Adversarial Robustness

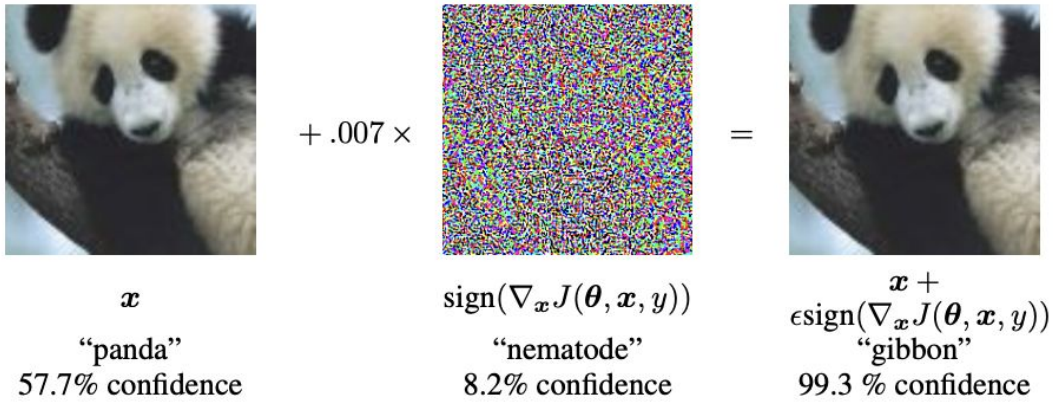
Eric Xie, Yagnik Panguluri, Anders Gyllenhoff, Caroline Gihlstorf

Agenda

- What are Adversarial Examples?
  - Why do they occur?
- Adversarial Attacks
  - Case analysis
  - Implementations extending into the real world
- Defenses against Adversarial Examples
- Alternative Perspectives on Adversarial Robustness

What Are Adversarial Examples?

Examples (e.g., images) perturbed such that they differ minimally from the original example (humans can't perceive the difference) but differ enough to make a classifier assign them the wrong label.



(Goodfellow et al., (2015), Figure 1, page 3)

Intriguing Properties of Neural Networks

Interpreting Properties of Neural Networks

- Often difficult to interpret their properties and mechanisms of decision-making
- Traditionally, computer vision approaches have sought to interpret individual hidden units by finding images that maximize the activation of a specific unit, assuming each unit captures a distinct semantic feature.
    - *Intriguing Properties of Neural Network*, challenges this on the heels of AlexNet (2012)

Unit Analysis

The visual inspection of images x' can be described:

$$x' = \arg\max_{x \in \mathcal{I}} \langle \phi(x), e_i \rangle$$

- I: hold out set of images
- e_i: natural basis vector of the i^{th} hidden unit

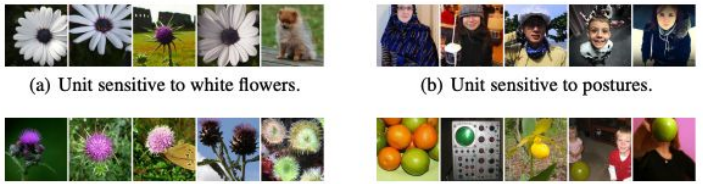
Compared performance between natural basis vector and a random basis vector $v \in R^m$

$$x' = \arg\max_{x \in \mathcal{I}} \langle \phi(x), v \rangle$$

Unit Analysis

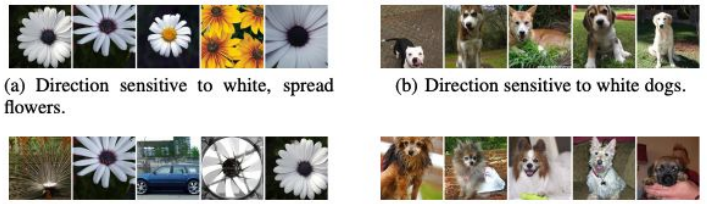
Tested on MNIST & AlexNet

- natural basis is no better than a random basis for inspecting properties of activation layers
- overall space of activations contains the semantic information
- Little practical utility except confirm assumptions on higher level complexity of representations



(a) Unit sensitive to white flowers. (b) Unit sensitive to postures. (c) Unit senstive to round, spiky flowers. (d) Unit senstive to round green or yellow objects.

Figure 3: Experiment performed on ImageNet. Images stimulating single unit most (maximum stimulation in natural basis direction). Images within each row share many semantic properties.



(a) Direction sensitive to white, spread flowers. (b) Direction sensitive to white dogs. (c) Direction sensitive to spread shapes. (d) Direction sensitive to dogs with brown heads.

Figure 4: Experiment performed on ImageNet. Images giving rise to maximum activations in a random direction (maximum stimulation in a random basis). Images within each row share many semantic properties.

Adversarial Attacks

Existence of Adversarial Examples

Output layer of a neural network = highly nonlinear function of the input

- Local generalization: The model assigns high probabilities to small perturbations of training data points ... slightly change an image should still predict the same class.
- Non-local generalization: The model can recognize objects in different contexts even if those regions of input space are far from the training examples.

Does smoothness assumption regarding local generalization hold? Can tiny perturbations fool deep neural networks?

Adversarial Example Generation

Let $f:R^m \rightarrow 1,\ldots,k$ be a classifier that maps an input image $x \in R^m$ (represented as a pixel vector) to a discrete set of labels, where f is assumed to have a continuous loss function.

Goal: For a given image x and a target label $l \in \{1,\ldots,k\}$, find a perturbation $r \in R^m$ s.t. x+r is classified as l (targeted misclassification) while remaining in the valid pixel range.

Minimize $\|r\|_2$ subject to:

1. $f(x+r) = l$
2. $x + r \in [0,1]^m$

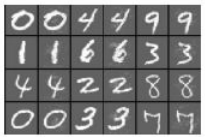
$\longrightarrow$ **Minimize $c|r| + \text{loss}_f(x+r, l)$ subject to $x + r \in [0,1]^m$**

Challenging optimization (nonconvex), but can be approximated using a penalty function and a box-constrained L-BFGS optimizer

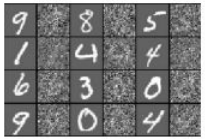
(a) (b)

Figure 5: Adversarial examples generated for AlexNet [9].(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. Plase refer to `http://goo.gl/huaGPb` for full resolution images. The examples are strictly randomly chosen. There is not any postselection involved.

(a) Even columns: adversarial examples for a linear (FC) classifier (stddev=0.06)



(b) Even columns: adversarial examples for a 200-200-10 sigmoid network (stddev=0.063)



(c) Randomly distorted samples by Gaussian noise with stddev=1. Accuracy: 51%.

Figure 7: Adversarial examples for a randomly chosen subset of MNIST compared with randomly distorted examples. Odd columns correspond to original images, and even columns correspond to distorted counterparts. The adversarial examples generated for the specific model have accuracy 0% for the respective model. Note that while the randomly distorted examples are hardly readable, still they are classified correctly in half of the cases, while the adversarial examples are never classified correctly.

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.

Last column: Average minimum distortion necessary to reach 0% accuracy on training set

Columns: show the error (proportion of samples misclassified) on the distorted training sets for different models

Cross-Training-Set Generalization

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.

	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 Instability

Lipschitz constraint used at each layer, which measures how much the layer's output can change relative to its input, to examine the stability of a network

- Results show that instabilities can arise as early as the first convolutional layer, with some layers having significantly higher bounds than others.
- Can penalizing the Lipschitz bounds during training could improve network stability?

How do we defend against adversarial examples and increase model stability?

Discussion

- Given that models will always have some finite amount of computational power, is it ever possible to have an accurate model that is completely immune to adversarial examples?
- Does the existence of adversarial examples expose a *fundamental flaw* in deep learning?

Case Analysis: Designing Adversaries against Defensive Distillation

Defensive Distillation - A False Sense of Security

What is Defensive Distillation?

- A method to increase robustness by training a model on softened labels
- Claimed to reduce attack success rate from **95%** to **0.5%**

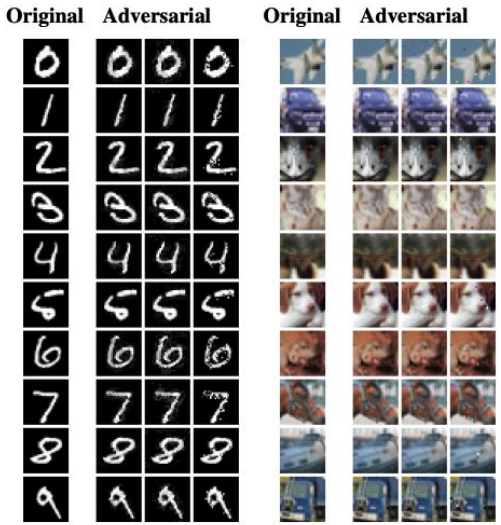


Fig. 1. An illustration of our attacks on a defensively distilled network. The leftmost column contains the starting image. The next three columns show adversarial examples generated by our L_2, L_∞, and L_0 algorithms, respectively. All images start out classified correctly with label l, and the three misclassified instances share the same misclassified label of $l+1 \pmod{10}$. Images were chosen as the first of their class from the test set.

Distance Metrics

There are three widely-used distance metrics in the literature for generating adversarial examples, all of which are Lp norms.

$$\|v\|_p = \left(\sum_{i=1}^{n} |v_i|^p \right)^{\frac{1}{p}}.$$

- 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_2 distance measures the standard Euclidean (RMS) distance between x and x'. The L_2 distance can remain small when there are many small changes to many pixels.
- L_{inf} distance measures the maximum change to any of the coordinates.

Adversarial Attacks

- **L_0 Attack** - Finds the **minimum number of pixels** that must be changed to induce misclassification, making changes as sparse as possible
- **L_2 Attack** - Creates **small but distributed perturbations** that are imperceptible to humans but shift the model's prediction
- **L_{inf} Attack** - Ensures that no individual pixel is changed by more than a certain threshold, often leading to subtle but widespread changes

Different attack methods exploit different weaknesses in neural networks, but **no single defense can protect against all distance-based attacks.**

Experimental Setup

- Models trained on MNIST and CIFAR-10
- MNIST achieved **99.5% accuracy** and CIFAR achieved **80% accuracy**
- Pre-trained **Inception v3** network with **96% top-5 accuracy**

Layer Type	MNIST Model	CIFAR Model
Convolution + ReLU	3×3×32	3×3×64
Convolution + ReLU	3×3×32	3×3×64
Max Pooling	2×2	2×2
Convolution + ReLU	3×3×64	3×3×128
Convolution + ReLU	3×3×64	3×3×128
Max Pooling	2×2	2×2
Fully Connected + ReLU	200	256
Fully Connected + ReLU	200	256
Softmax	10	10

TABLE I
MODEL ARCHITECTURES FOR THE MNIST AND CIFAR MODELS. THIS ARCHITECTURE IS IDENTICAL TO THAT OF THE ORIGINAL DEFENSIVE DISTILLATION WORK. [39]

Parameter	MNIST Model	CIFAR Model
Learning Rate	0.1	0.01 (decay 0.5)
Momentum	0.9	0.9 (decay 0.5)
Delay Rate	-	10 epochs
Dropout	0.5	0.5
Batch Size	128	128
Epochs	50	50

TABLE II
MODEL PARAMETERS FOR THE MNIST AND CIFAR MODELS. THESE PARAMETERS ARE IDENTICAL TO THAT OF THE ORIGINAL DEFENSIVE DISTILLATION WORK. [39]

Experimental Results

	Best Case				Average Case				Worst Case			
	MNIST		CIFAR		MNIST		CIFAR		MNIST		CIFAR	
	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob
Our L_0	8.5	100%	5.9	100%	16	100%	13	100%	33	100%	24	100%
JSMA-Z	20	100%	20	100%	56	100%	58	100%	180	98%	150	100%
JSMA-F	17	100%	25	100%	45	100%	110	100%	100	100%	240	100%
Our L_2	1.36	100%	0.17	100%	1.76	100%	0.33	100%	2.60	100%	0.51	100%
Deepfool	2.11	100%	0.85	100%	–	-	–	-	–	-	–	-
Our L_∞	0.13	100%	0.0092	100%	0.16	100%	0.013	100%	0.23	100%	0.019	100%
Fast Gradient Sign	0.22	100%	0.015	99%	0.26	42%	0.029	51%	–	0%	0.34	1%
Iterative Gradient Sign	0.14	100%	0.0078	100%	0.19	100%	0.014	100%	0.26	100%	0.023	100%

TABLE IV

COMPARISON OF THE THREE VARIANTS OF TARGETED ATTACK TO PREVIOUS WORK FOR OUR MNIST AND CIFAR MODELS. WHEN SUCCESS RATE IS NOT 100%, THE MEAN IS ONLY OVER SUCCESSES.

	Untargeted		Average Case		Least Likely	
	mean	prob	mean	prob	mean	prob
Our L_0	48	100%	410	100%	5200	100%
JSMA-Z	-	0%	-	0%	-	0%
JSMA-F	-	0%	-	0%	-	0%
Our L_2	0.32	100%	0.96	100%	2.22	100%
Deepfool	0.91	100%	-	-	-	-
Our L_∞	0.004	100%	0.006	100%	0.01	100%
FGS	0.004	100%	0.064	2%	-	0%
IGS	0.004	100%	0.01	99%	0.03	98%

TABLE V

COMPARISON OF THE THREE VARIANTS OF TARGETED ATTACK TO PREVIOUS WORK FOR THE INCEPTION V3 MODEL ON IMAGENET. WHEN SUCCESS RATE IS NOT 100%, THE MEAN IS ONLY OVER SUCCESSES.

Attack Evaluation and Comparison

- Comparison Against Prior Attacks
  - Re-implementation of DeepFool, FGSM, Iterative Gradient Sign, and JSMA
  - New attacks outperform previous methods in all three distance metrics
  - JSMA fails on ImageNet due to computational cost
- Success Rates Across Datasets
  - MNIST and CIFAR - 100% success for all attack methods
  - ImageNet - The Linf attack can change an image's classification by flipping the lowest bit of each pixel
  - L0 and L2 attacks require 2x to 10x fewer modifications than previous attacks

Visualization of Attacks

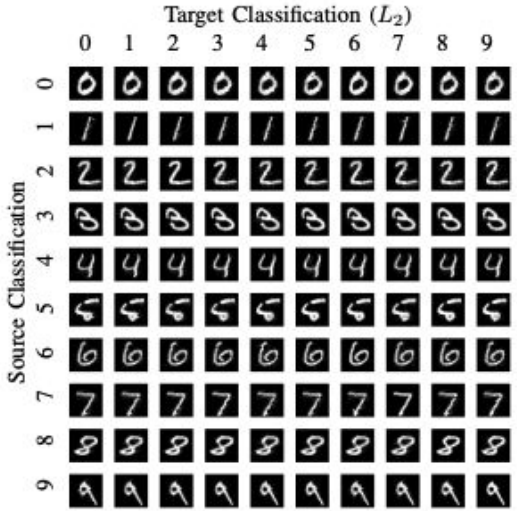


Fig. 3. Our L_2 adversary applied to the MNIST dataset performing a targeted attack for every source/target pair. Each digit is the first image in the dataset with that label.

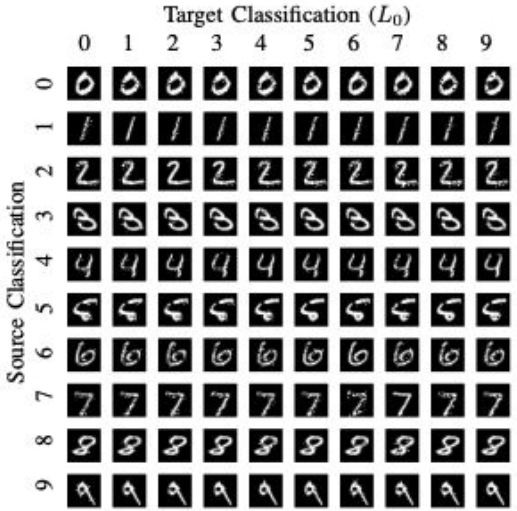


Fig. 4. Our L_0 adversary applied to the MNIST dataset performing a targeted attack for every source/target pair. Each digit is the first image in the dataset with that label.

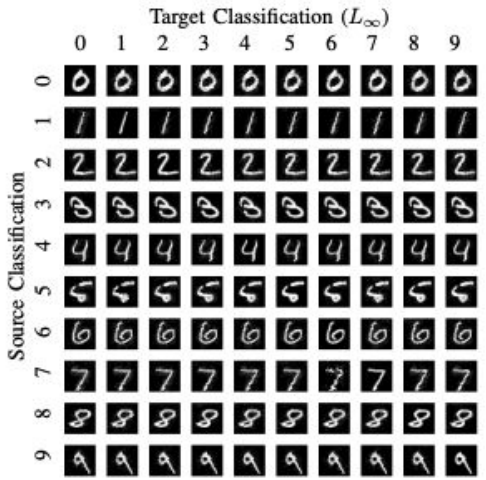


Fig. 5. Our L_∞ adversary applied to the MNIST dataset performing a targeted attack for every source/target pair. Each digit is the first image in the dataset with that label.

Role of Transferability

- Even if a model is trained against adversarial attacks, adversarial examples often remain effective on similar models
- High-confidence adversarial examples (crafted to be misclassified with high certainty) bypass distillation defenses
- Any proposed defense must prove it can break transferability

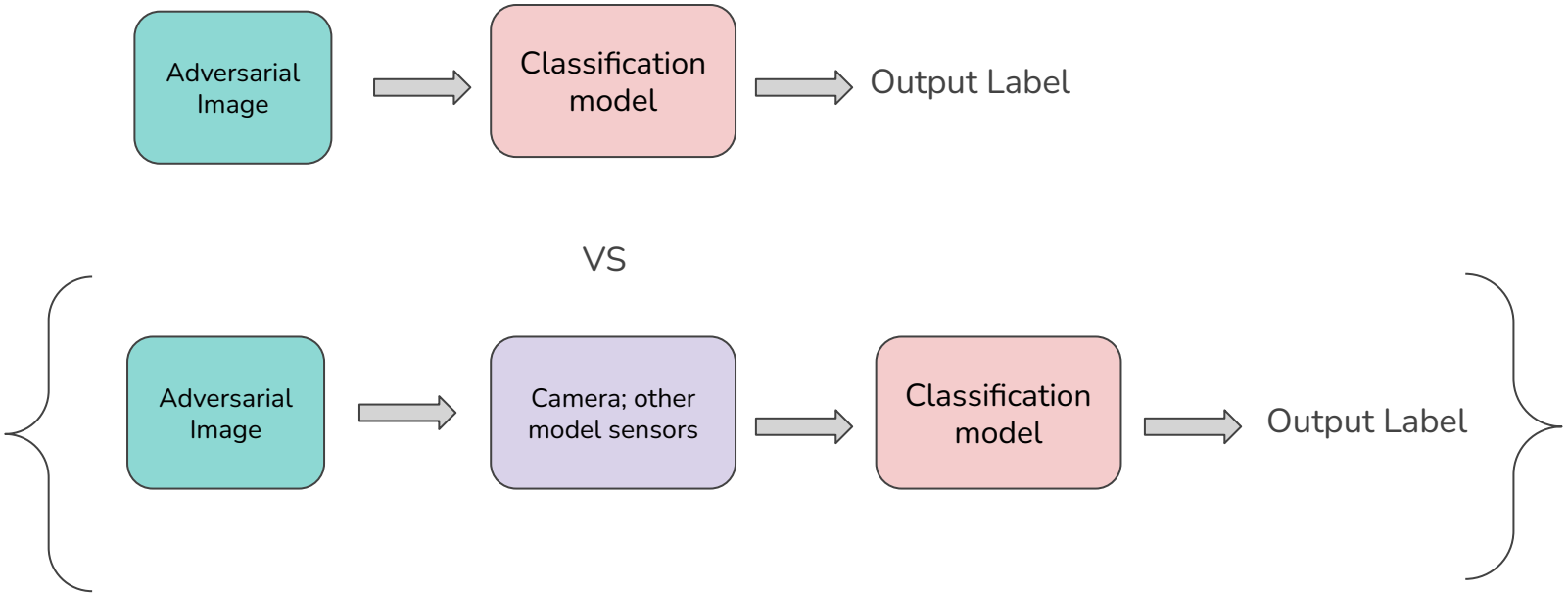
Why Do These Attacks Work?

- The Local Linearity Hypothesis
  - Neural networks are highly non-linear but behave linearly in small regions
- Adversarial perturbations scale with dimensionality
  - Small, imperceptible changes in high-dimensional spaces remain effective

Adversarial robustness cannot rely solely on distillation

Real-World Adversarial Examples

Image-to-Model Pipelines Can Differ



Research Pipeline:

Research Pipeline:

Generate Adversarial Images

Fast Method: maximize cost of true label

$$X^{adv} = X + \epsilon \operatorname{sign}\left(\nabla_X J(X, y_{true})\right)$$ (Kurakin et al., (2017), page 4)

Basic Iterative Method: maximize cost of true label over multiple iterations

$$X_0^{adv} = X, \quad X_{N+1}^{adv} = Clip_{X,\epsilon}\left\{X_N^{adv} + \alpha \operatorname{sign}\left(\nabla_X J(X_N^{adv}, y_{true})\right)\right\}$$ (Kurakin et al., (2017), page 4)

Iterative Least-Likely Class Method: minimize cost of least likely label over multiple iterations

$$X_0^{adv} = X, \quad X_{N+1}^{adv} = Clip_{X,\epsilon}\left\{X_N^{adv} - \alpha \operatorname{sign}\left(\nabla_X J(X_N^{adv}, y_{LL})\right)\right\}$$ (Kurakin et al., (2017), page 5)

Generate Adversarial Images

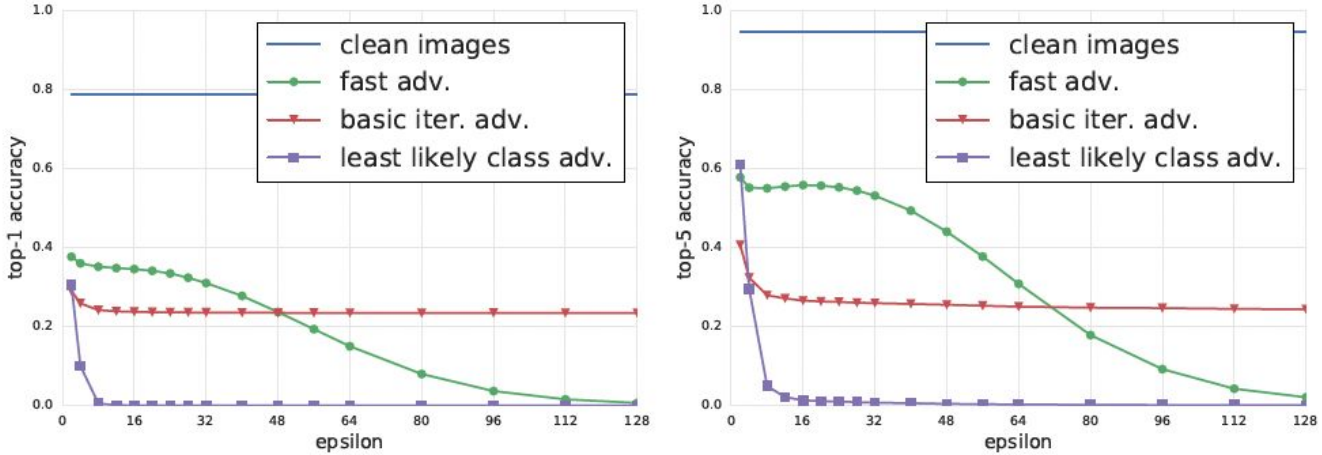


Figure 2: Top-1 and top-5 accuracy of Inception v3 under attack by different adversarial methods and different ϵ compared to "clean images" — unmodified images from the dataset. The accuracy was computed on all 50,000 validation images from the ImageNet dataset. In these experiments ϵ varies from 2 to 128.

(Kurakin et al., (2017), Figure 2, page 5)

Least likely class iterative method performs best wrt reducing model accuracy

Research Pipeline:

Apply Real-World Transformations

Experiment 1 - Photography Transformations:

(a) Printout (b) Photo of printout (c) Cropped image

Figure 3: Experimental setup: (a) generated printout which contains pairs of clean and adversarial images, as well as QR codes to help automatic cropping; (b) photo of the printout made by a cellphone camera; (c) automatically cropped image from the photo.

(Kurakin et al., (2017), Figure 3, page 6)

- Print photos of adversarial and non-adversarial images
- Take photos of these printouts
- Input photos into the classification model

Apply Real-World Transformations

Experiment 1 - Photography Transformations:

- 2 Experimental Scenarios:
    - Test on random set of examples
    - Test on a select set of examples: model correctly classifies non-adversarial images but misclassifies adversarial images

Apply Real-World Transformations

Experiment 2 - Additional Transformations:

- Test on additional transformations: brightness, contrast, blur, noise, and JPEG encoding

Research Pipeline:

Measure Adversarial Image Retention After Transformation

$$d = \frac{\sum_{k=1}^{n} C(\boldsymbol{X}^k, y_{true}^k)\overline{C(\boldsymbol{X}_{adv}^k, y_{true}^k)}C(T(\boldsymbol{X}_{adv}^k), y_{true}^k)}{\sum_{k=1}^{n} C(\boldsymbol{X}^k, y_{true}^k)\overline{C(\boldsymbol{X}_{adv}^k, y_{true}^k)}} \quad (1)$$

(Kurakin et al., (2017), Equation 1, page 6)

Destruction rate (*d*): measures how many adversarial images lose their adversarial nature after a transformation (i.e., the classifier gets them right after the transformation)

Results - Photographed Images

- Fast Generation method lost the fewest adversarial examples
- Destruction rate was sometimes lower for randomly sampled images
- Randomly selected images from the basic iterative and iterative least-likely class methods did better at remaining adversarial than adversarially cherry-picked images

Results - Other Transformations

- Fast Generation method lost the fewest adversarial examples

- Most effective transformations: Gaussian blur, Gaussian noise, and JPEG encoding

- Takeaway: adversarial examples are not absent from real-world settings

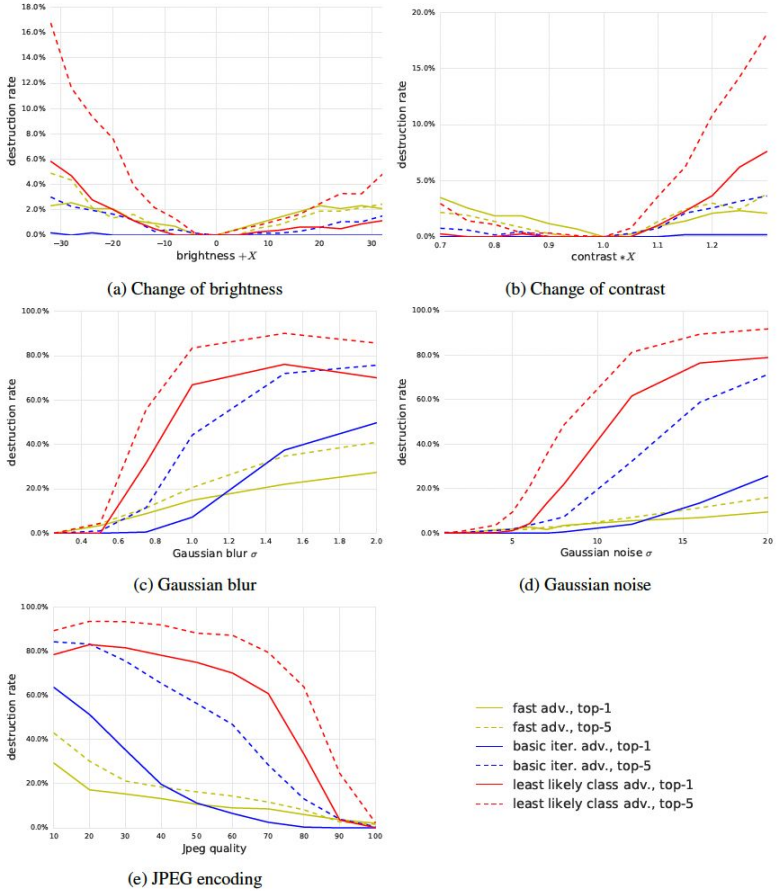


Figure 6: Comparison of adversarial destruction rates for various adversarial methods and types of transformations. All experiments were done with $\epsilon = 16$.

(Kurakin et al., (2017), Figure 6, page 14)

Transferability to Another Model

Discussion

- Besides taking photos of printed images and using the other adjustments (e.g., brightness/contrast/noise, etc) mentioned in the presentation, what other “real-world transformations” can you think of applying to adversarial examples (either in computer vision or in another domain entirely)?

- Although some adversarial examples are classified correctly after going through real-world transformations, others still cause the classifier to misclassify them. Do you think this could be attributed simply to randomness or could there be a shared property of these image that make them particularly robust to real-world transformations?

- Given that adversarial examples can successfully fool neural networks across multiple architectures, what are some limitations of current deep learning models that make them so vulnerable, and do you think true adversarial robustness is possible, or is this an inherent flaw of neural networks?

Defending Against Adversarial Examples

A Linear Explanation of Adversarial Examples

- Input features have limited precision (e.g., 8 bits/pixel)
- Perturbations smaller than this precision threshold $||\eta||_\infty < \epsilon$ should be ignored, but, in high dimensions, small, coordinated perturbations aligned with the model's weight vectors and amplify linearly across features.
    - dot product $w^\top \eta$ grows proportionally to the dimensionality

Neural networks are non linear, but their architectural choices (ReLUs, LSTMs, maxout units) prioritize linear like behavior to ease optimization → Can craft adversarial perturbations via linear approximations

Fast Gradient Sign Method

Adversarial examples are generated via:

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

where

- θ: parameters of the model
- x: input
- y: targets associated with x
- ϵ: Perturbation magnitude
- $\nabla_x J$: Gradient of the loss with respect to the input.

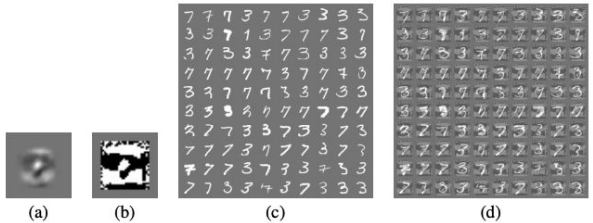


Figure 2: The fast gradient sign method applied to logistic regression (where it is not an approximation, but truly the most damaging adversarial example in the max norm box). a) The weights of a logistic regression model trained on MNIST. b) The sign of the weights of a logistic regression model trained on MNIST. This is the optimal perturbation. Even though the model has low capacity and is fit well, this perturbation is not readily recognizable to a human observer as having anything to do with the relationship between 3s and 7s. c) MNIST 3s and 7s. The logistic regression model has a 1.6% error rate on the 3 versus 7 discrimination task on these examples. d) Fast gradient sign adversarial examples for the logistic regression model with $\epsilon = .25$. The logistic regression model has an error rate of 99% on these examples.

Model Type	Perturbation Magnitude (ε)	Error on Adversarial Examples (%)	Avg. Confidence on Misclassified Examples (%)
Shallow Softmax (MNIST)	0.25	99.9%	79.3
Maxout Network (MNIST)	0.25	97.6%	97.6
Conv Maxout (CIFAR-10)	0.1	87.15%	96.6

Effectiveness of Fast Gradient Sign Method on Different Models

Why does this work?

Deep networks can theoretically resist adversarial examples via the universal approximator theorem, but standard training fails to enforce it.

Can instead use FGSM as an effective regularizer, continually updating the supply of adversarial examples:

$$\sim J(\theta,x,y)=\alpha J(\theta,x,y)+(1-\alpha)J(\theta,x+\eta,y) \qquad \eta=\epsilon\cdot\text{sign}(\nabla_x J(\theta,x,y))$$

What about the model learning how adversaries will react to changes in weights?

Effectiveness

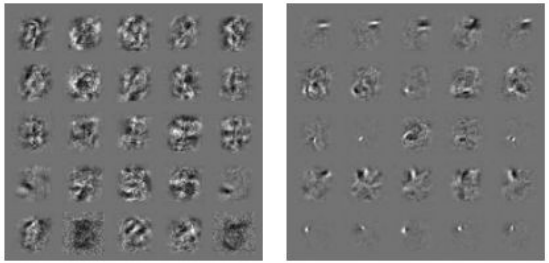


Figure 3: Weight visualizations of maxout networks trained on MNIST. Each row shows the filters for a single maxout unit. Left) Naively trained model. Right) Model with adversarial training.

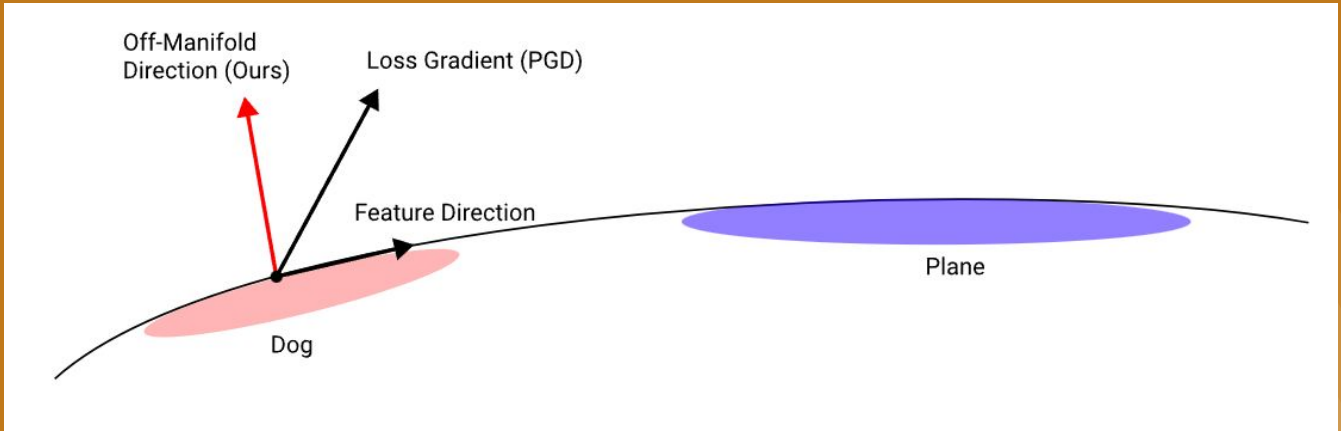
Model Type Test Error (%)	Test Error (%)	Error on Adversarial Examples (%)	Confidence on Misclassified Examples (%)
Standard Model	0.94	89.4	97.6
Adversarially Trained Model	0.84	17.9	81.4

Robustness Impact with Adversarial Training

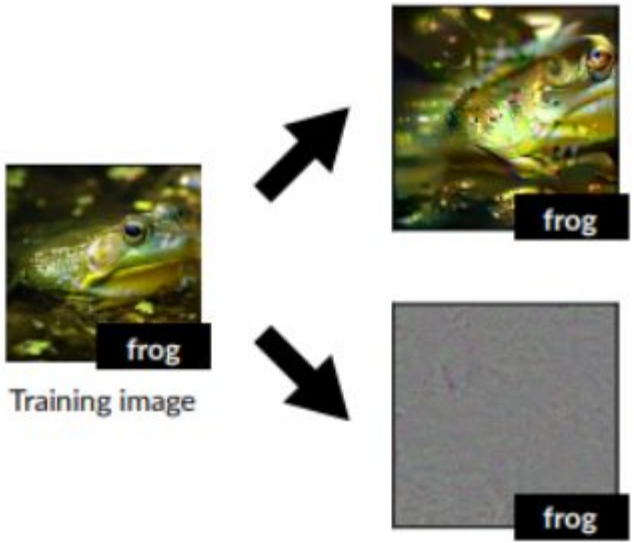
Regularization Method	Error on Fast Gradient Sign Adversarial Examples (%)	Confidence on Misclassified Examples (%)
Adding Random $\pm\varepsilon$ Noise to Pixels	86.2	97.3
Adding $U(-\varepsilon, \varepsilon)$ Noise to Pixels Trained Model	90.4	97.8

Robustness of Alternative Adversarial Training Methods vs FGSM Samples

A Different Understanding of Adversarial Examples (Features, not bugs)



Discussion



Do models and humans rely on the same types of features to classify images?

An Alternate Perspective of Robustness

- Classifiers are trained to identify features within the data that will generalize across members in each class.

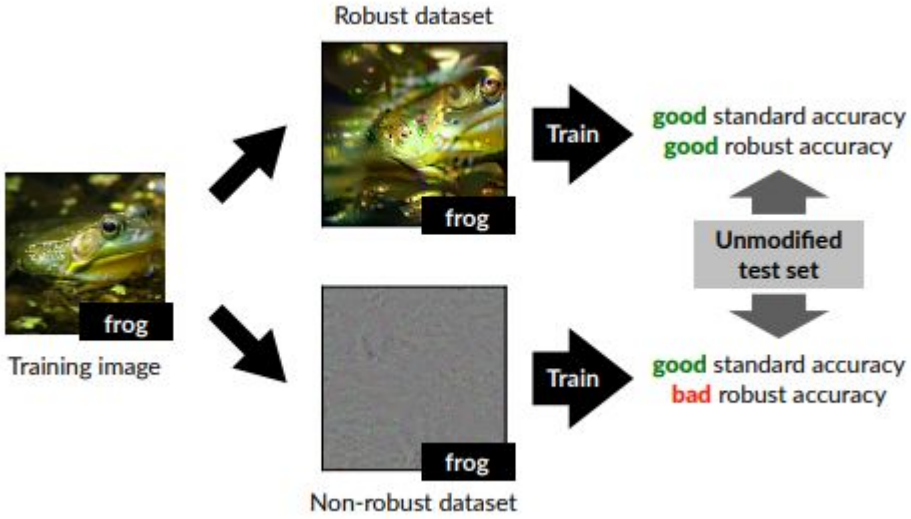
Robust features: Recognizable to humans
Non-robust features: Highly informative for a model, but imperceptible to humans

- Can features within the data that are useful for models but not humans be extracted and analyzed independently?

Robust vs Non-Robust Separation

If we can disentangle robust and non-robust features within the dataset, we should expect:

- Training on robust features gives some level of adversarial robustness
- Training on non-robust features still retains predictive utility



Robust Dataset

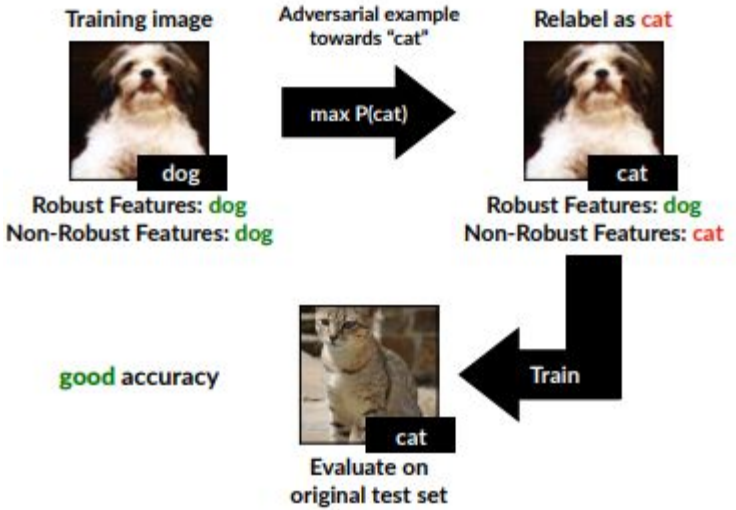
Remove all predictive non-robust features from each input

- Take an adversarially trained model's outputs for a given input
- Adjust a randomly selected non-correlated image until the adv. trained model's output matches

Non-Robust Dataset

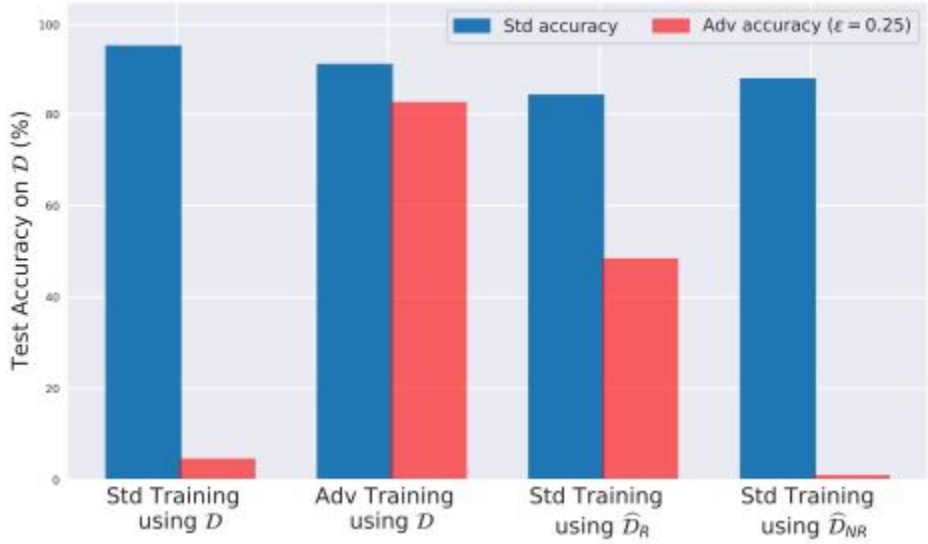
Remove all predictive robust features from each input

- Switch the label of each input
- Add adversarial perturbations until a standard model predicts the new label



Robust vs Non-Robust

- Training on the robust dataset gives some level of adversarial robustness
- Training on the non-robust dataset actually gives higher accuracy on the test set



Why Does This Matter?

Suppose there is a distinction between robust and non-robust features within the data, what does this tell us?

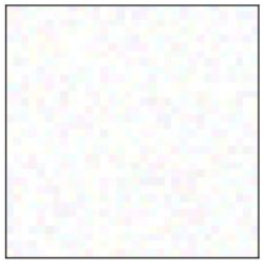
- What happens to the model when it learns robust vs non-robust features?

Through this lens, we can better analyze model robustness and behavior with respect to adversarial examples

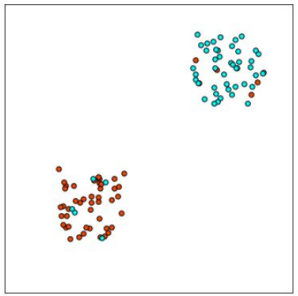
Are Adversarial Examples always Features?

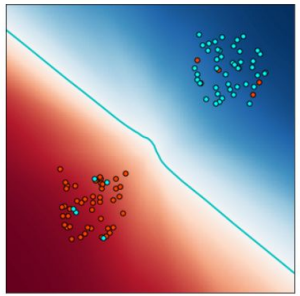
- Dataset containing only robust features (black and white) with added:
  - Random noise
  - Label noise
- With a robust linear classifier, a significant portion of each sample can be altered without affecting classification ($\epsilon = 0.9$)
- With a deep neural network, $\epsilon = 0.01$ is enough to flip almost all labels

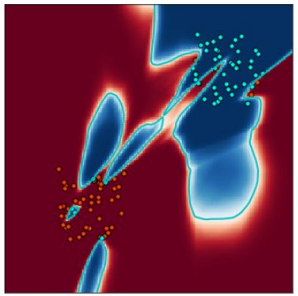
Y = -1



Y = +1



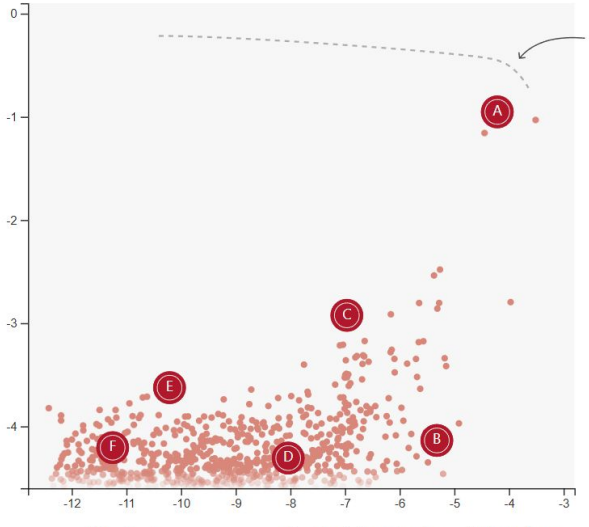


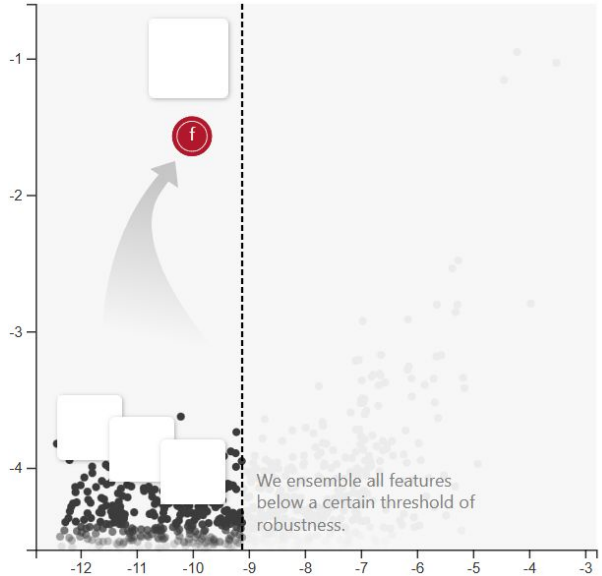


Adversarial Example Transferability

Non-robust features are inherent to the underlying data distribution

- Regardless of the model size/architecture, it is possible for models to simply learn the same useful (non-robust) features as each other





Discussion

- Should we continue to use non-robust features for our predictions?
  - Predictive power vs robustness and interpretability
- How does the separation between robust and non-robust features affect biases within the model?
  - If a model classifies only using robust features, does this remove subtleties essential for utility with respect to underrepresented groups?
  - If a model classifies using non-robust features, will there be human-unintelligible biases within the predictions as well?

Thank You!

References

Carlini, Nicholas, and David Wagner. “Towards Evaluating the Robustness of Neural Networks,” 2017. https://arxiv.org/abs/1608.04644.

Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. “Explaining and Harnessing Adversarial Examples,” 2015. https://arxiv.org/abs/1412.6572.

Ilyas, Andrew, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. “Adversarial Examples Are Not Bugs, They Are Features,” 2019. https://arxiv.org/abs/1905.02175.

Kurakin, Alexey, Ian Goodfellow, and Samy Bengio. “Adversarial Examples in the Physical World,” 2017. https://arxiv.org/abs/1607.02533.

Szegedy, Christian, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. “Intriguing Properties of Neural Networks,” 2014. https://arxiv.org/abs/1312.6199.

Appendix

Adv Training vs Weight Decay

Adv Training more effective than weight decay

- For logistic regression $P(y=1)=\sigma(w^\top x+b)$ adversarial training minimizes $E_{x,y\sim p_{data}}\zeta(y(\epsilon||w||_1-w^\top x-b))$, where $\zeta(z)=\log(1+\exp(z))$
- adversarial training subtracts $\epsilon||w||_1$ from activations, while L^1 adds penalties to the loss
    - adversarial penalties to vanish if predictions become confident, L^1 remains aggressive.

Adv Training vs Weight Decay Results

Weight decay tends to overestimate adversarial damage, especially in deep networks, requiring a smaller coefficient. In Maxout networks on MNIST, FGSM with ϵ=0.25 showed positive results, but even a weight decay of 0.0025 caused a stagnant 5% training error.

L1 regularization can approximate adversarial effects in binary logistic regression, allowing training but offering no regularization benefit and sometimes failing. In multiclass softmax regression, weight decay is even harsher since adversarial perturbations must simultaneously affect multiple weight vectors, leading to conflicts in high-dimensional space.

Adversarial training always minimizes the worst-case loss over perturbed inputs.