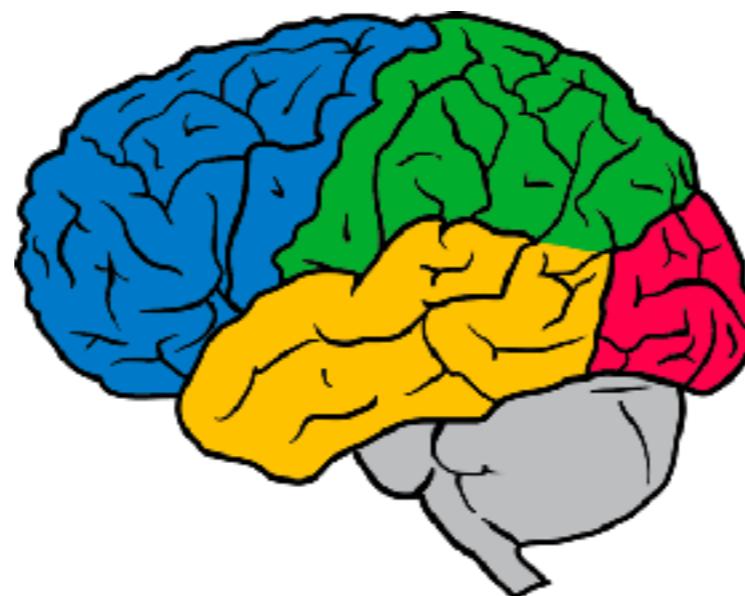


# Adversarial Examples and Adversarial Training

Ian Goodfellow, Staff Research Scientist, Google Brain

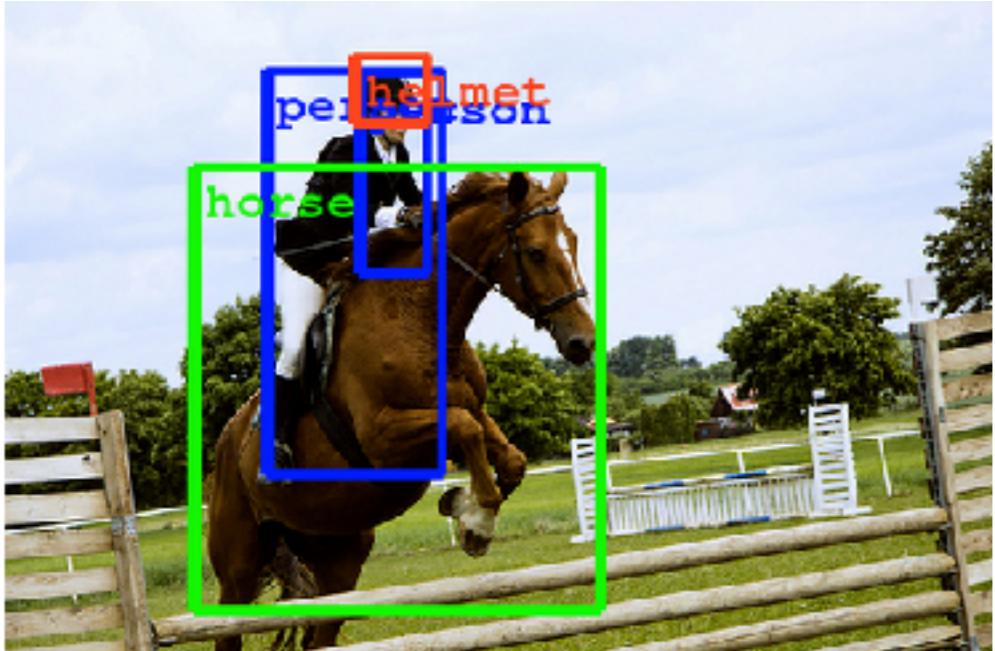
CS 231n, Stanford University, 2017-05-30



# Overview

- What are adversarial examples?
- Why do they happen?
- How can they be used to compromise machine learning systems?
- What are the defenses?
- How to use adversarial examples to improve machine learning, even when there is no adversary

Since 2013, deep neural networks have matched human performance at...



(Szegedy et al, 2014)

...recognizing objects  
and faces....



(Taigmen et al, 2013)



(Goodfellow et al, 2013)

...solving CAPTCHAS and  
reading addresses...



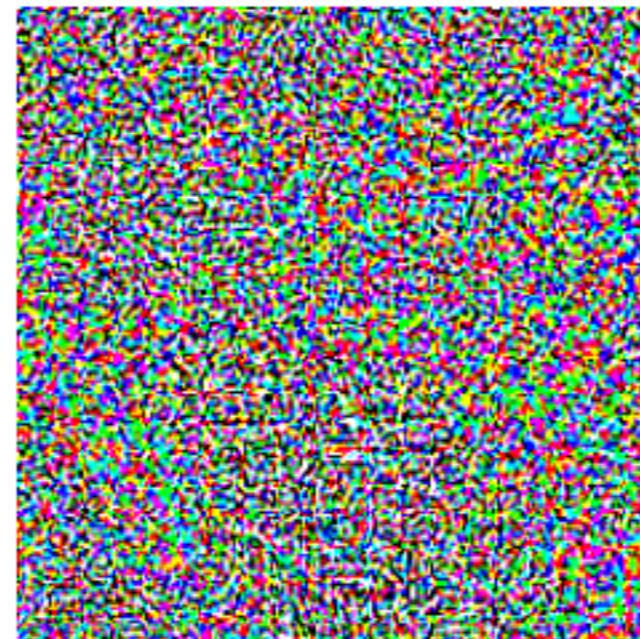
(Goodfellow et al, 2013)

and other tasks...

# Adversarial Examples



+ .007 ×



=



Timeline:

“Adversarial Classification” Dalvi et al 2004: fool spam filter

“Evasion Attacks Against Machine Learning at Test Time”

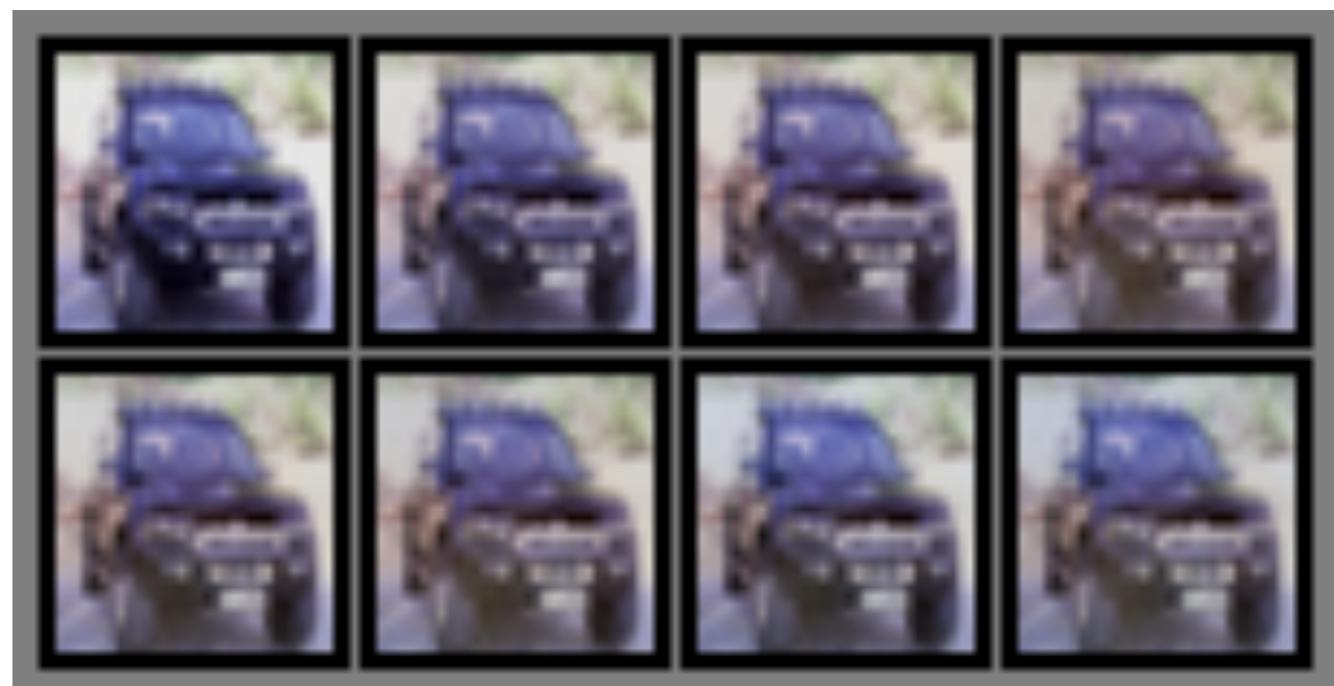
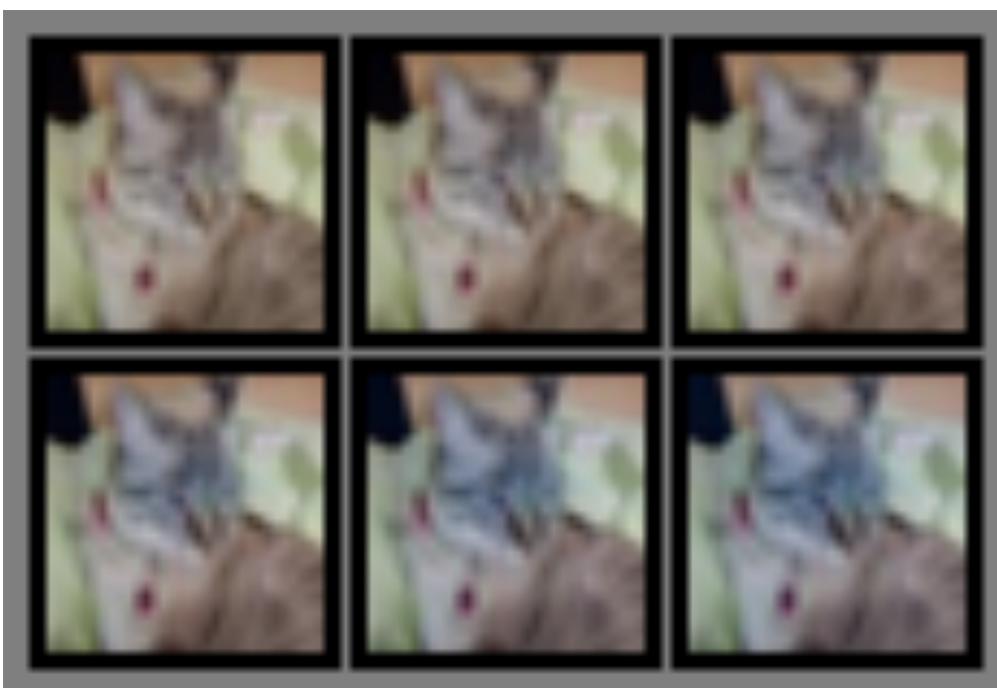
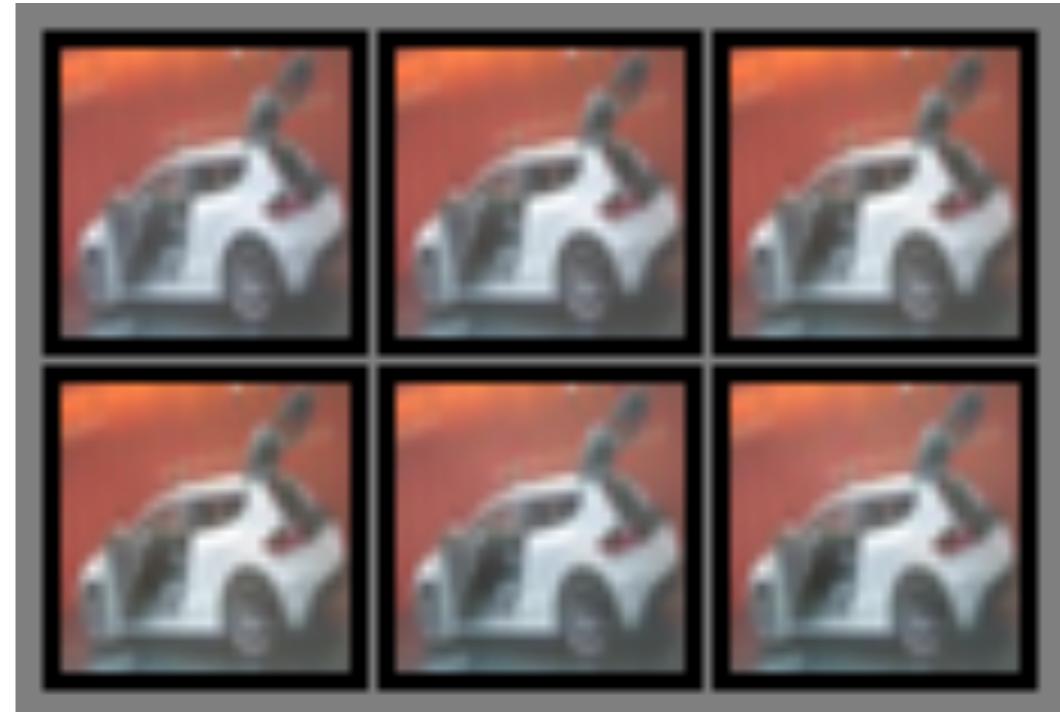
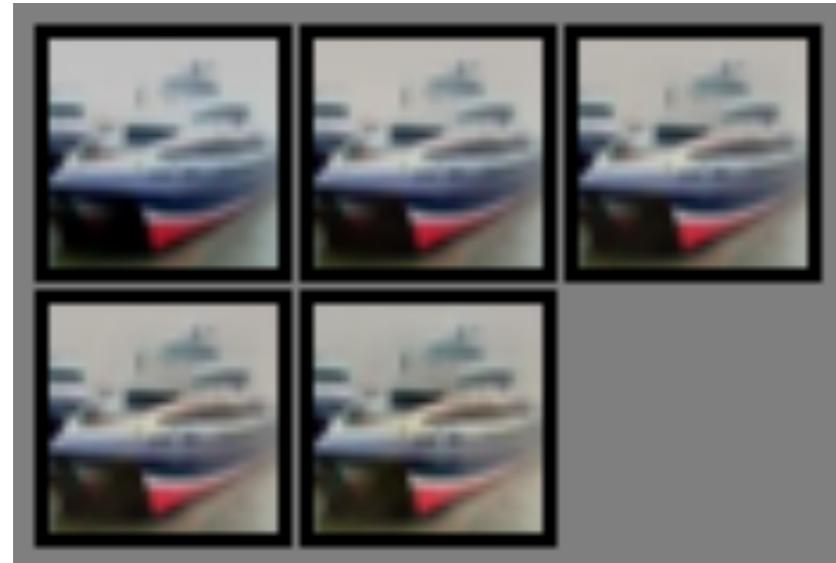
Biggio 2013: fool neural nets

Szegedy et al 2013: fool ImageNet classifiers imperceptibly

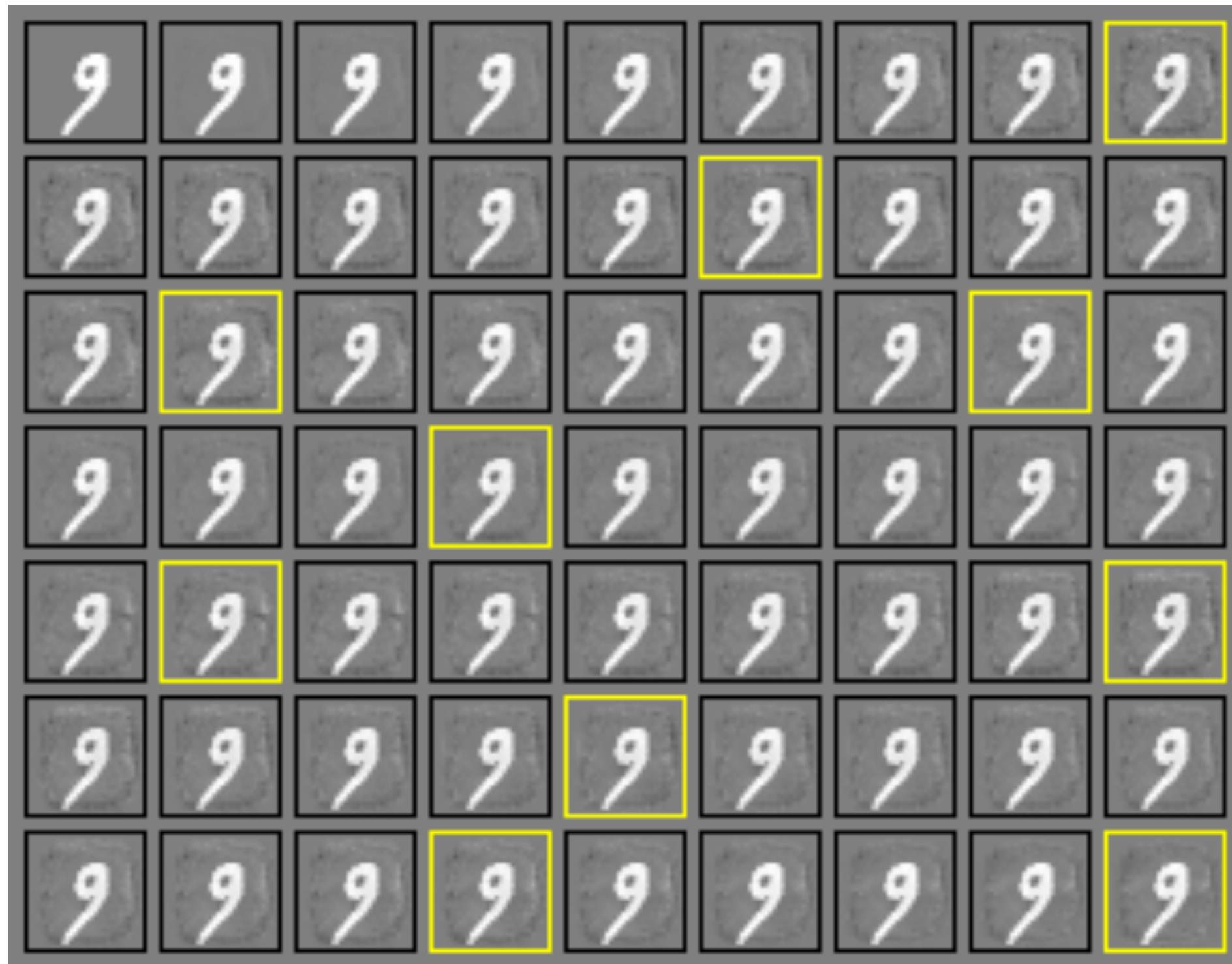
Goodfellow et al 2014: cheap, closed form attack

(Goodfellow 2016)

# Turning Objects into “Airplanes”



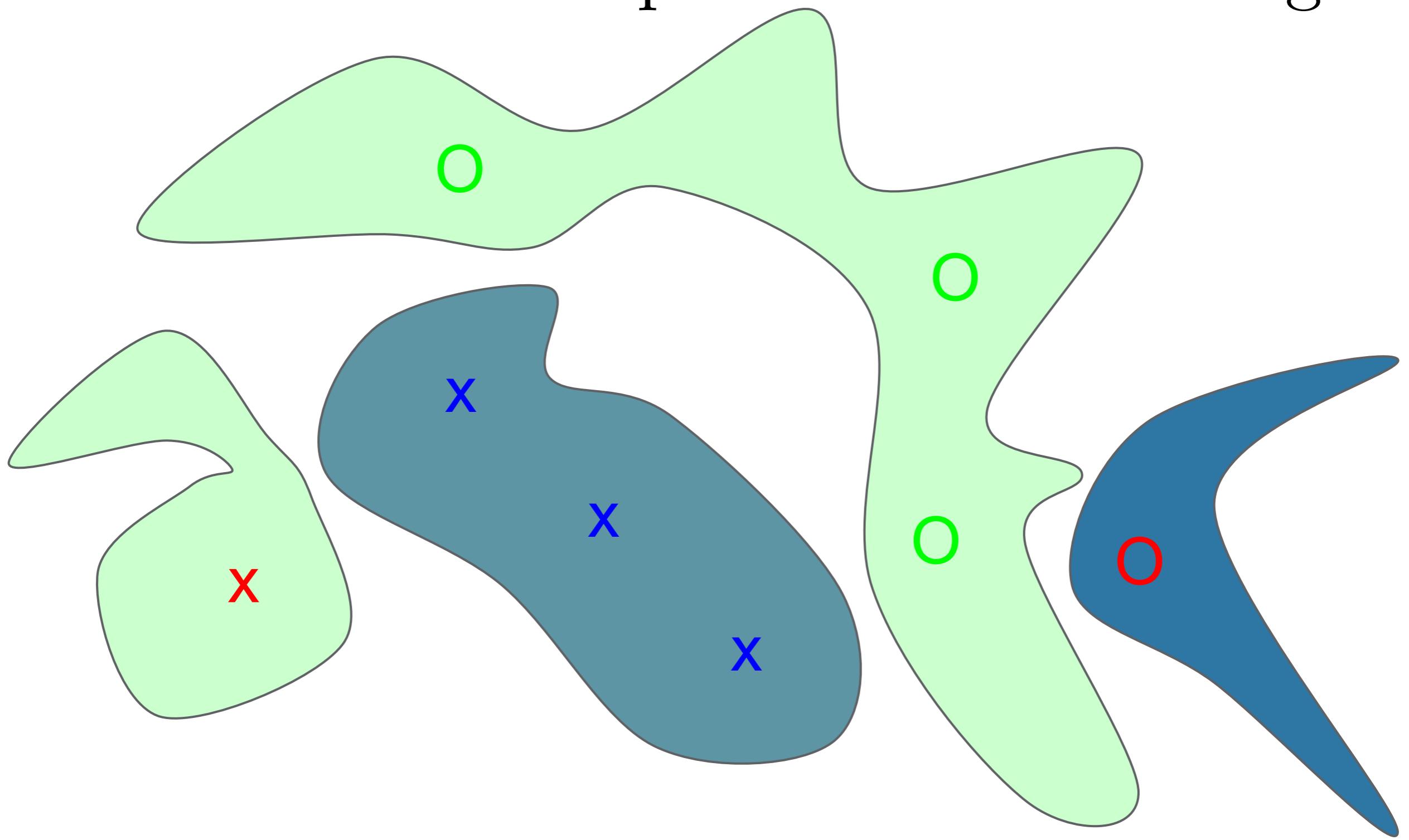
# Attacking a Linear Model



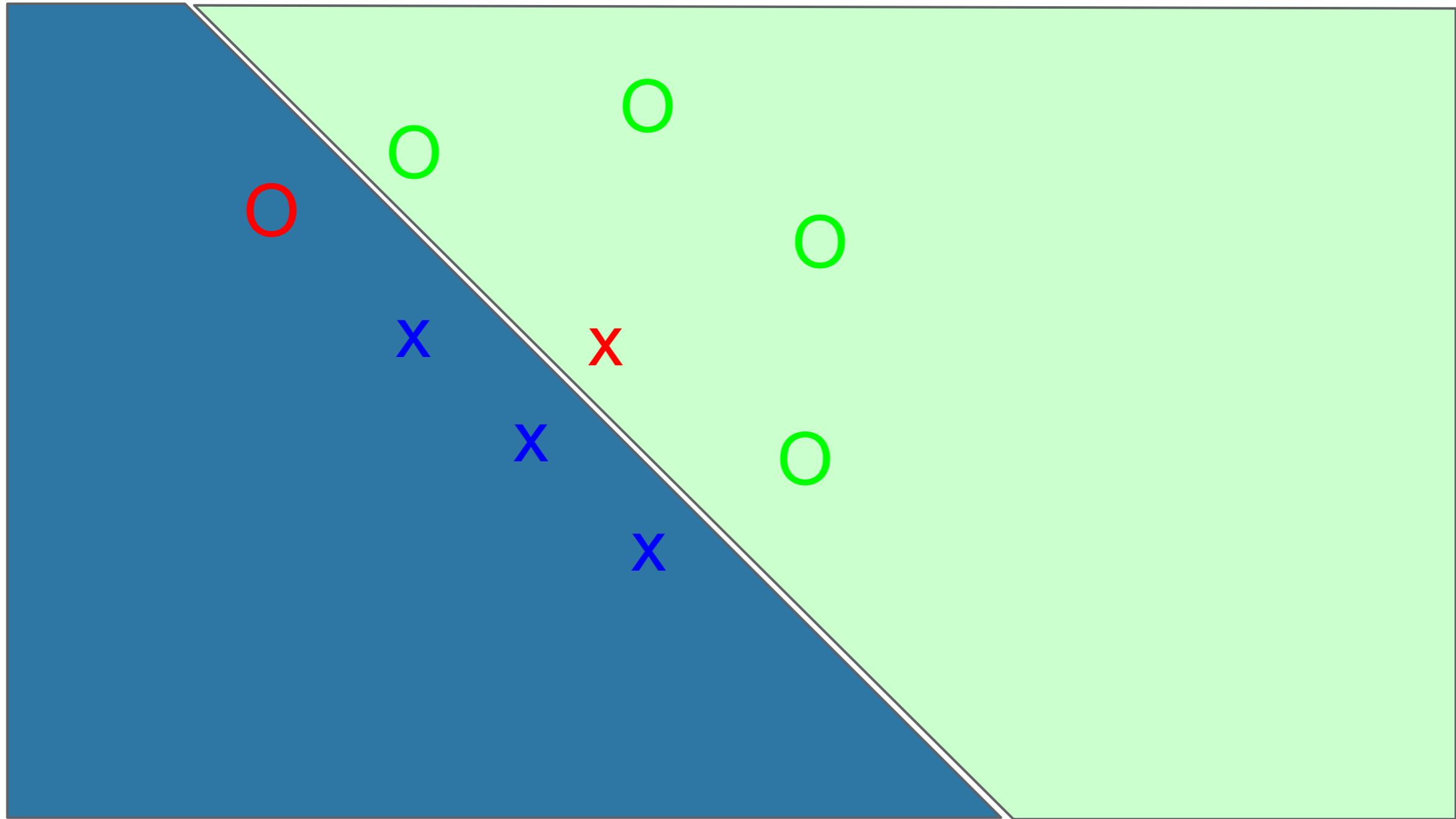
# Not just for neural nets

- Linear models
  - Logistic regression
  - Softmax regression
  - SVMs
- Decision trees
- Nearest neighbors

# Adversarial Examples from Overfitting

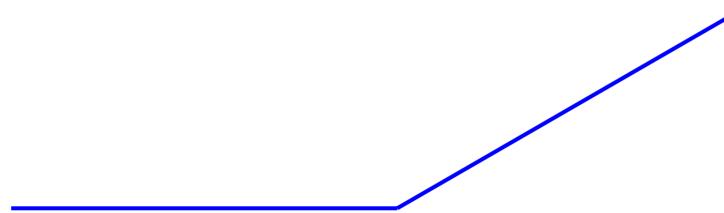


# Adversarial Examples from Excessive Linearity

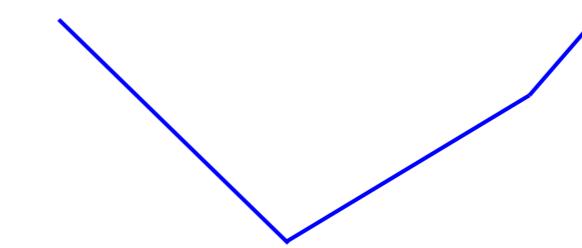


# Modern deep nets are very piecewise linear

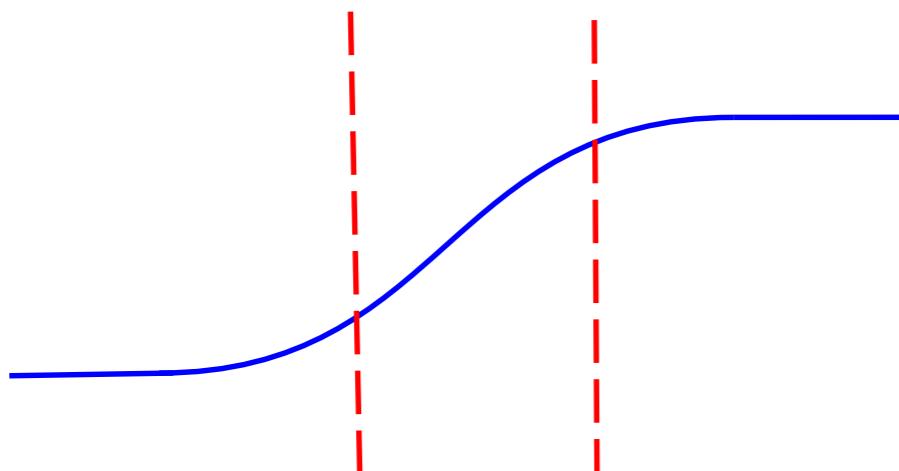
Rectified linear unit



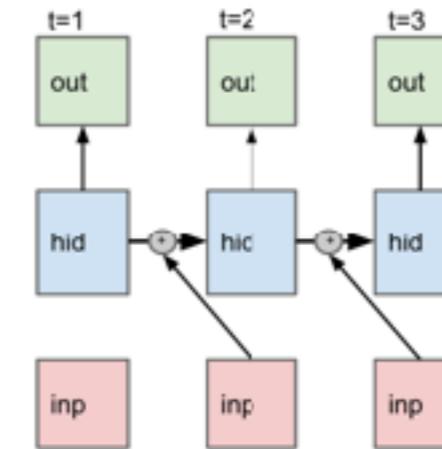
Maxout



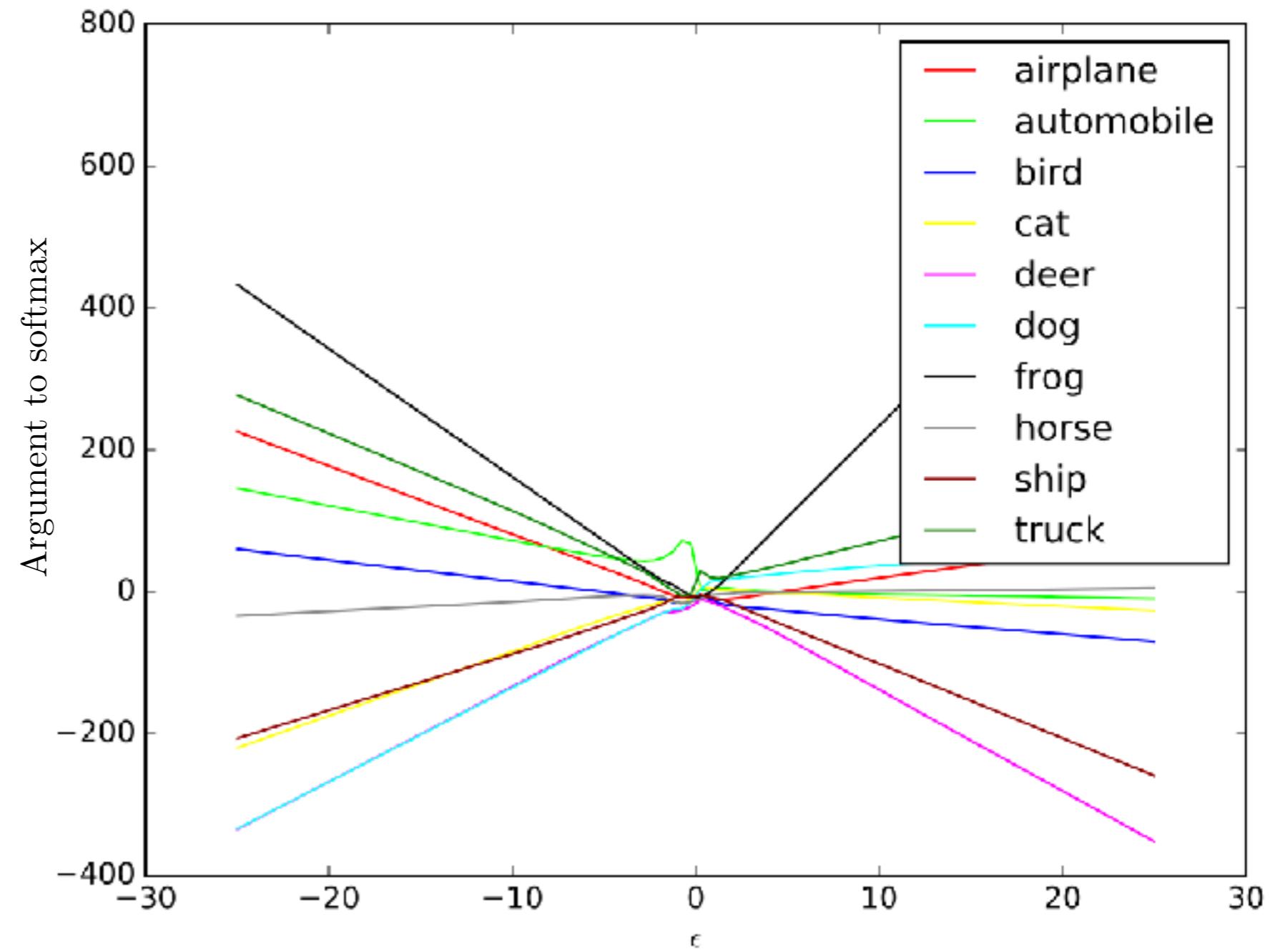
Carefully tuned sigmoid



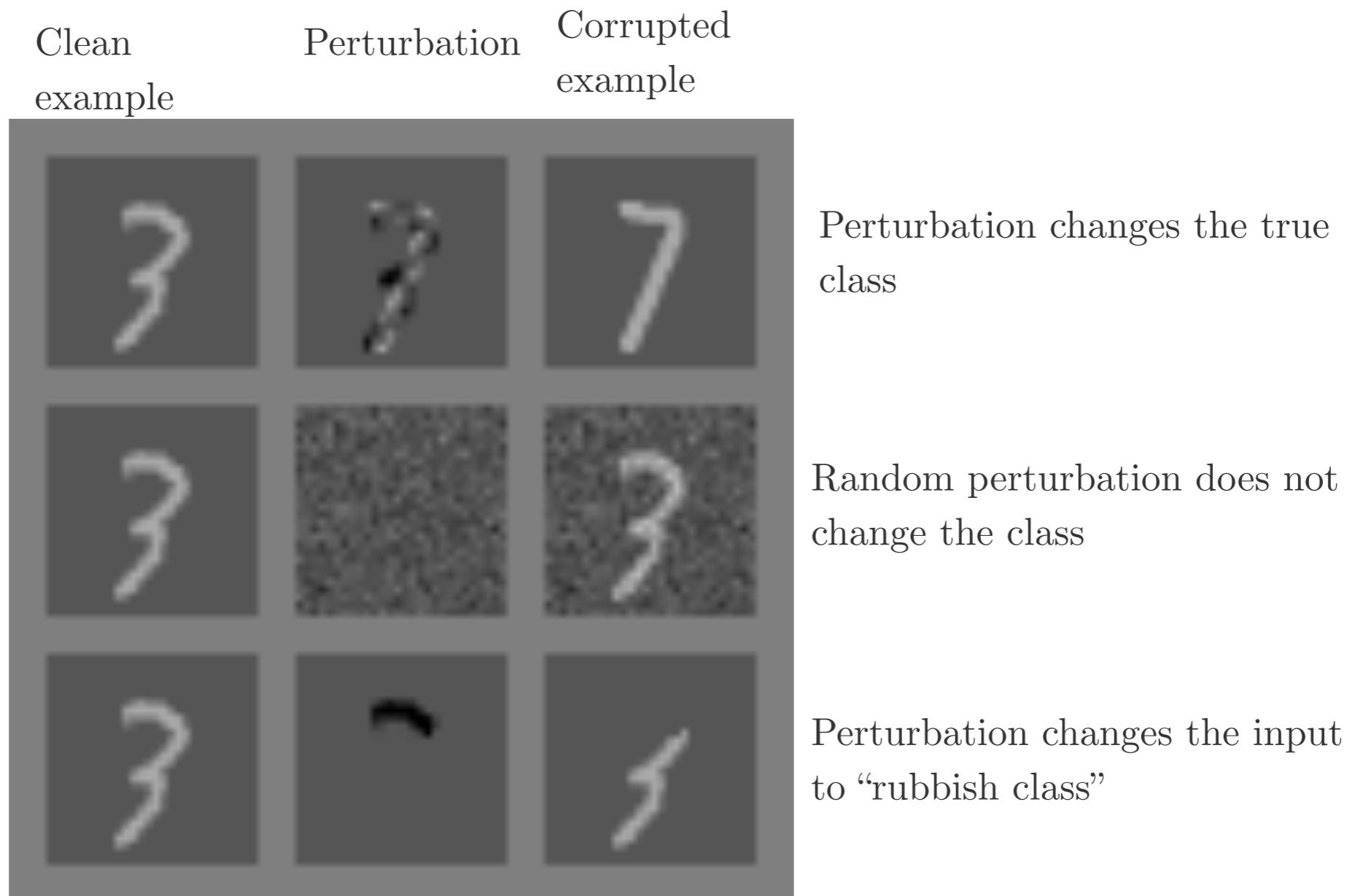
LSTM



# Nearly Linear Responses in Practice



# Small inter-class distances



All three perturbations have L2 norm 3.96

This is actually small. We typically use 7!

# The Fast Gradient Sign Method

$$J(\tilde{\boldsymbol{x}}, \theta) \approx J(\boldsymbol{x}, \theta) + (\tilde{\boldsymbol{x}} - \boldsymbol{x})^\top \nabla_{\boldsymbol{x}} J(\boldsymbol{x}).$$

Maximize

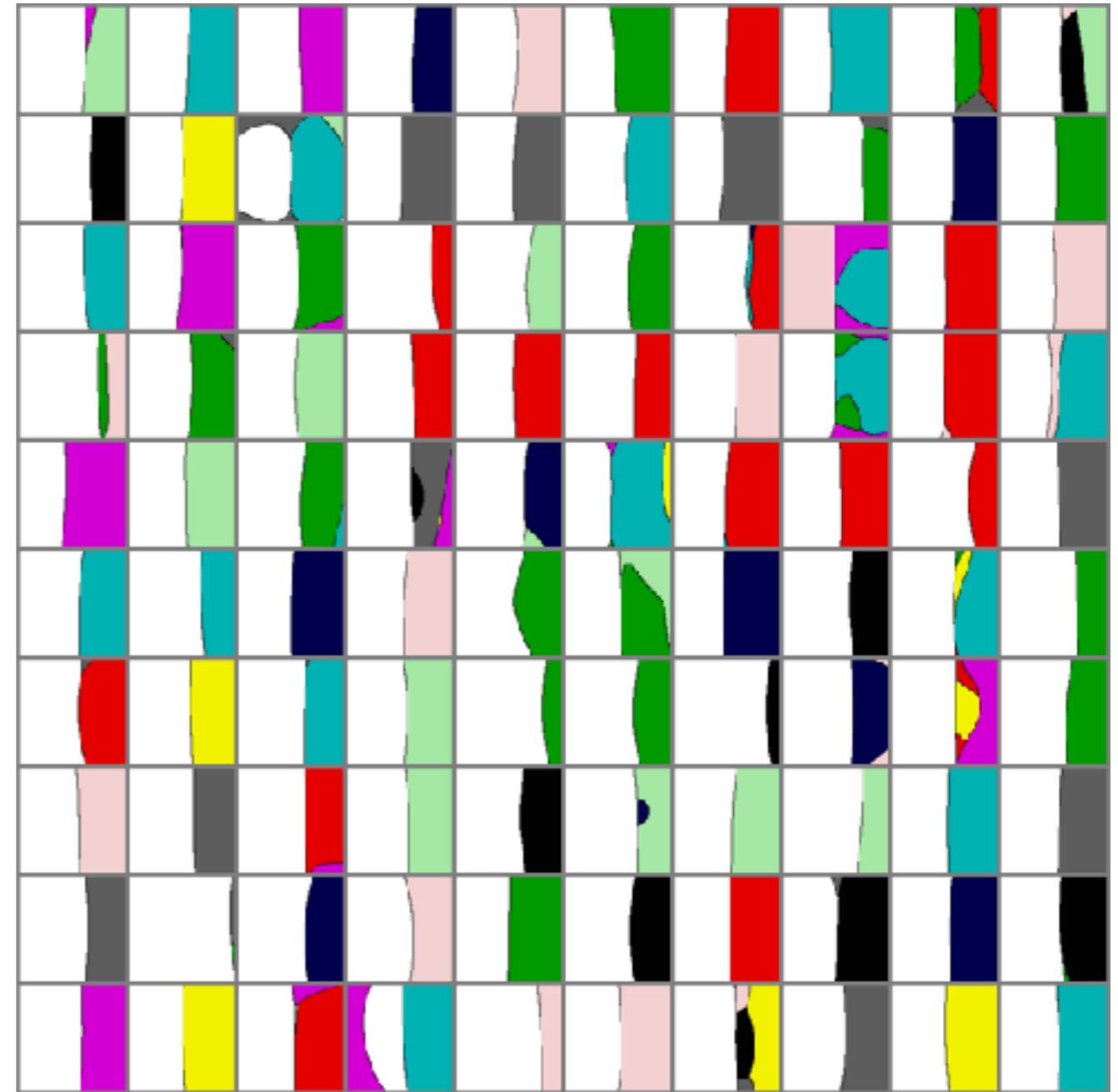
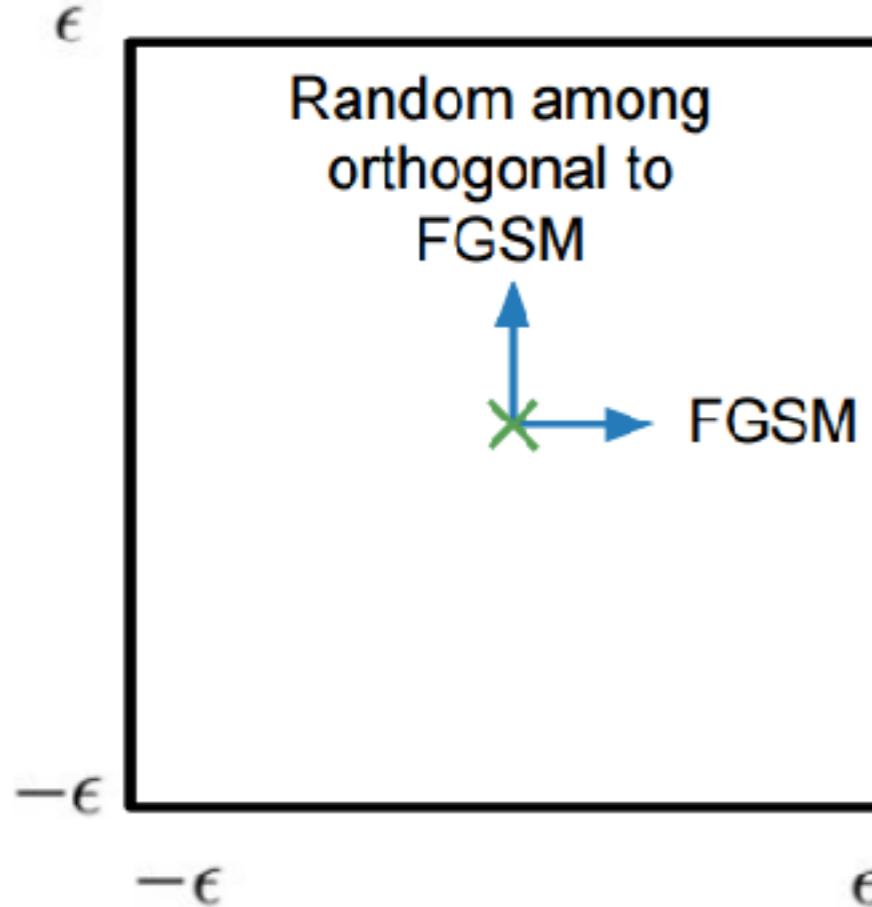
$$J(\boldsymbol{x}, \theta) + (\tilde{\boldsymbol{x}} - \boldsymbol{x})^\top \nabla_{\boldsymbol{x}} J(\boldsymbol{x})$$

subject to

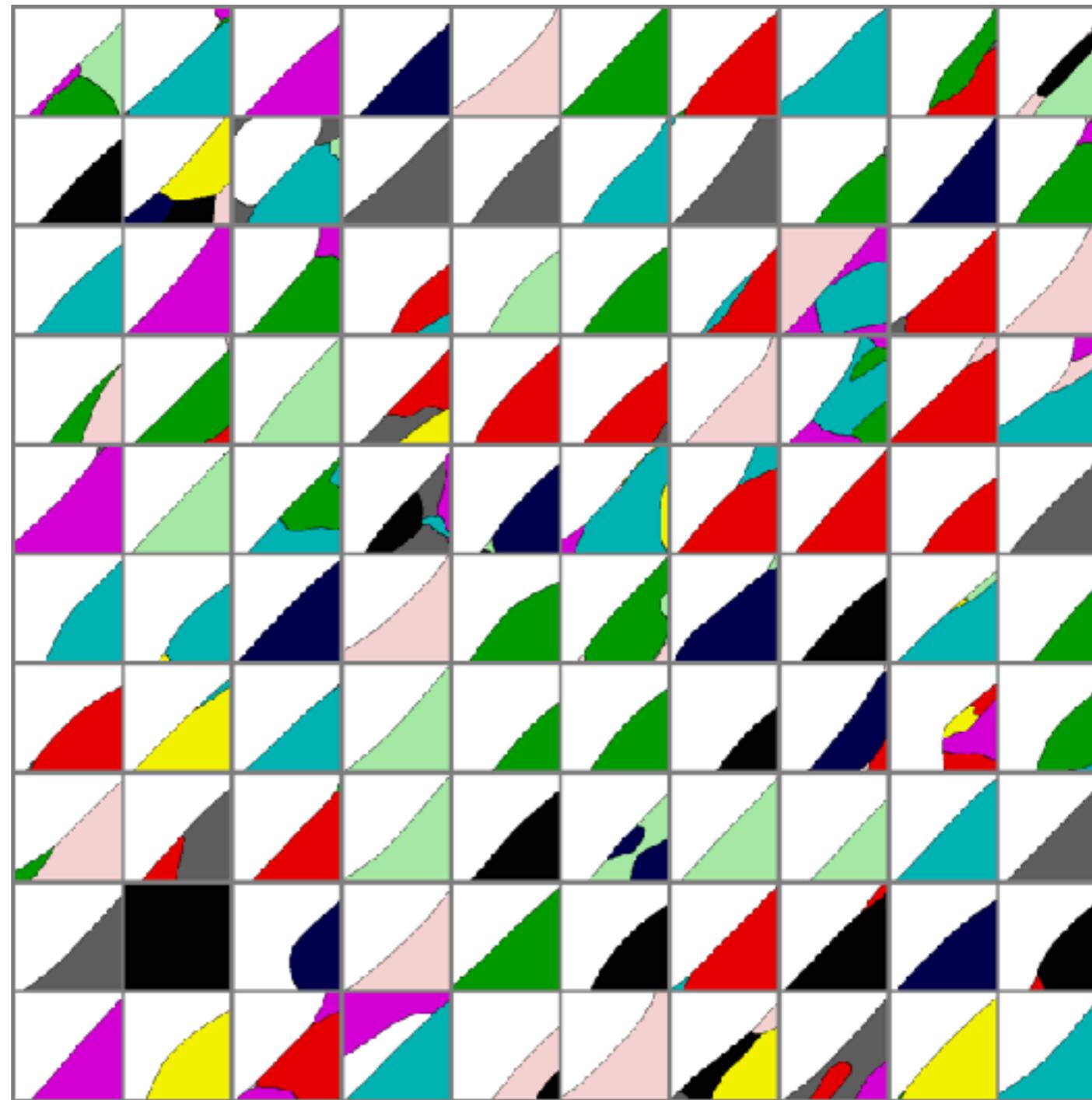
$$\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_\infty \leq \epsilon$$

$$\Rightarrow \tilde{\boldsymbol{x}} = \boldsymbol{x} + \epsilon \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{x})).$$

# Maps of Adversarial and Random Cross-Sections

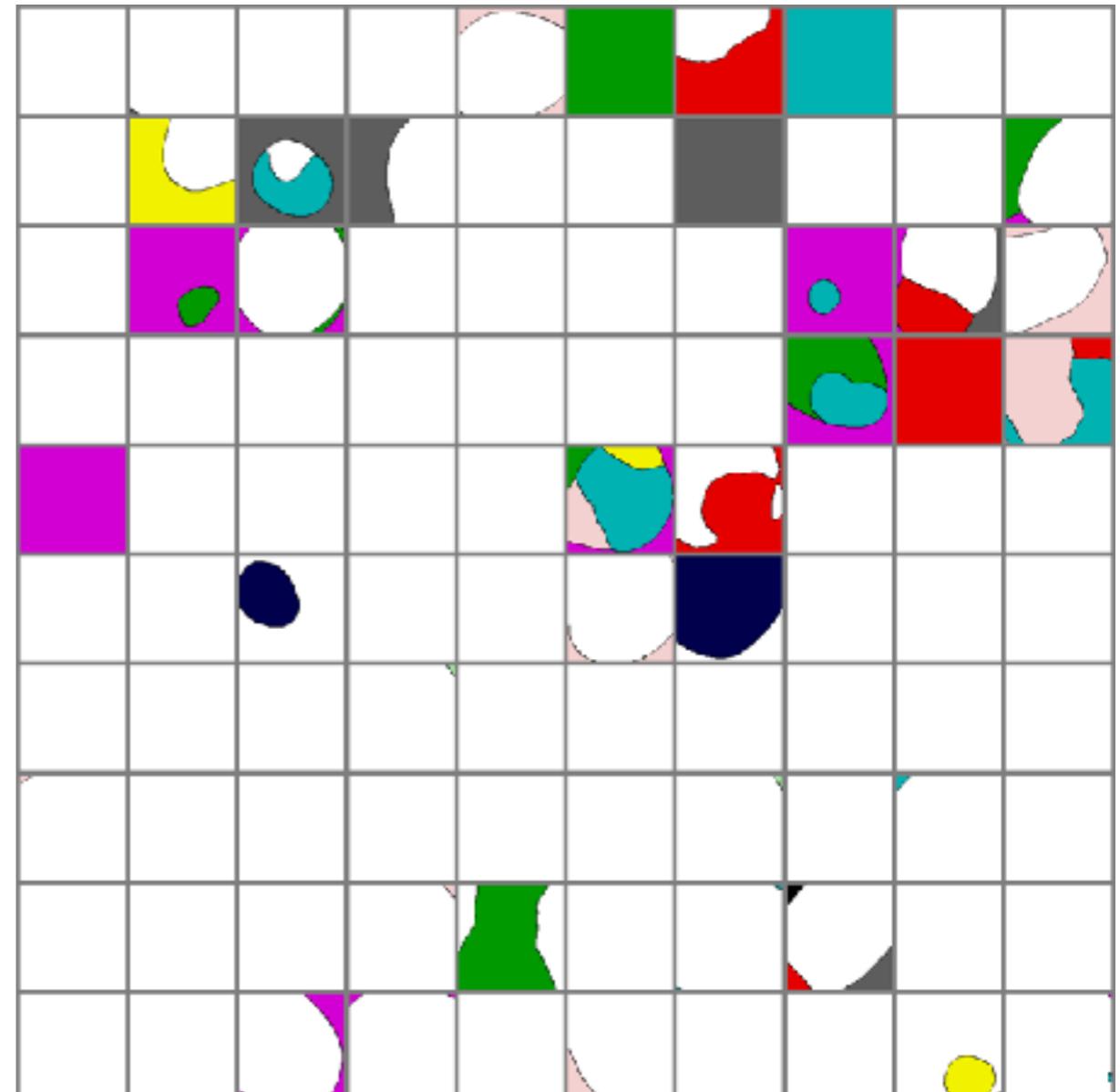
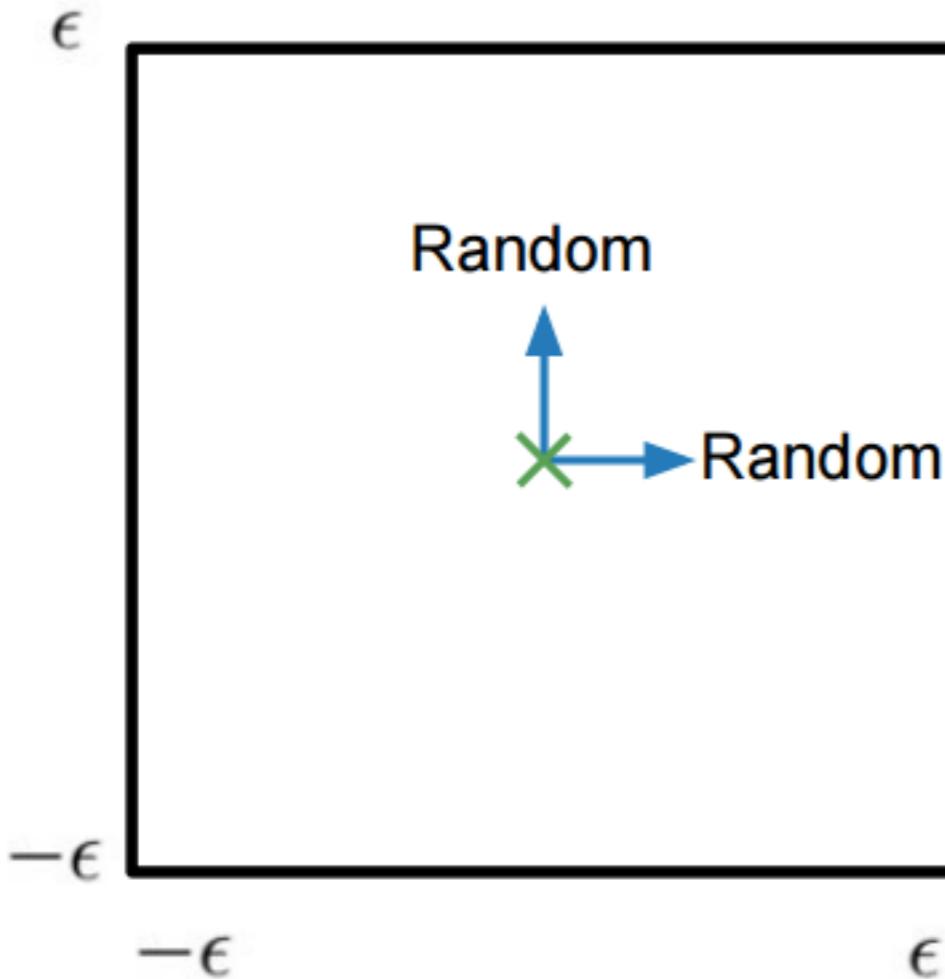


# Maps of Adversarial Cross-Sections

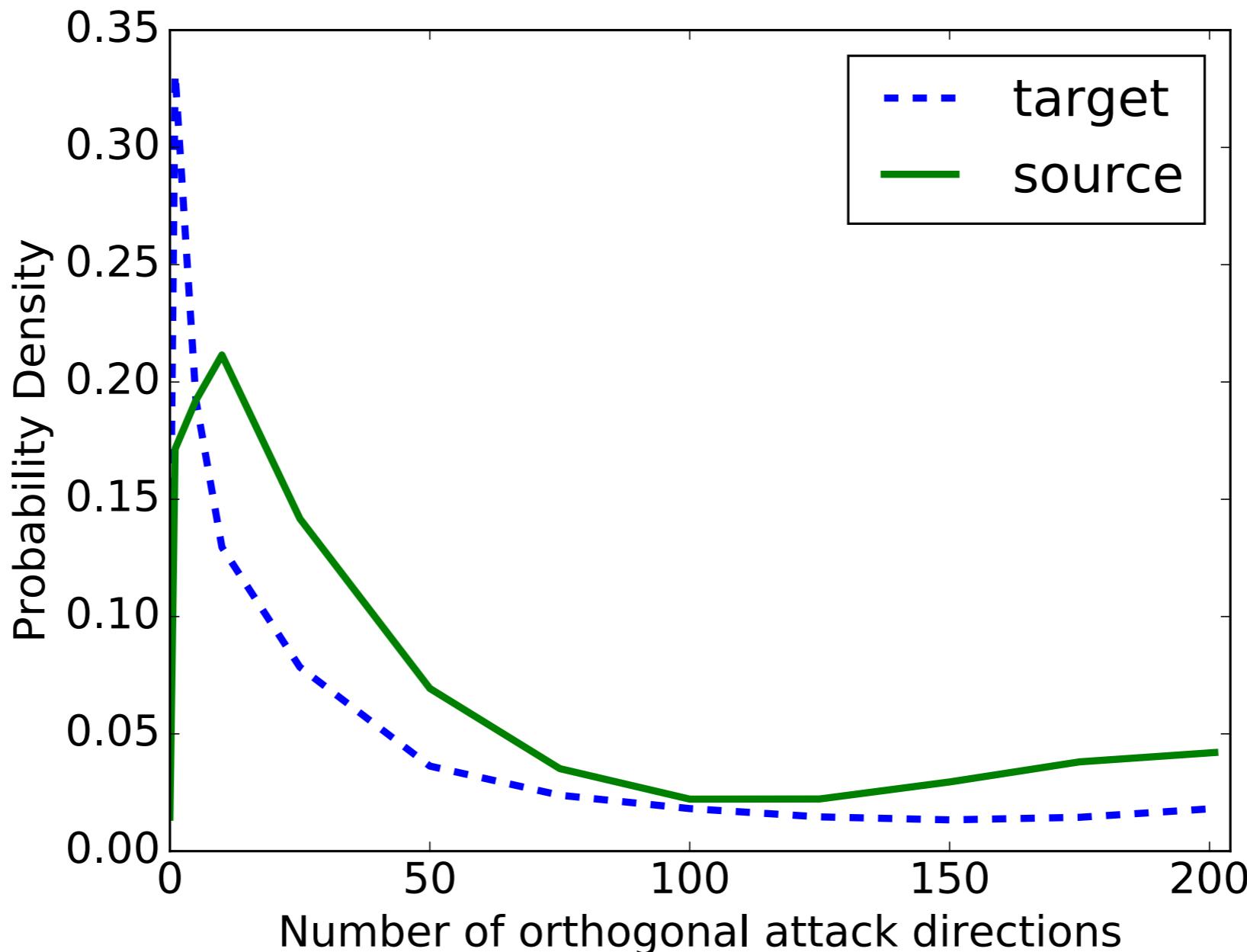


# Maps of Random Cross-Sections

Adversarial examples  
are not noise



# Estimating the Subspace Dimensionality



(Tramèr et al, 2017)

(Goodfellow 2016)

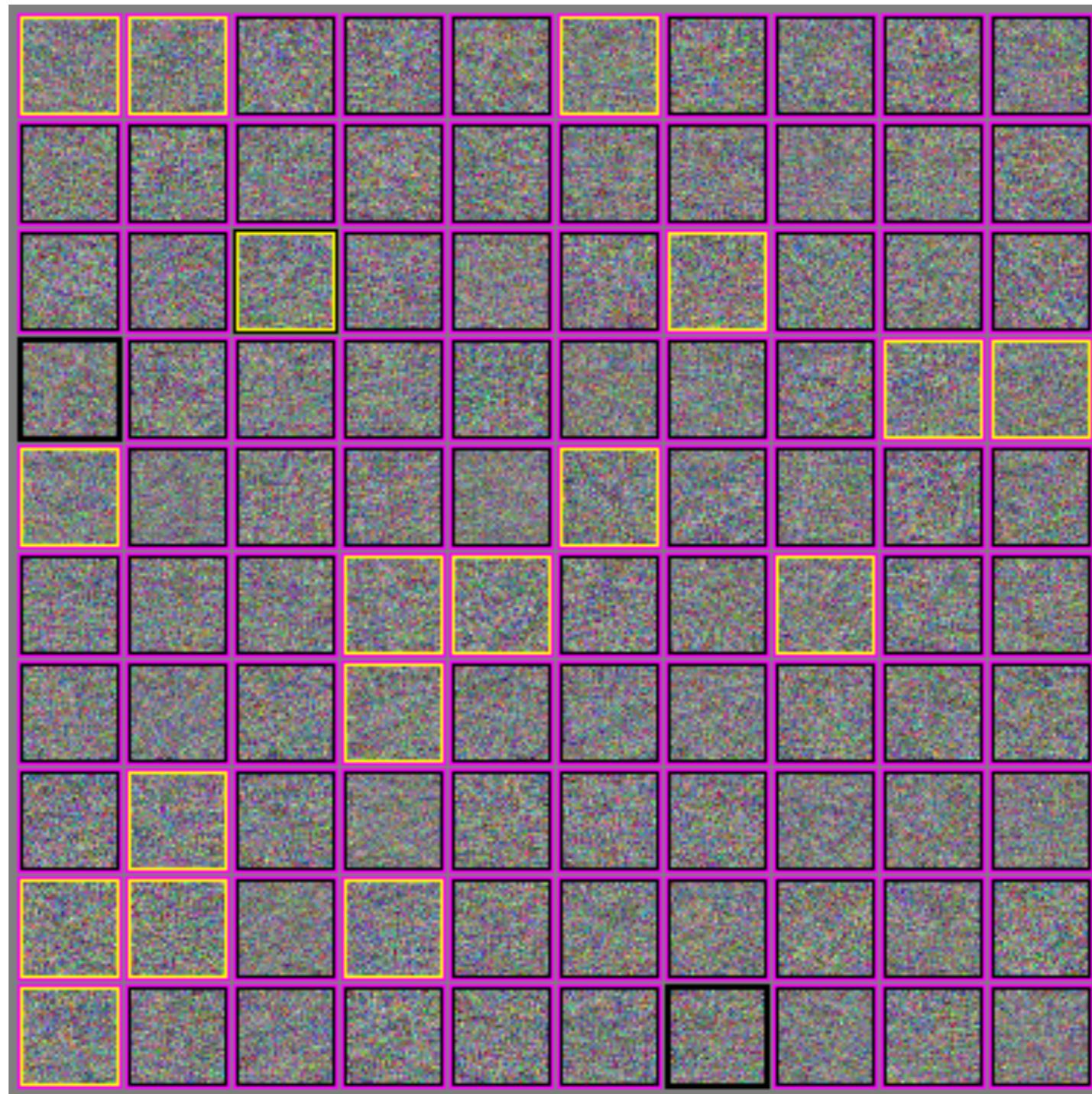
# Clever Hans



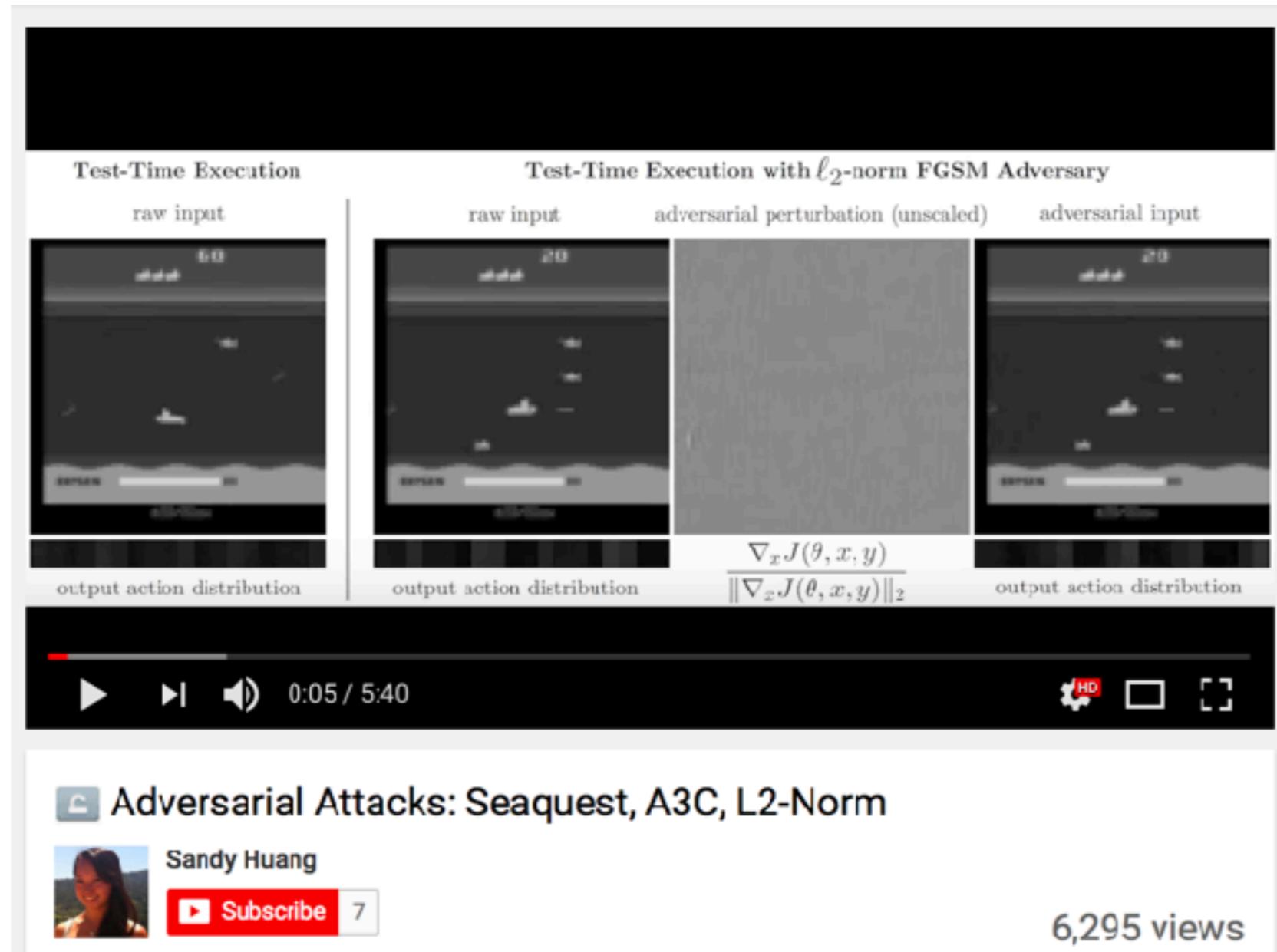
(“Clever Hans,  
Clever  
Algorithms,”  
Bob Sturm)



# Wrong almost everywhere

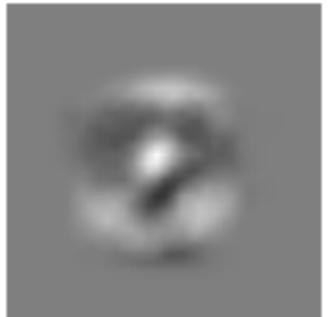
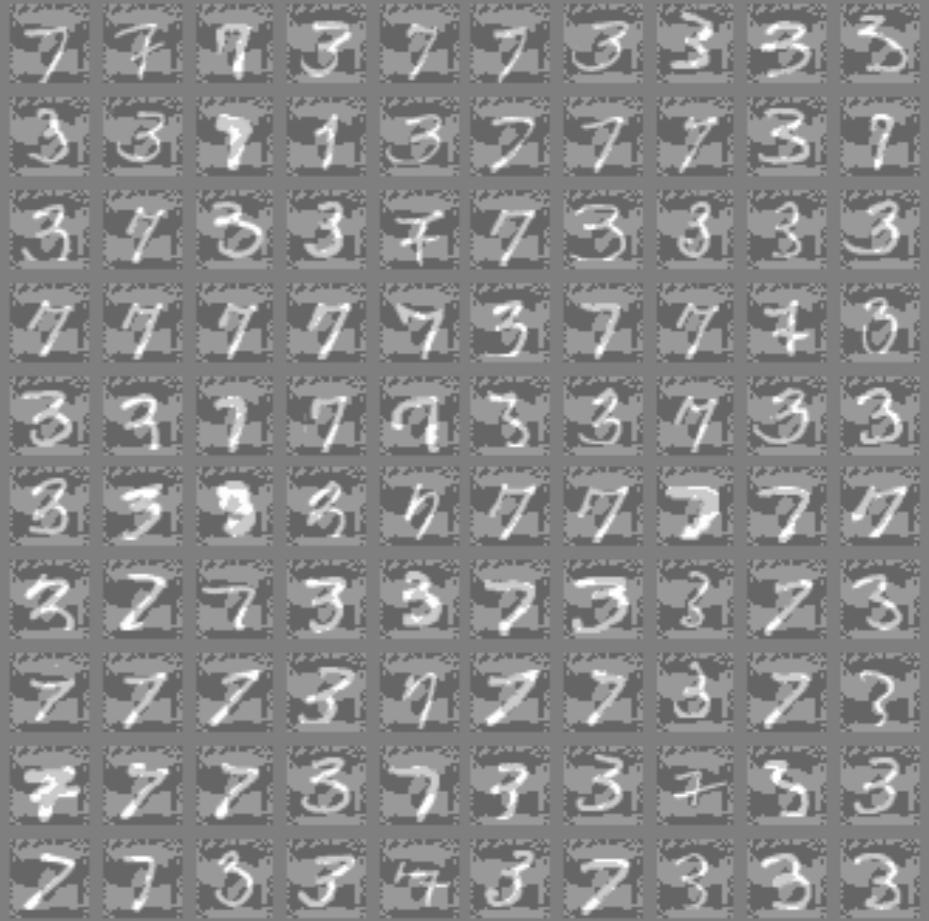
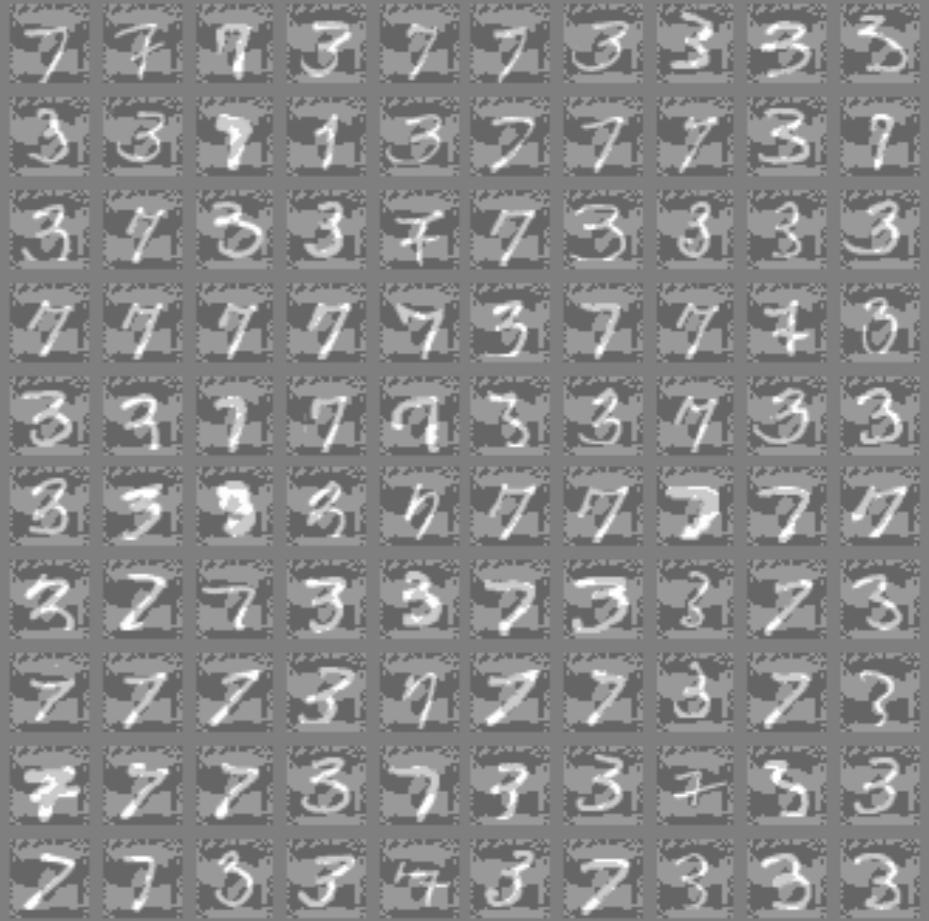


# Adversarial Examples for RL

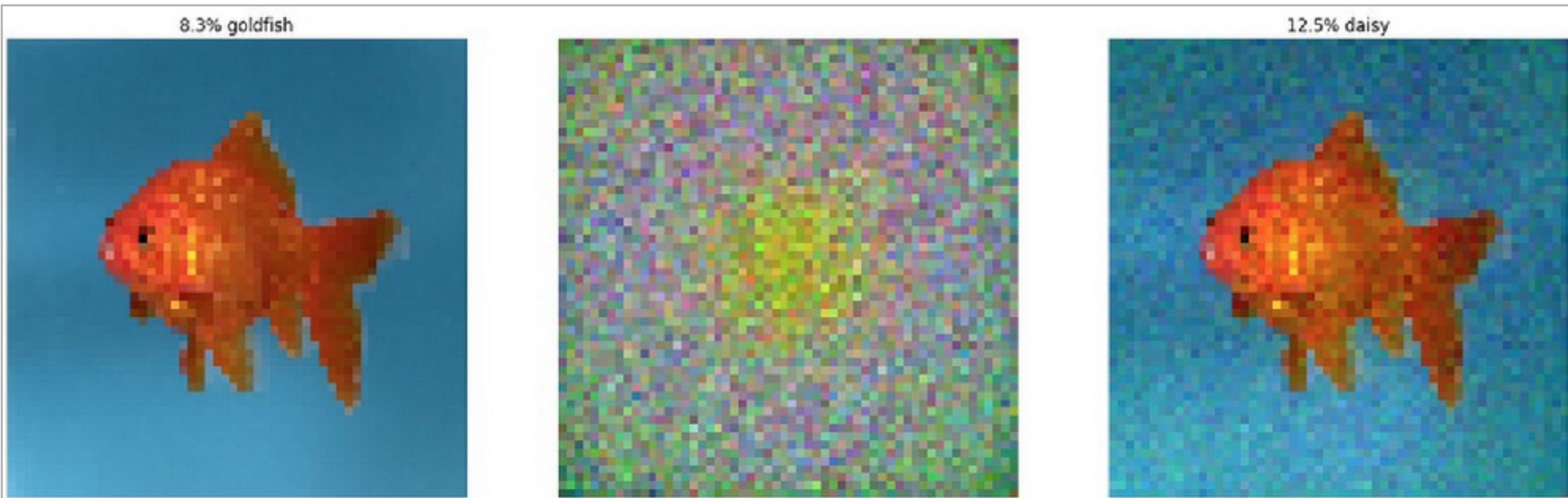


(Huang et al., 2017)

# High-Dimensional Linear Models

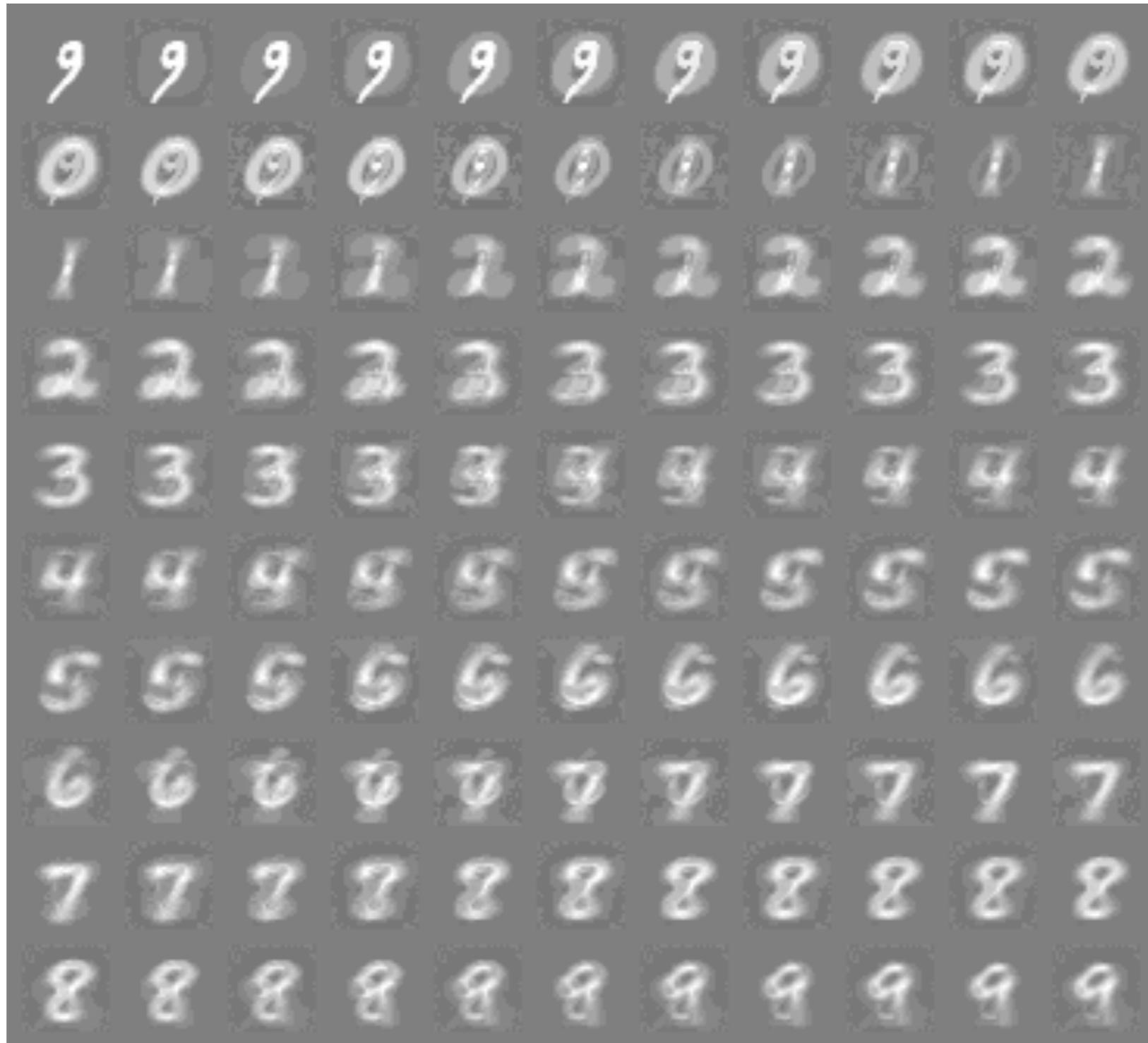
|                  | Clean examples  | Adversarial  |
|------------------|---|--|
| Weights          |   |  |
| Signs of weights |  |  |

# Linear Models of ImageNet

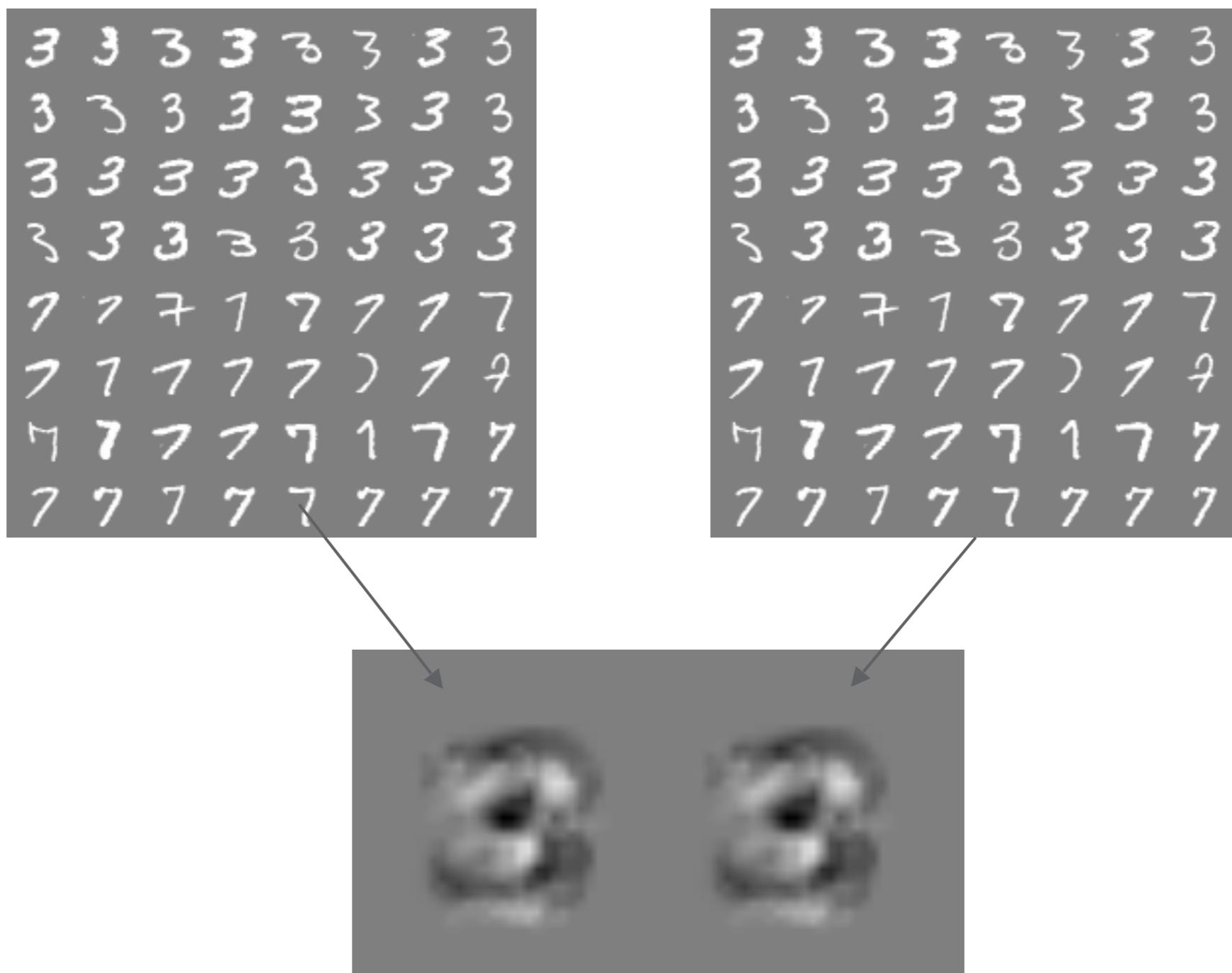


(Andrej Karpathy, “Breaking Linear Classifiers on ImageNet”)

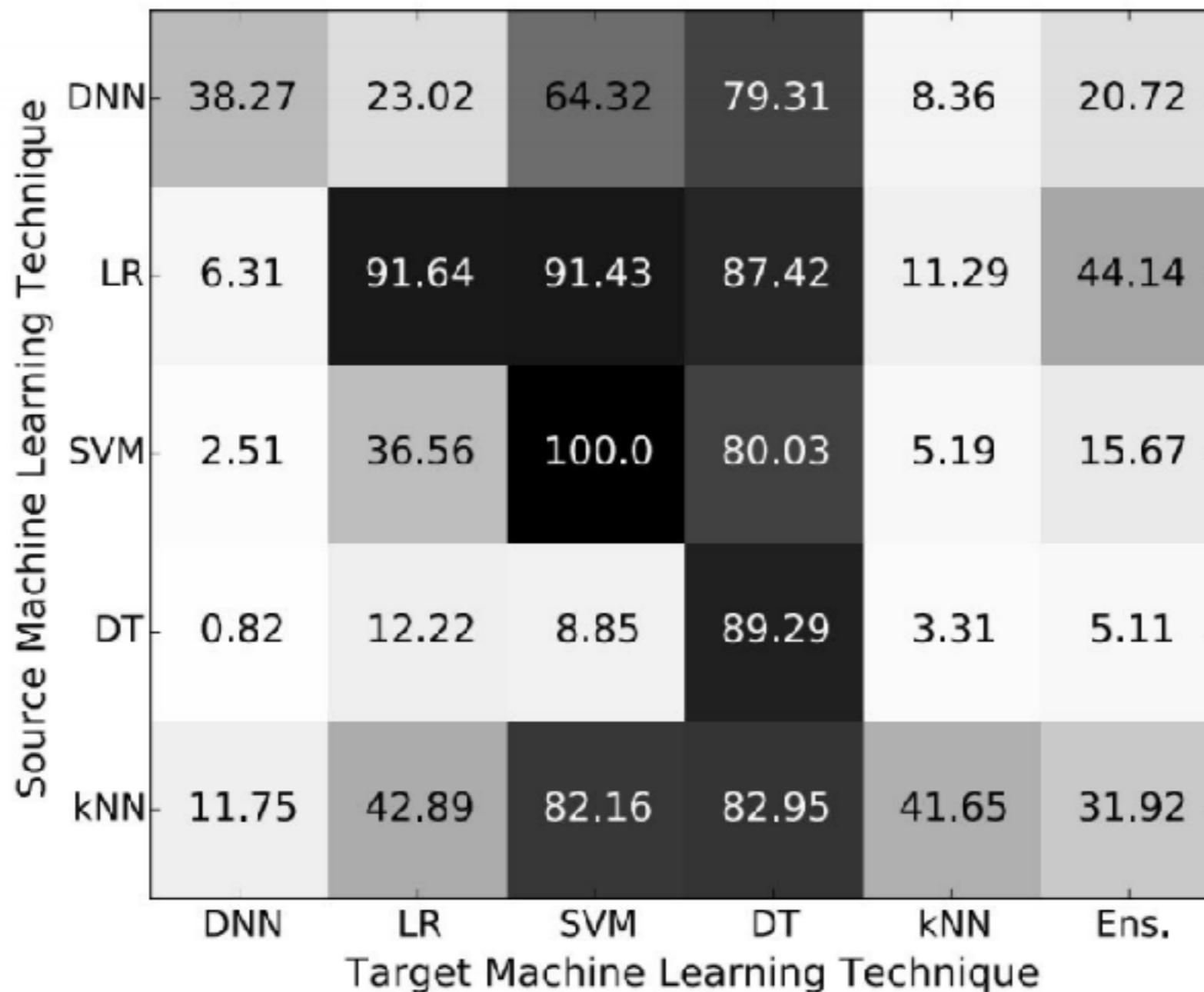
# RBFs behave more intuitively



# Cross-model, cross-dataset generalization



# Cross-technique transferability



(Papernot 2016)

# Transferability Attack

Target model with unknown weights, machine learning algorithm, training set; maybe non-differentiable

*Train your own model*

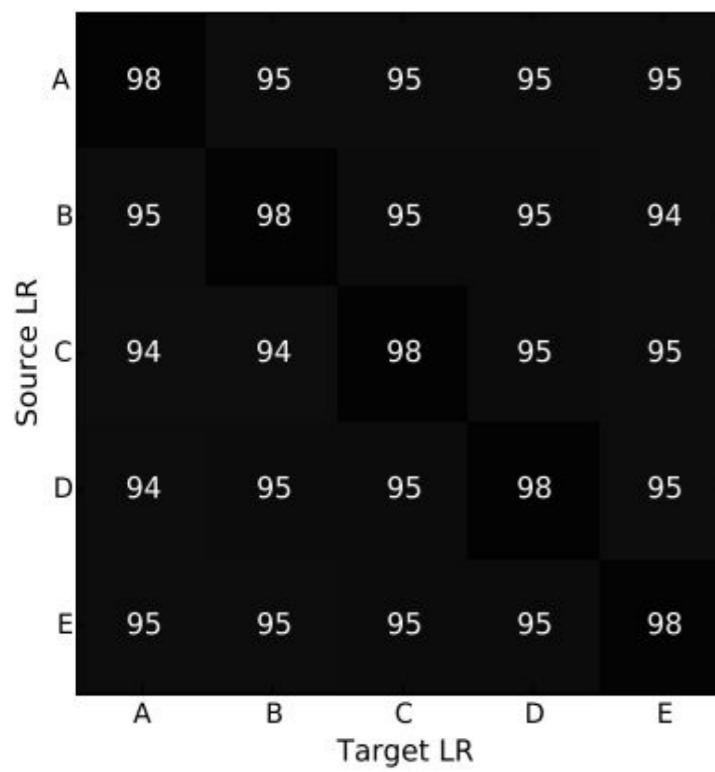
Substitute model mimicking target model with known, differentiable function

*Deploy adversarial examples against the target; transferability property results in them succeeding*

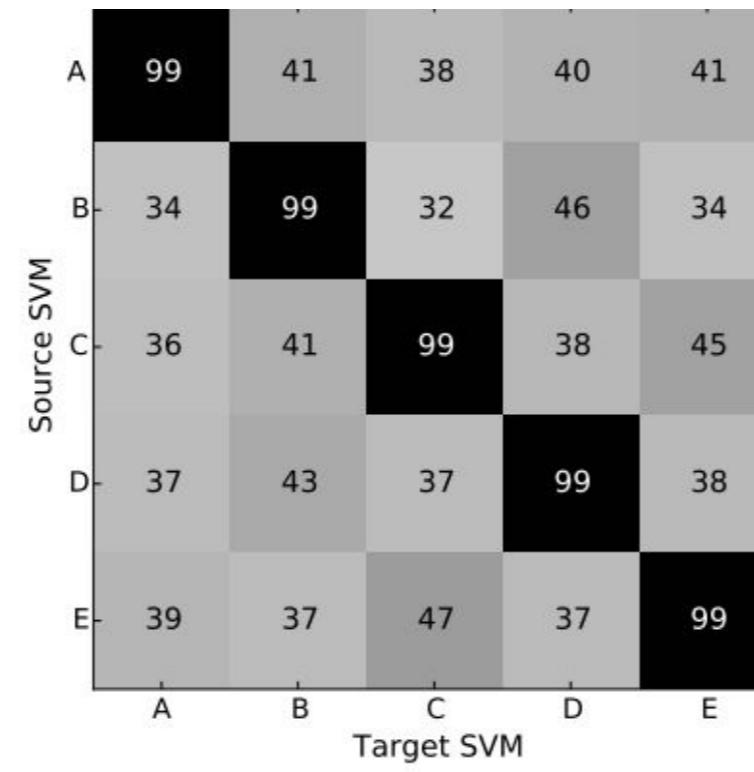
Adversarial examples

*Adversarial crafting against substitute*

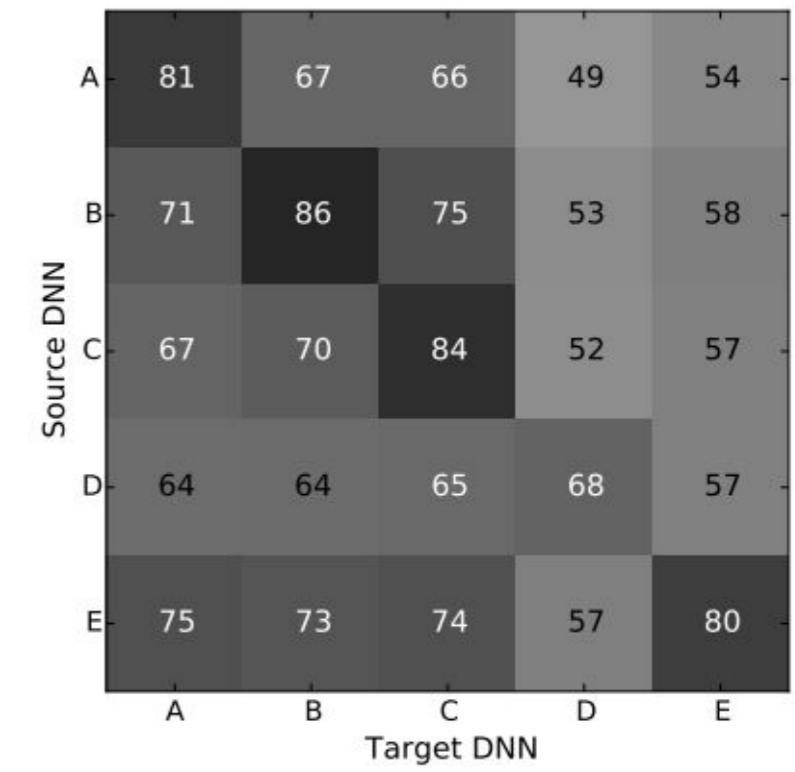
# Cross-Training Data Transferability



Strong



Weak



Intermediate

(Papernot 2016)

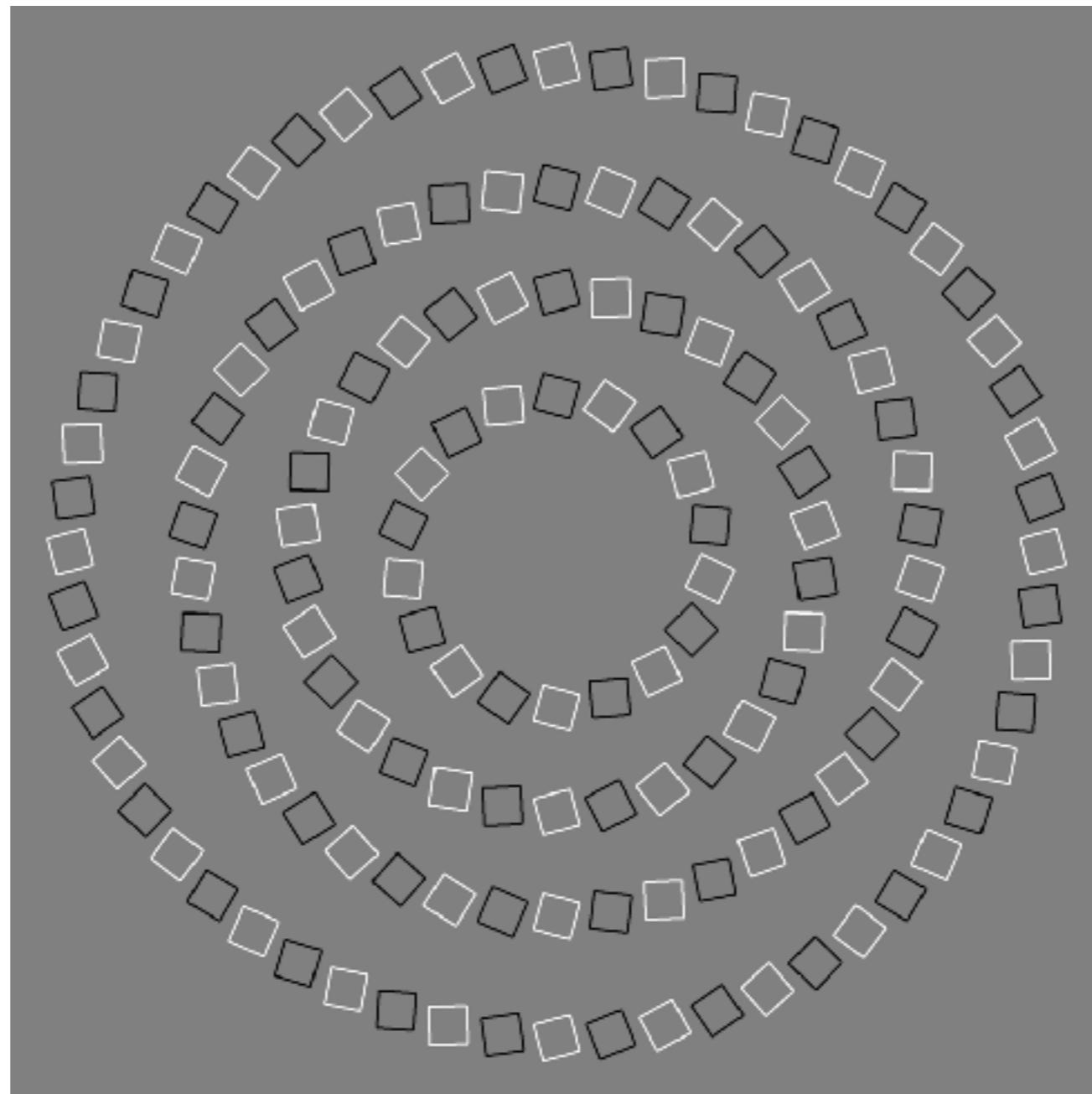
# Enhancing Transfer With Ensembles

|             | RMSD  | ResNet-152 | ResNet-101 | ResNet-50 | VGG-16 | GoogLeNet |
|-------------|-------|------------|------------|-----------|--------|-----------|
| -ResNet-152 | 17.17 | 0%         | 0%         | 0%        | 0%     | 0%        |
| -ResNet-101 | 17.25 | 0%         | 1%         | 0%        | 0%     | 0%        |
| -ResNet-50  | 17.25 | 0%         | 0%         | 2%        | 0%     | 0%        |
| -VGG-16     | 17.80 | 0%         | 0%         | 0%        | 6%     | 0%        |
| -GoogLeNet  | 17.41 | 0%         | 0%         | 0%        | 0%     | 5%        |

Table 4: Accuracy of non-targeted adversarial images generated using the optimization-based approach. The first column indicates the average RMSD of the generated adversarial images. Cell  $(i, j)$  corresponds to the accuracy of the attack generated using four models except model  $i$  (row) when evaluated over model  $j$  (column). In each row, the minus sign “-” indicates that the model of the row is not used when generating the attacks. Results of top-5 accuracy can be found in the appendix (Table 14).

(Liu et al, 2016)

# Adversarial Examples in the Human Brain



These are  
concentric  
circles,  
not  
intertwined  
spirals.

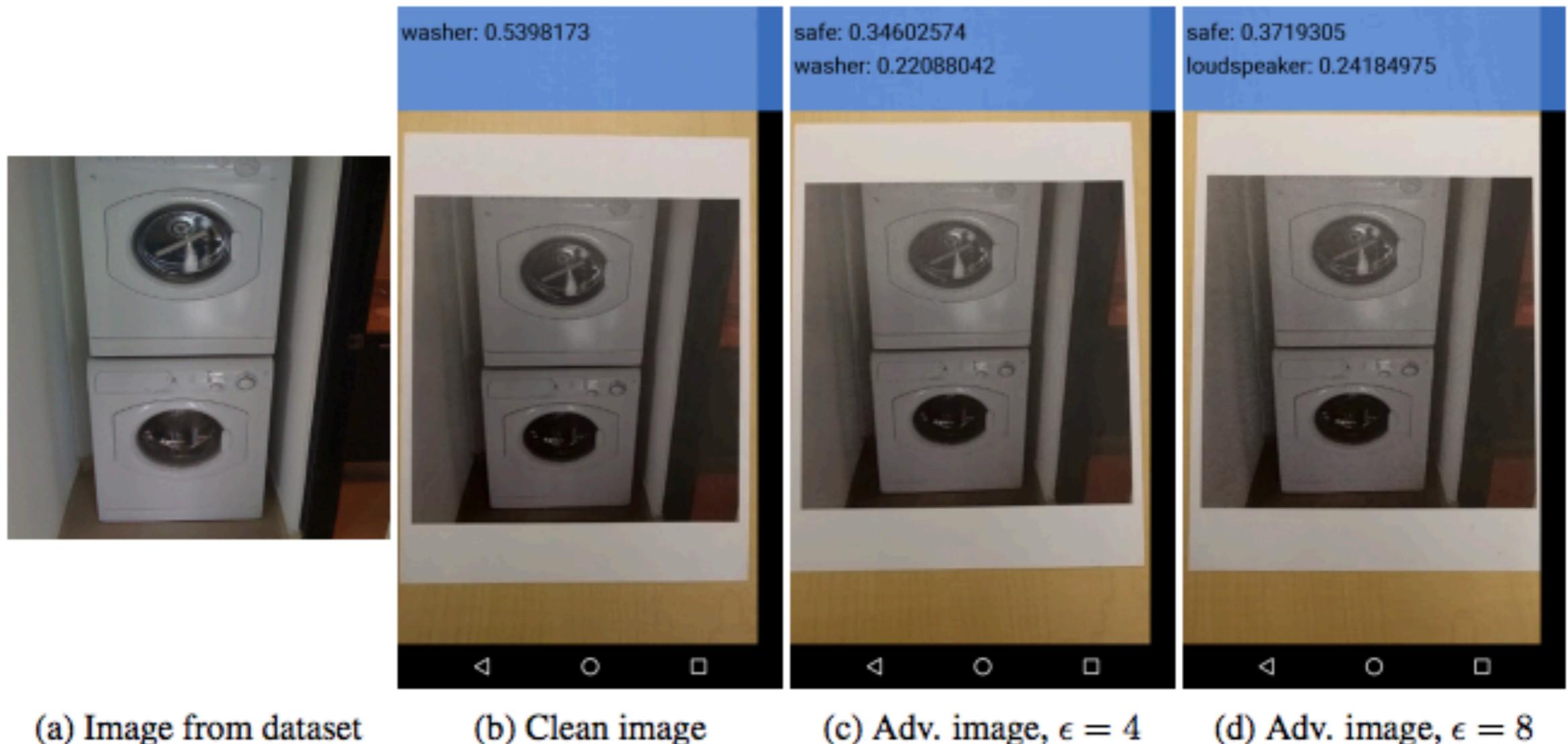
(Pinna and Gregory, 2002)

(Goodfellow 2016)

# Practical Attacks

- Fool real classifiers trained by remotely hosted API  
(MetaMind, Amazon, Google)
- Fool malware detector networks
- Display adversarial examples in the physical world  
and fool machine learning systems that perceive  
them through a camera

# Adversarial Examples in the Physical World



(a) Image from dataset

(b) Clean image

(c) Adv. image,  $\epsilon = 4$

(d) Adv. image,  $\epsilon = 8$

(Kurakin et al, 2016)

(Goodfellow 2016)

# Failed defenses

Generative  
pretraining

Adding noise  
at test time

Confidence-reducing  
perturbation at test time

Weight decay

Various  
non-linear units

Double backprop  
Dropout

Removing perturbation  
with an autoencoder

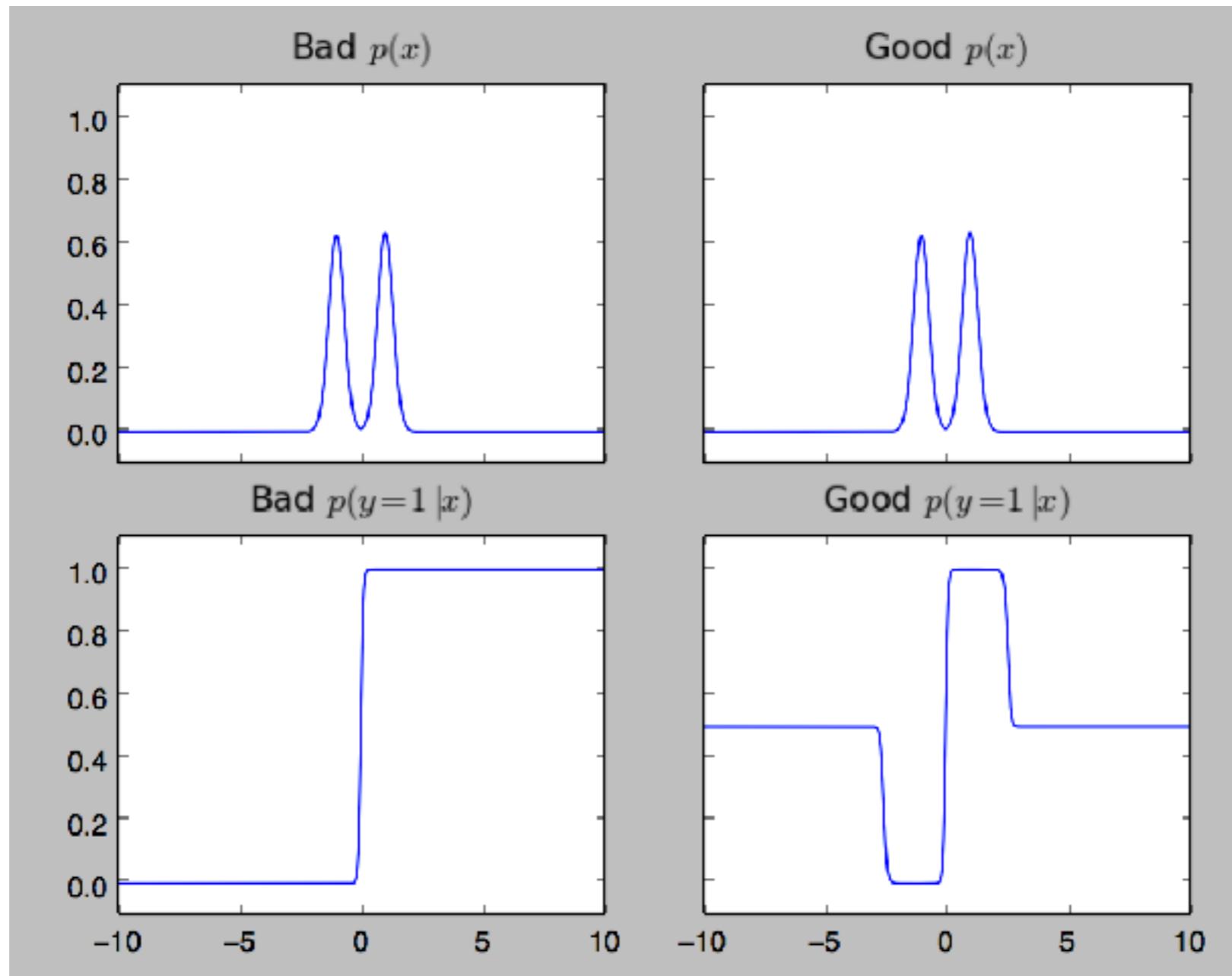
Ensembles

Error correcting  
codes

Multiple glimpses

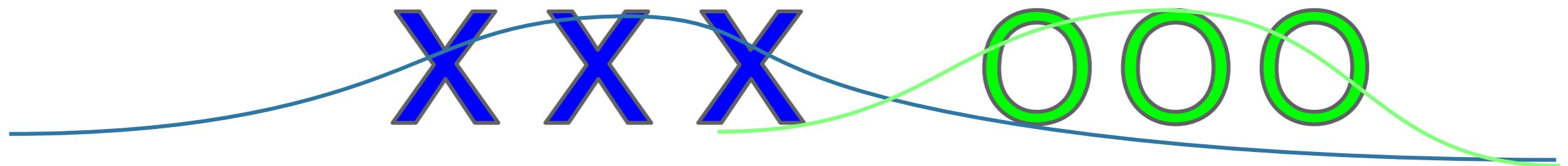
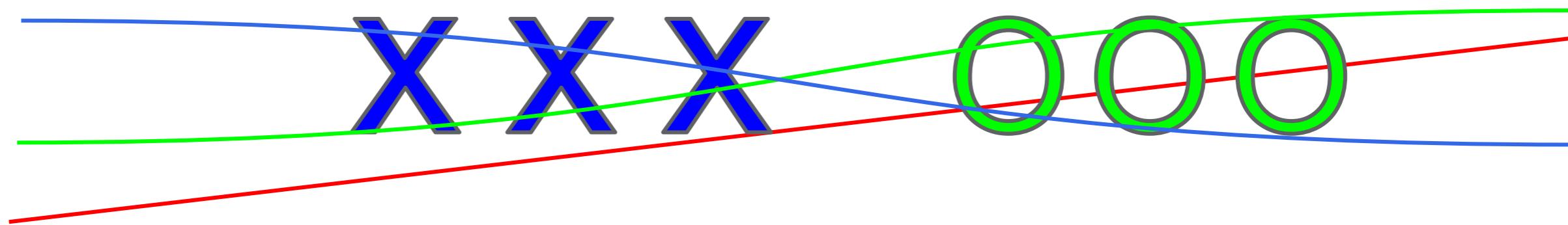
Adding noise  
at train time

# Generative Modeling is not Sufficient to Solve the Problem



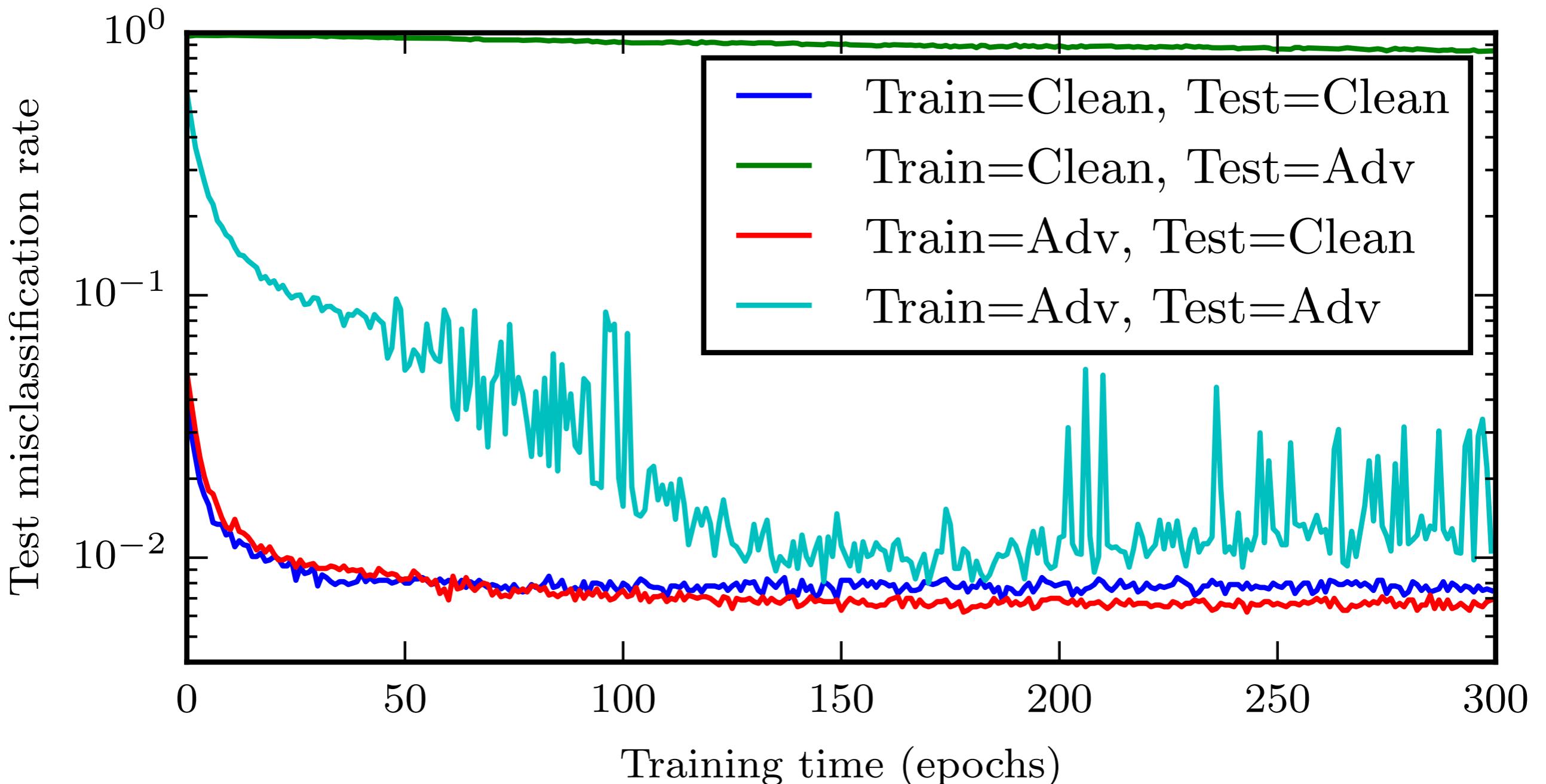
# Universal approximator theorem

Neural nets can represent either function:



Maximum likelihood doesn't cause them to learn the right function. But we can fix that...

# Training on Adversarial Examples



# Adversarial Training of other Models

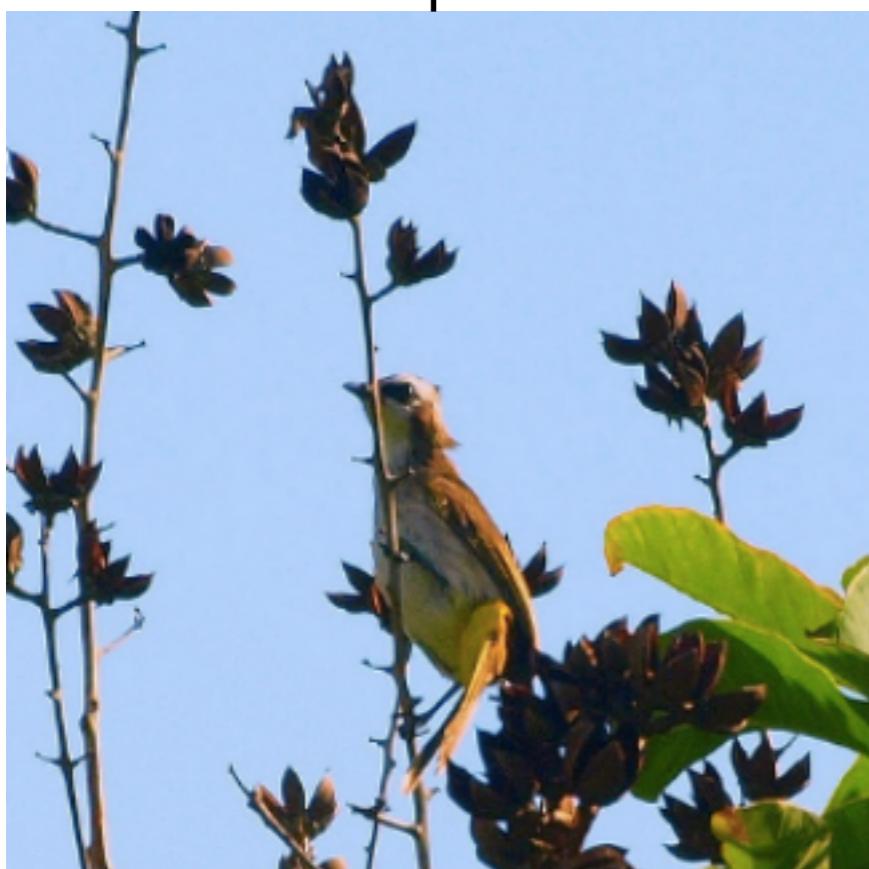
- Linear models: SVM / linear regression cannot learn a step function, so adversarial training is less useful, very similar to weight decay
- $k$ -NN: adversarial training is prone to overfitting.
- Takeaway: neural nets can actually become more secure than other models. *Adversarially trained neural nets have the best empirical success rate on adversarial examples of any machine learning model.*

# Weaknesses Persist



# Adversarial Training

Labeled as bird



Still has same label (bird)

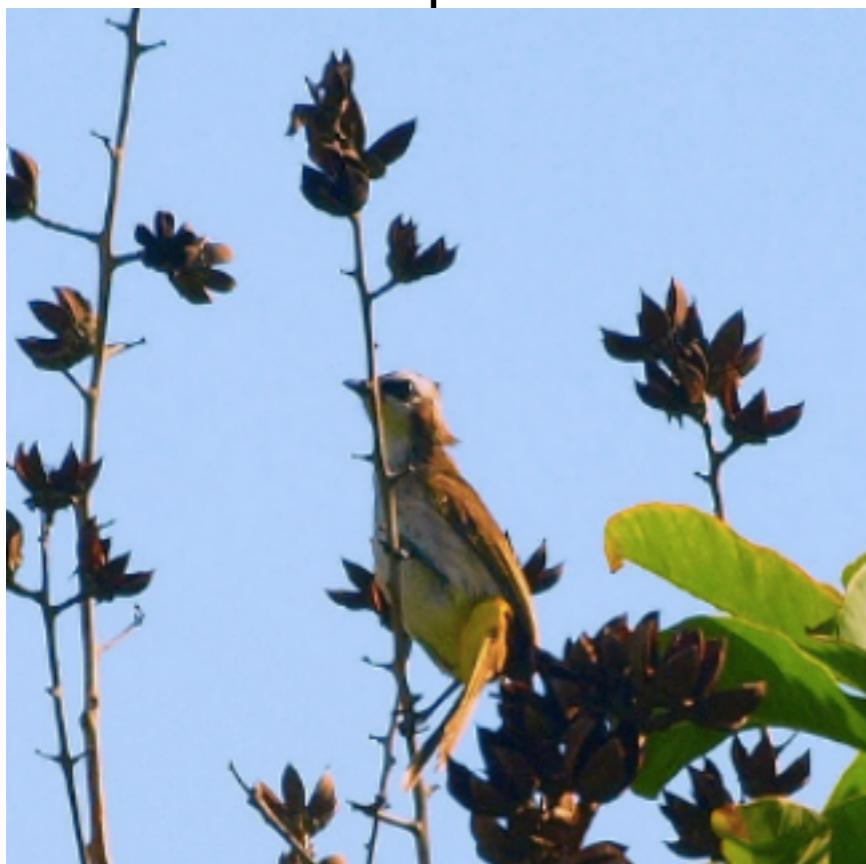


---

Decrease  
probability  
of bird class

# Virtual Adversarial Training

Unlabeled; model  
guesses it's probably  
a bird, maybe a plane



New guess should  
match old guess

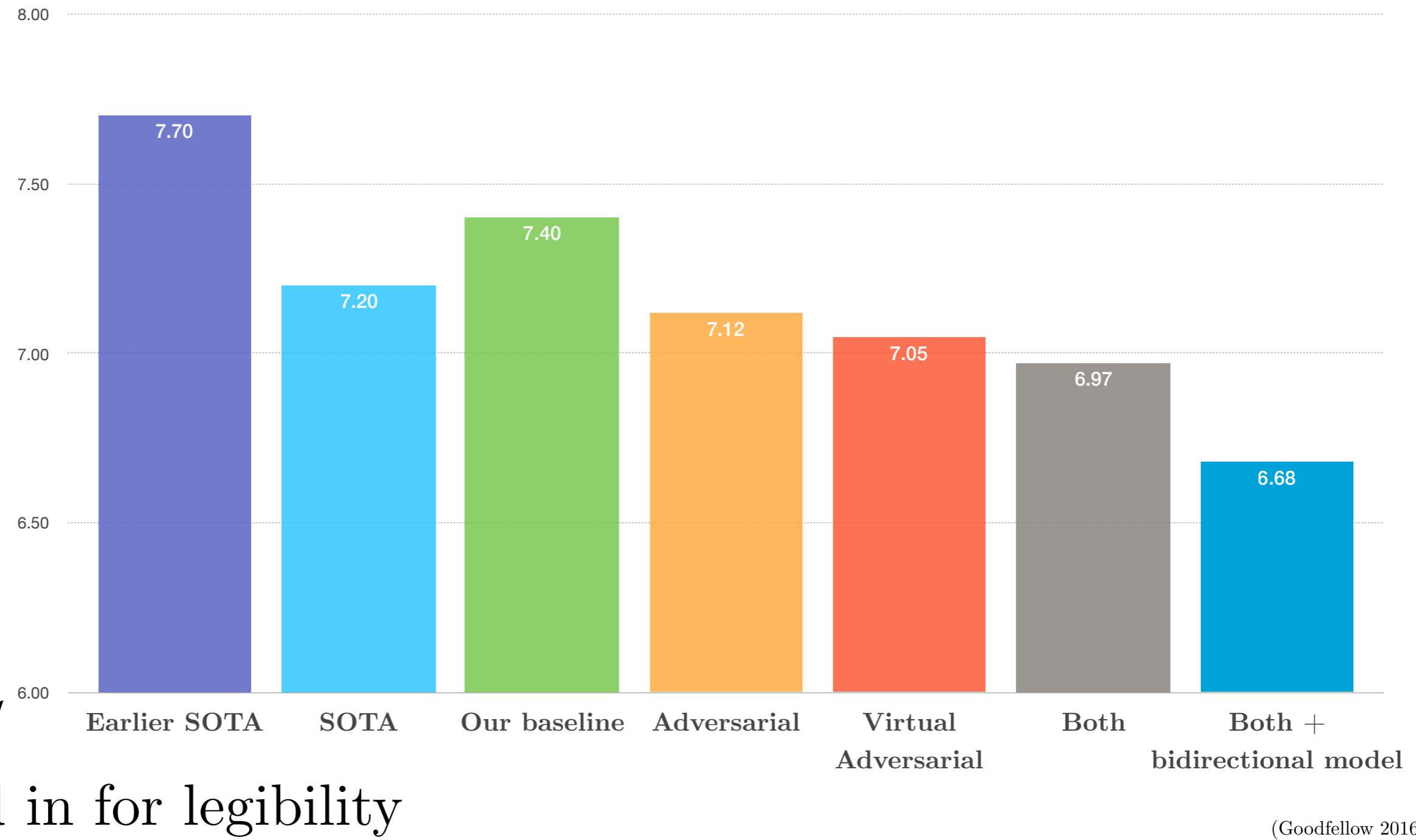
(probably bird, maybe plane)



→  
Adversarial  
perturbation  
intended to  
change the guess

# Text Classification with VAT

## RCV1 Misclassification Rate



# Universal engineering machine (model-based optimization)

Make new inventions  
by finding input  
that maximizes  
model's predicted  
performance

Training data

Extrapolation



# Conclusion

- Attacking is easy
- Defending is difficult
- Adversarial training provides regularization and semi-supervised learning
- The out-of-domain input problem is a bottleneck for model-based optimization generally

# cleverhans

Open-source library available at:

<https://github.com/openai/cleverhans>

Built on top of TensorFlow (Theano support anticipated)

Standard implementation of attacks, for adversarial training  
and reproducible benchmarks

