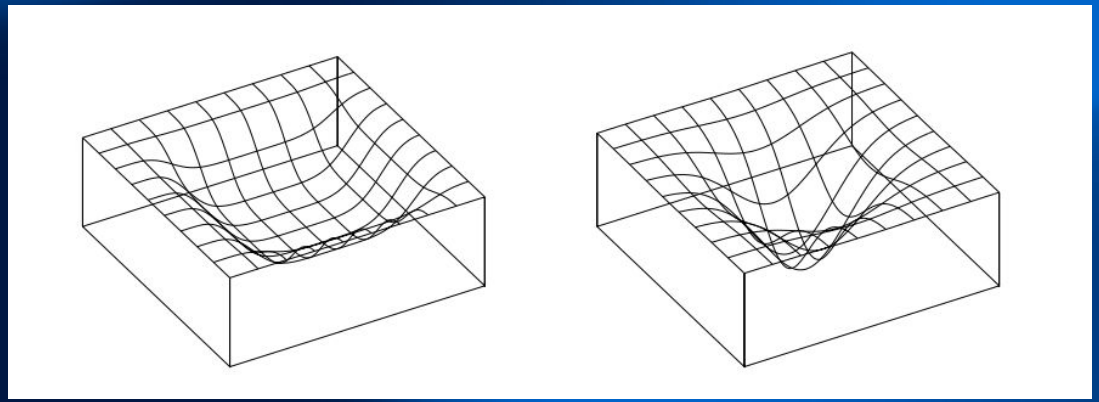
Recall this from earlier...

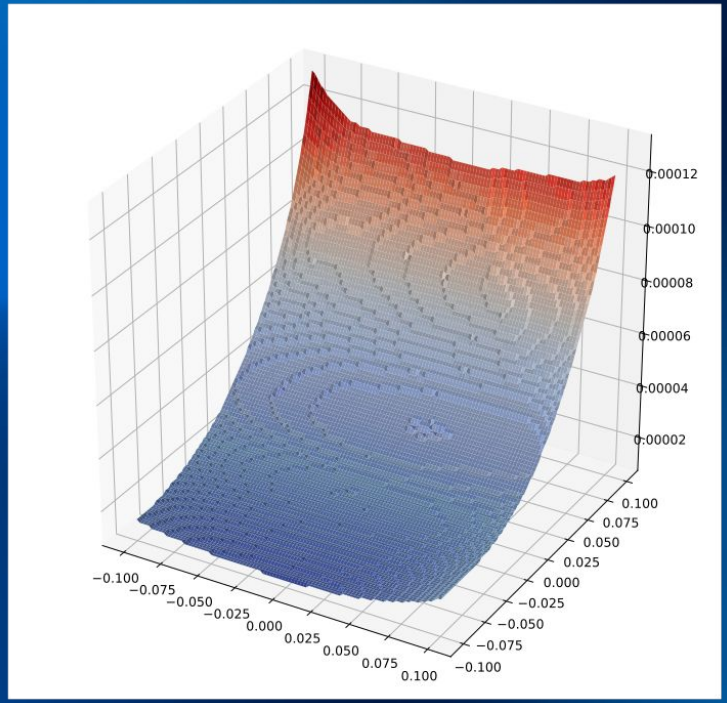


What Are the Defenses Doing?

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

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/