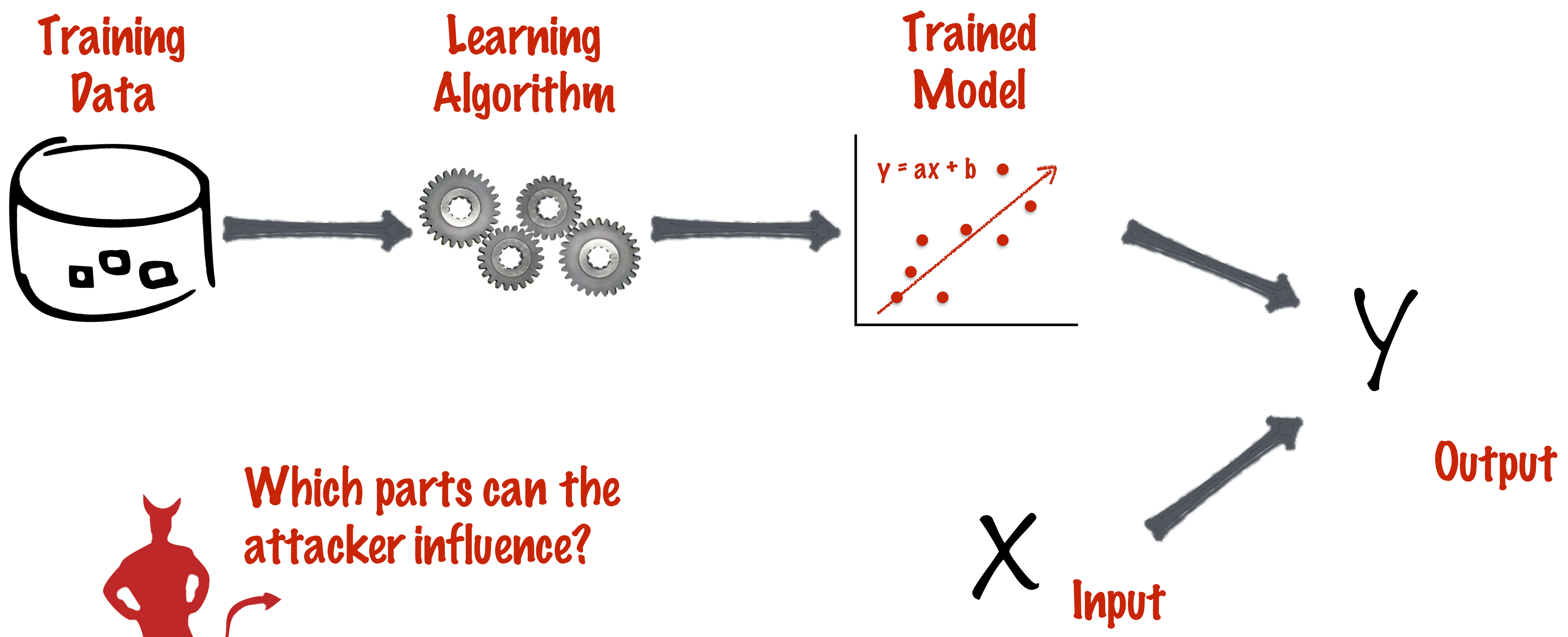
Adversarial Settings in Deep Learning

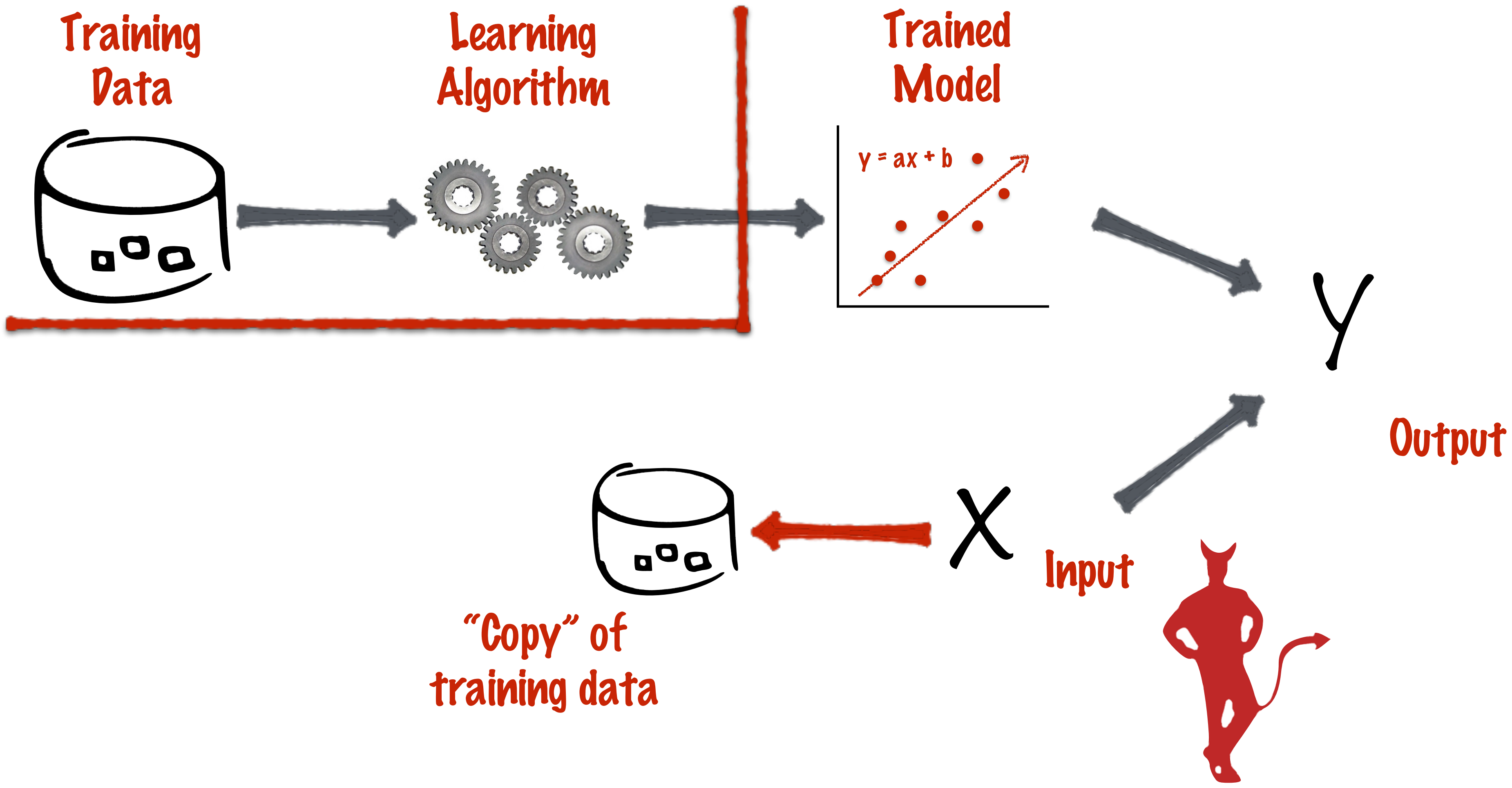
Klas Leino
(slides adapted from Matt Fredrikson)

Machine Learning Pipeline

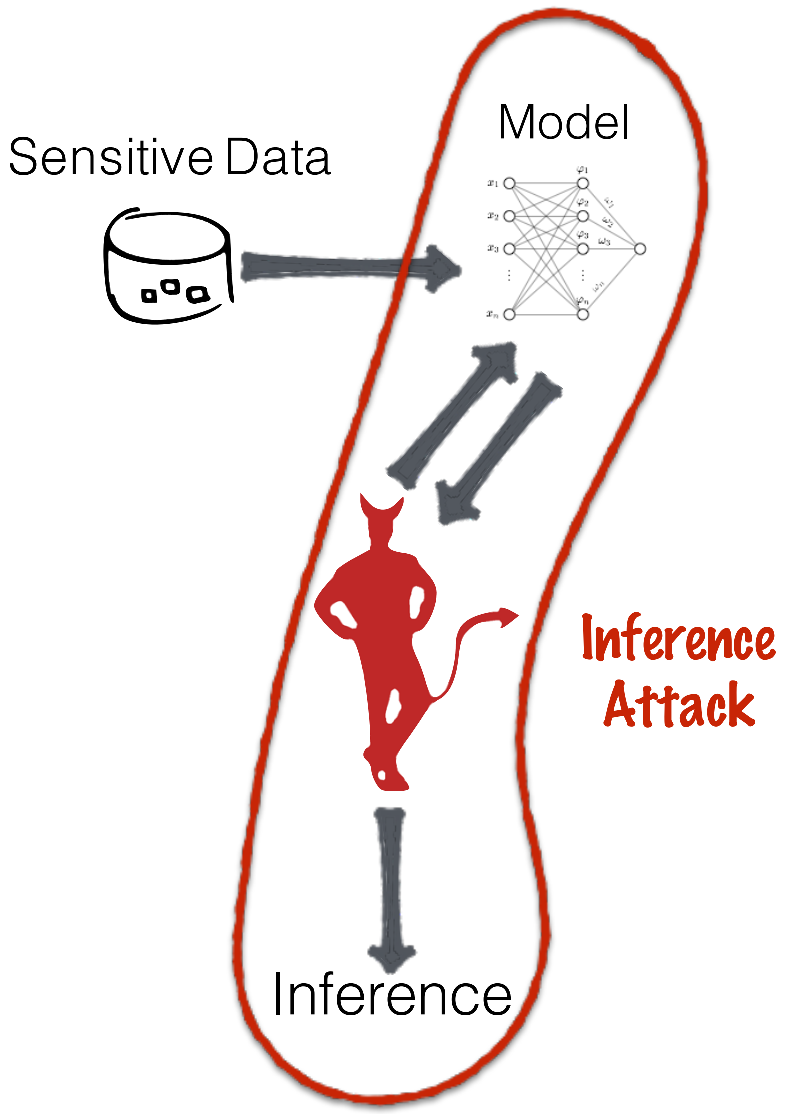


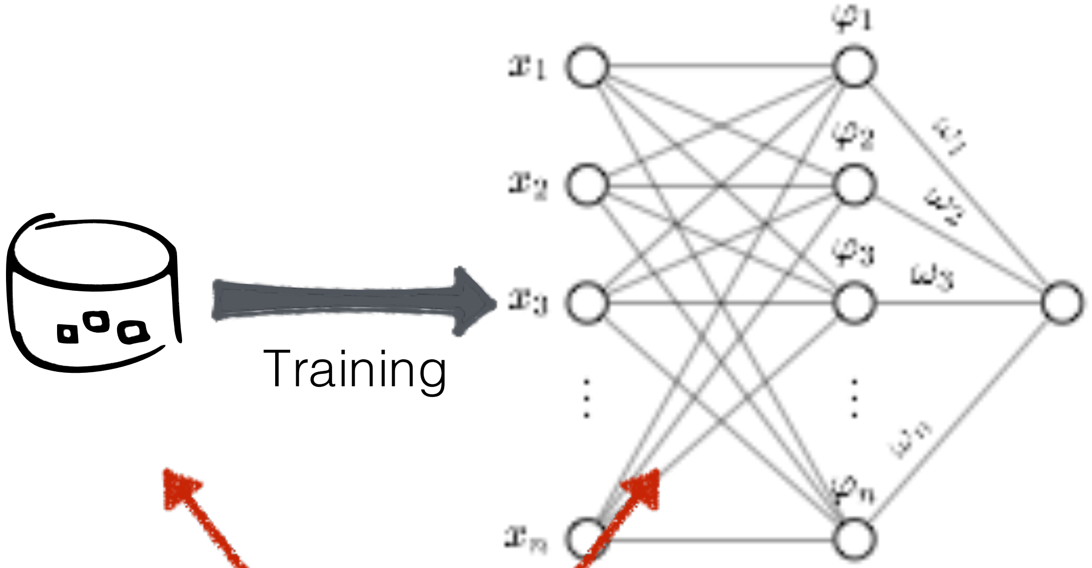
Which parts can the attacker influence?

Threat: Data privacy



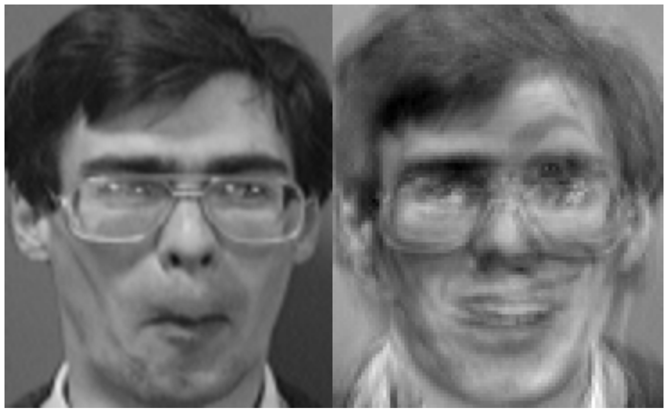
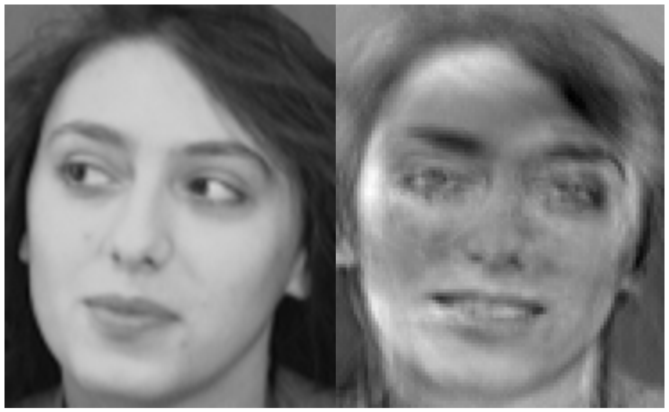
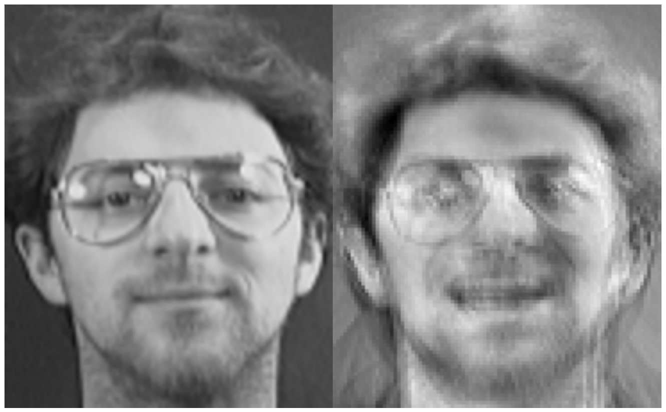
Inference attacks





What does accessing the model tell us about the sensitive training data?

Practical risks: facial recognition models

Can generate images of people in the training set

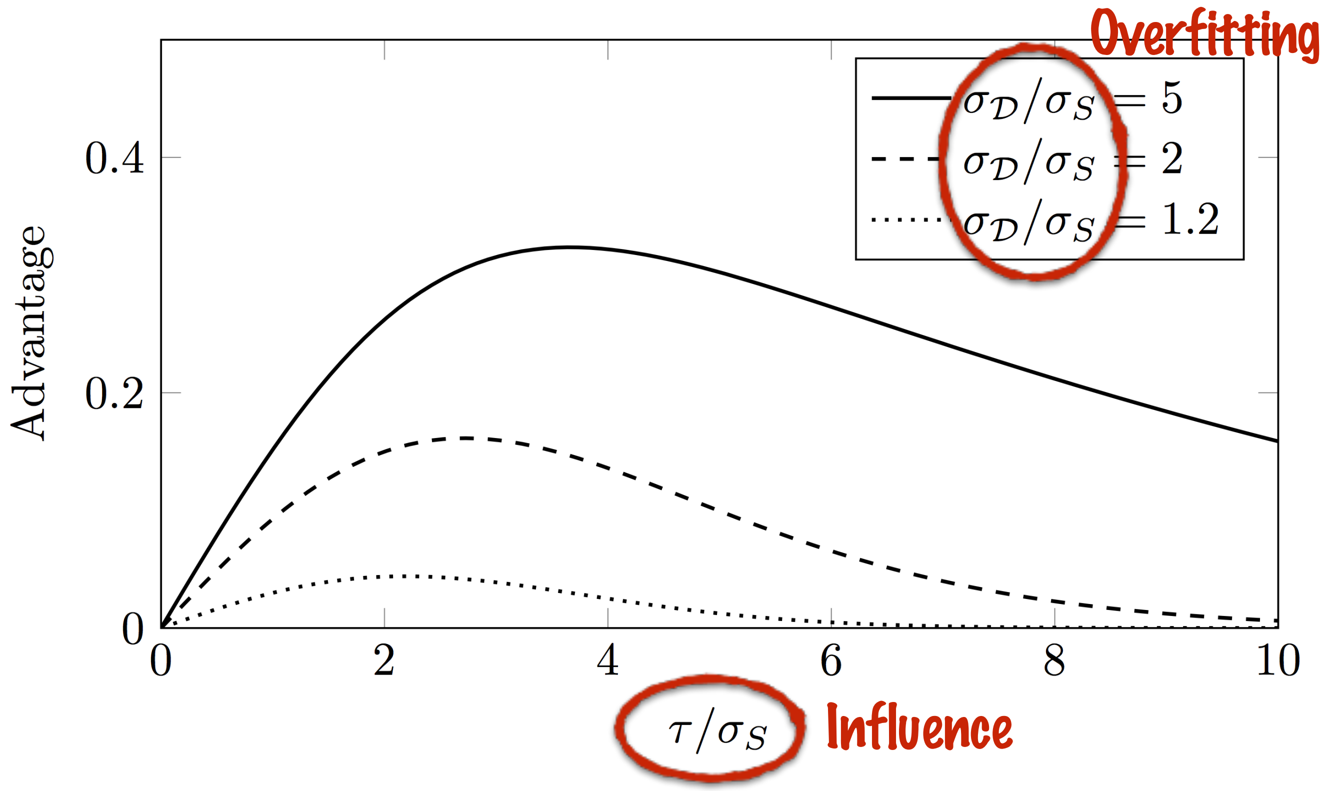
Surveyed several hundred Mechanical Turkers

Turkers could *identify* targeted individual up to 95% of the time

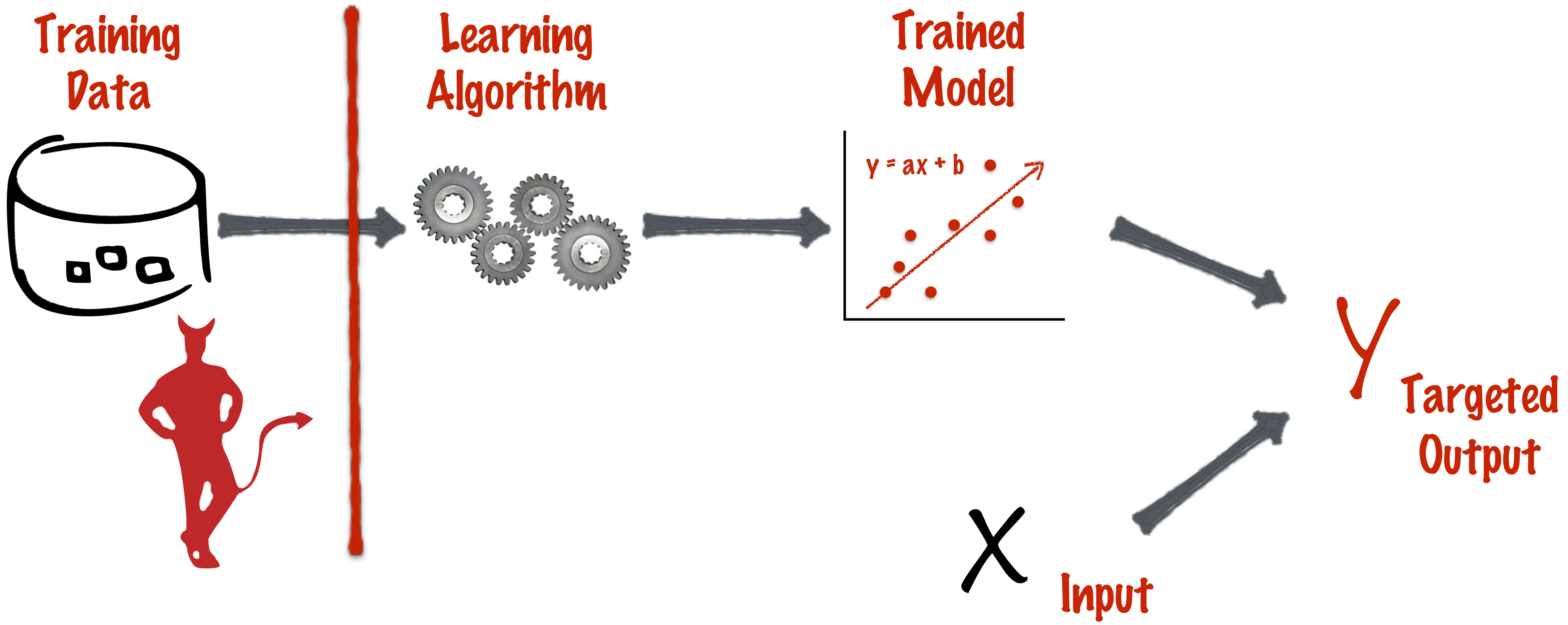
What causes leakage

Hypothesis: advantage comes from *influence* and *overfitting*

- **Influence:** how much does partial input affect the model's output?
- **Overfitting:** ratio of model's error on training and test data

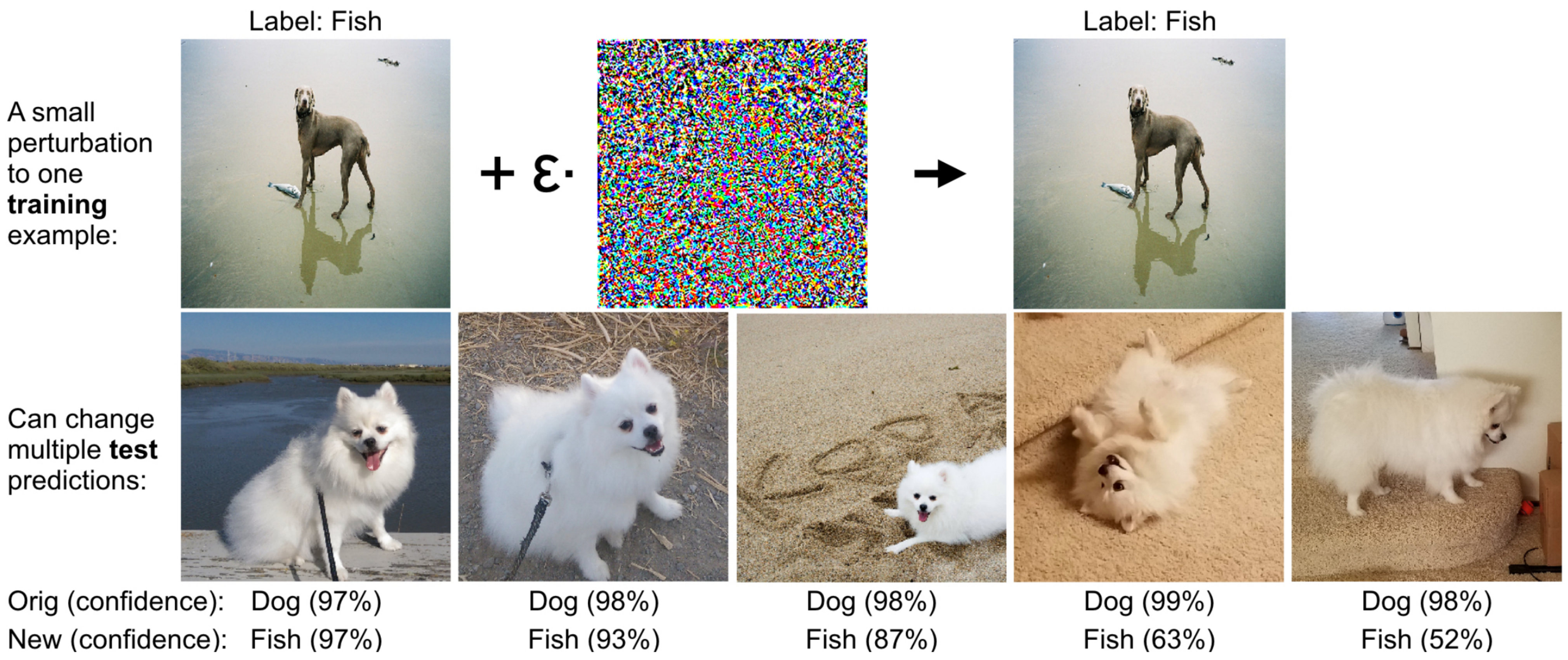


Threat: Data poisoning

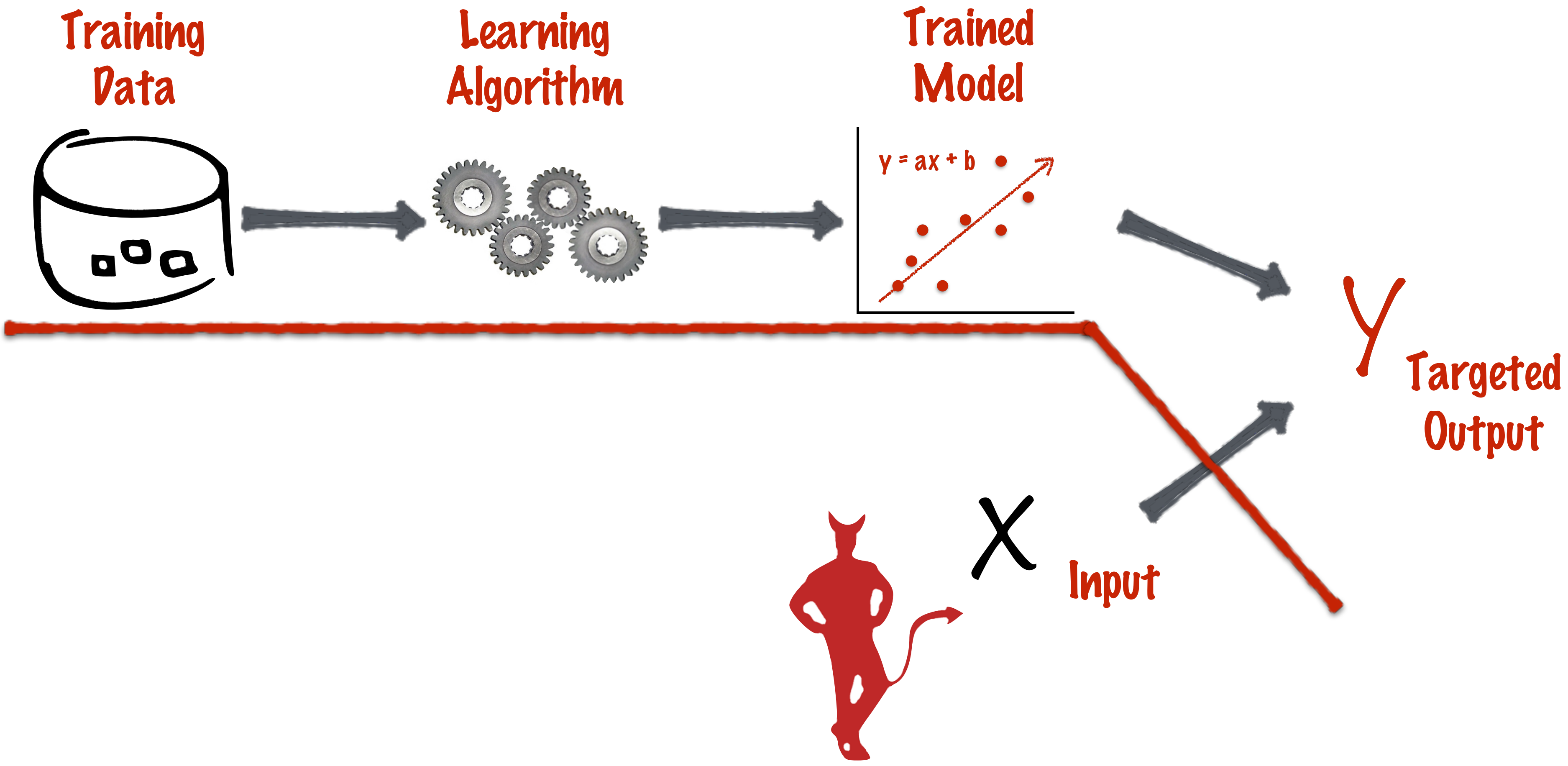


Poisoning by training influence

Koh & Liang 2017, *Understanding Black-box Predictions via Influence Functions*

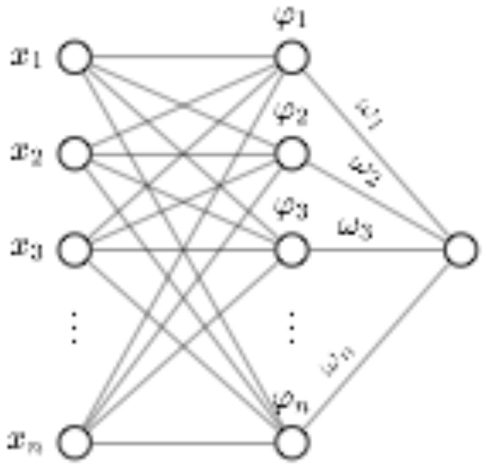


Threat: classifier evasion



Evasion attacks

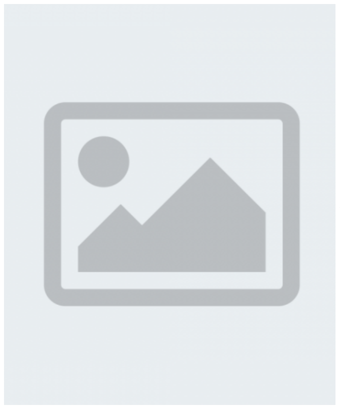
Given:



Bob

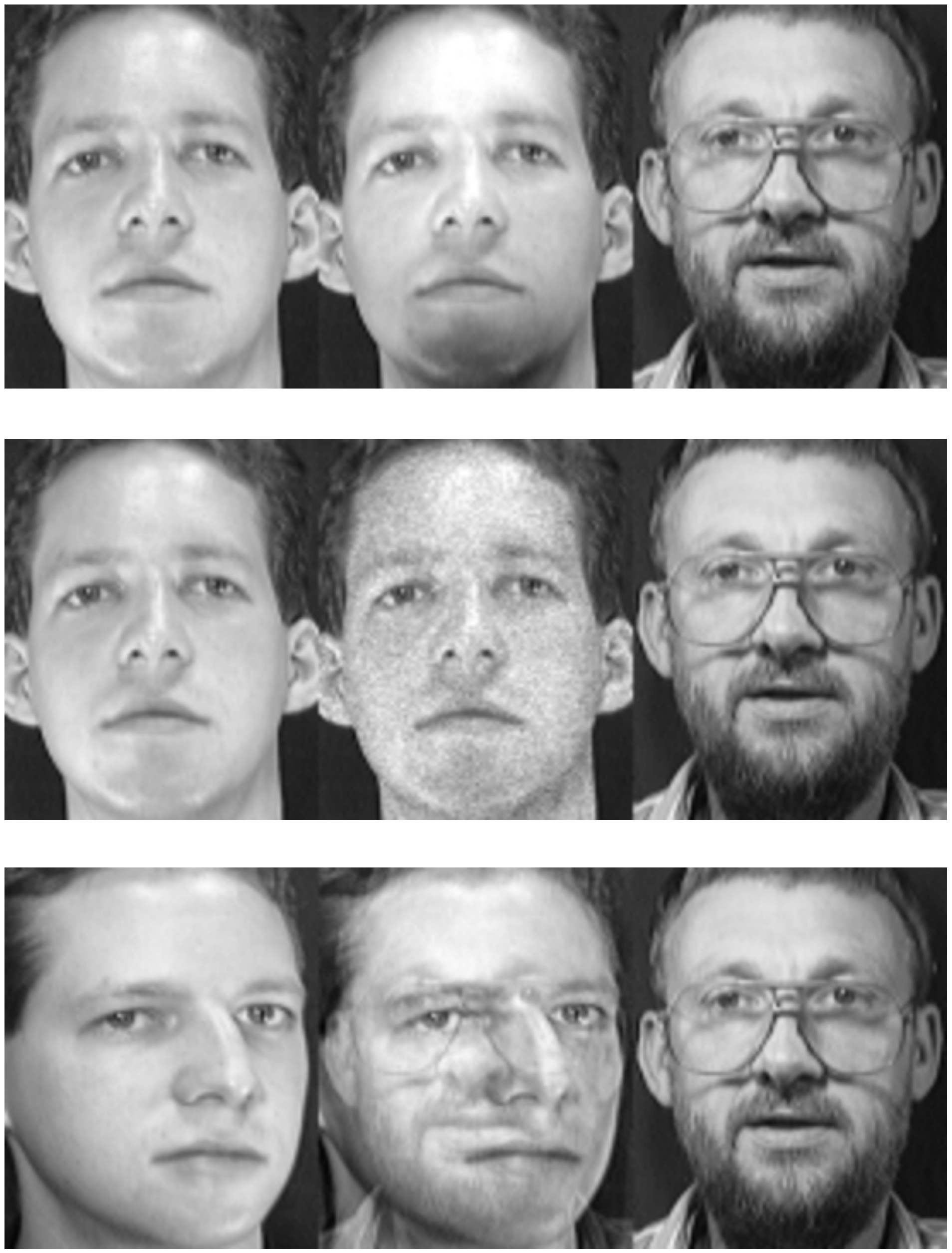
Find:

should look as much like Bob as possible



Such that the model classifies as Joe

Evasion attacks are **easy to find**



Many methods for attack, under varying constraints

Before DL: Evading spam detectors

- Dalvi et al. 2004, *Adversarial Classification*
- Looked at ML techniques for detecting spam
  - Naïve Bayes was tremendously successful
  - …but an obvious target for attackers
- Viewed classification as a game between classifier and adversary
  - Optimal strategy for adversary against unaware classifier
  - Optimal strategy for NB classifier against adversary

Problem definition

- $X = (X_1, X_2, \ldots X_n)$ a set of features
- Instance space $\boldsymbol{X}$. Instance $x \in \boldsymbol{X}$ has feature values x_i
- Instances belong to 2 classes:
  - Positive (spam) are i.i.d. from $P(X|+)$
  - Negative (benign email) are i.i.d. from $P(X|-)$
- Training set $\boldsymbol{S}$, test set $\boldsymbol{T}$

Slide credit: Nilesh Dalvi

Adversarial Classification Game

- **Classifier** tries to learn a function

  $$y_C = C(x)$$

  that will correctly predict classes

- **Adversary** attempts to make **Classifier** classify positive (harmful) instances as negative by modifying an instance x:

  $$x' = A(x)$$

  (note: adversary can not modify negative instances)

Slide credit: Nilesh Dalvi

Costs and utilities

- For the **classifier**:
  - V_i: cost of measuring feature X_i
  - $U_C(y_C, y)$: utility of classifying instance as y_C having true class y
  - Typically $U_C(y_C, y) > 0$ when $y_C = y$, < 0 otherwise
- For the **adversary:**
  - $W_i(x_i, x')$: cost of changing i^{th} feature from x_i to x_i'
  - $U_A(y_C, y)$: utility of classifying as y_c an instance of class y
  - Typically $U_A(-,+) > 0$, $U_A(+,+) < 0$

Slide credit: Nilesh Dalvi

Objectives

- Classifier wants to build *C* that will maximize expected utility taking into account adversary's actions:

$$U_{\mathcal{C}} = \sum_{(x,y)\in\mathcal{XY}} P(x,y)\left[U_{\mathcal{C}}(\mathcal{C}(\mathcal{A}(x)),y) - \sum_{X_i\in\mathcal{X}_{\mathcal{C}}(x)} V_i\right]$$

Utility given modified data

Cost of observing a feature

- Attacker wants to find feature change strategy *A* that will maximize utility given the costs:

$$U_{\mathcal{A}} = \sum_{(x,y)\in\mathcal{XY}} P(x,y)\left[U_{\mathcal{A}}(\mathcal{C}(\mathcal{A}(x)),y) - W(x,\mathcal{A}(x))\right]$$

Utility given modified data

Cost of changing features

The Game

- Assume *all parameters* are known to each player
- Game operates as follows:
  1. Classifier starts assuming data is unaltered
  2. Adversary deploys optimal plan *A(x)* against classifier
  3. Classifier deploys optimal classifier *C(A(x))* against adversary
  4. …iterate until convergence

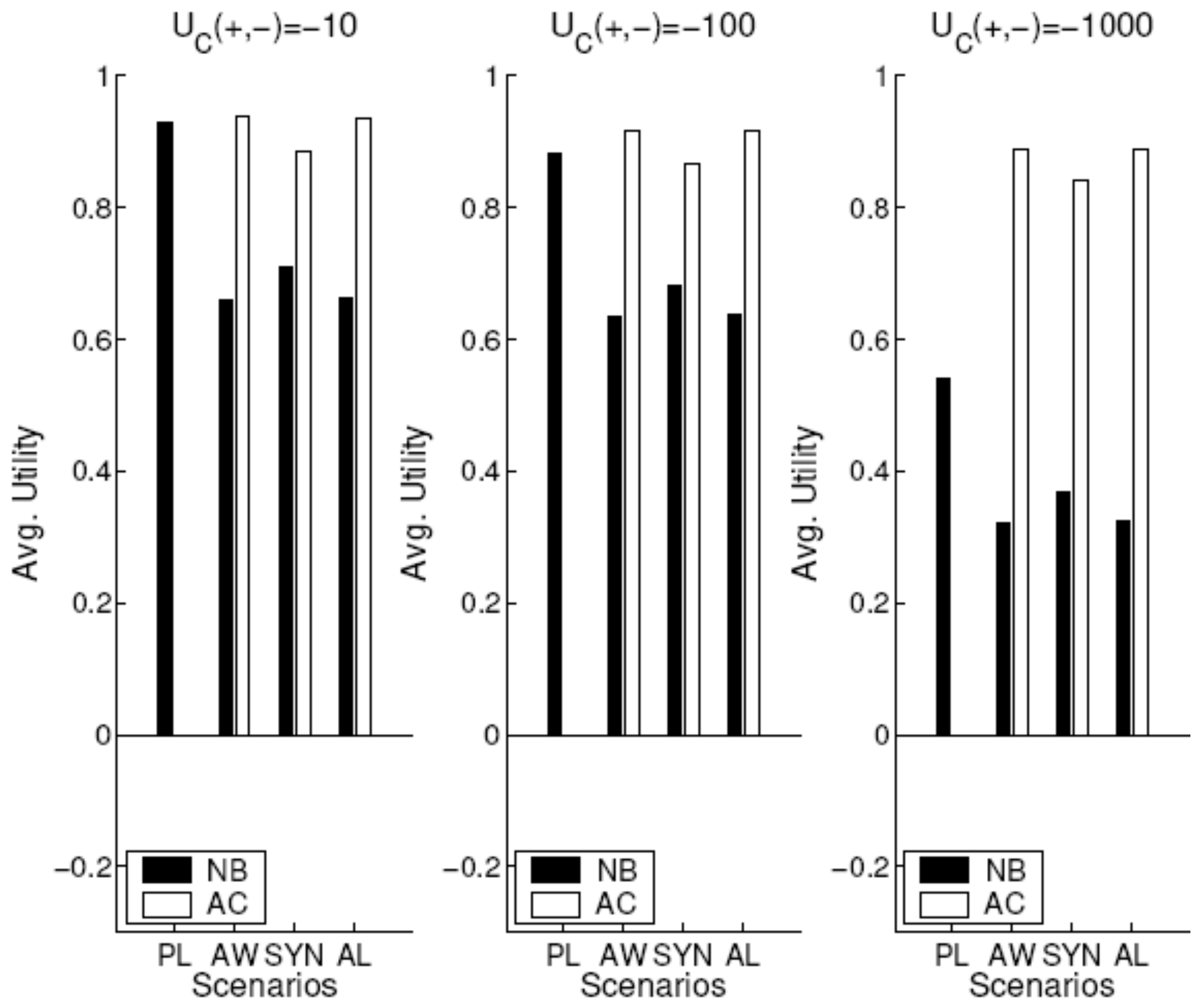
Key result: Adversary's solution can be characterized by an integer-linear program

Slide credit: Nilesh Dalvi

Results: Classifier's Utility

Scenarios:

- *AddWords*: add up to 20 words
- *AddLength*: add up to 200 chars
- *Synonmy*: change up to 20 synonyms



Slide credit: Nilesh Dalvi

Fast Forward: Evading Deep Learning

Szegedy et al. 2014, *Intriguing properties of neural networks*

"We describe a way to traverse the manifold represented by the network in an efficient way and finding adversarial examples in the input space"

Minimize $\|r\|_2$ subject to: Minimize to make "inconspicuous"

1. $f(x+r) = l$ Attacker's main objective
2. $x + r \in [0,1]^m$ Still a valid input

Attacking ImageNet

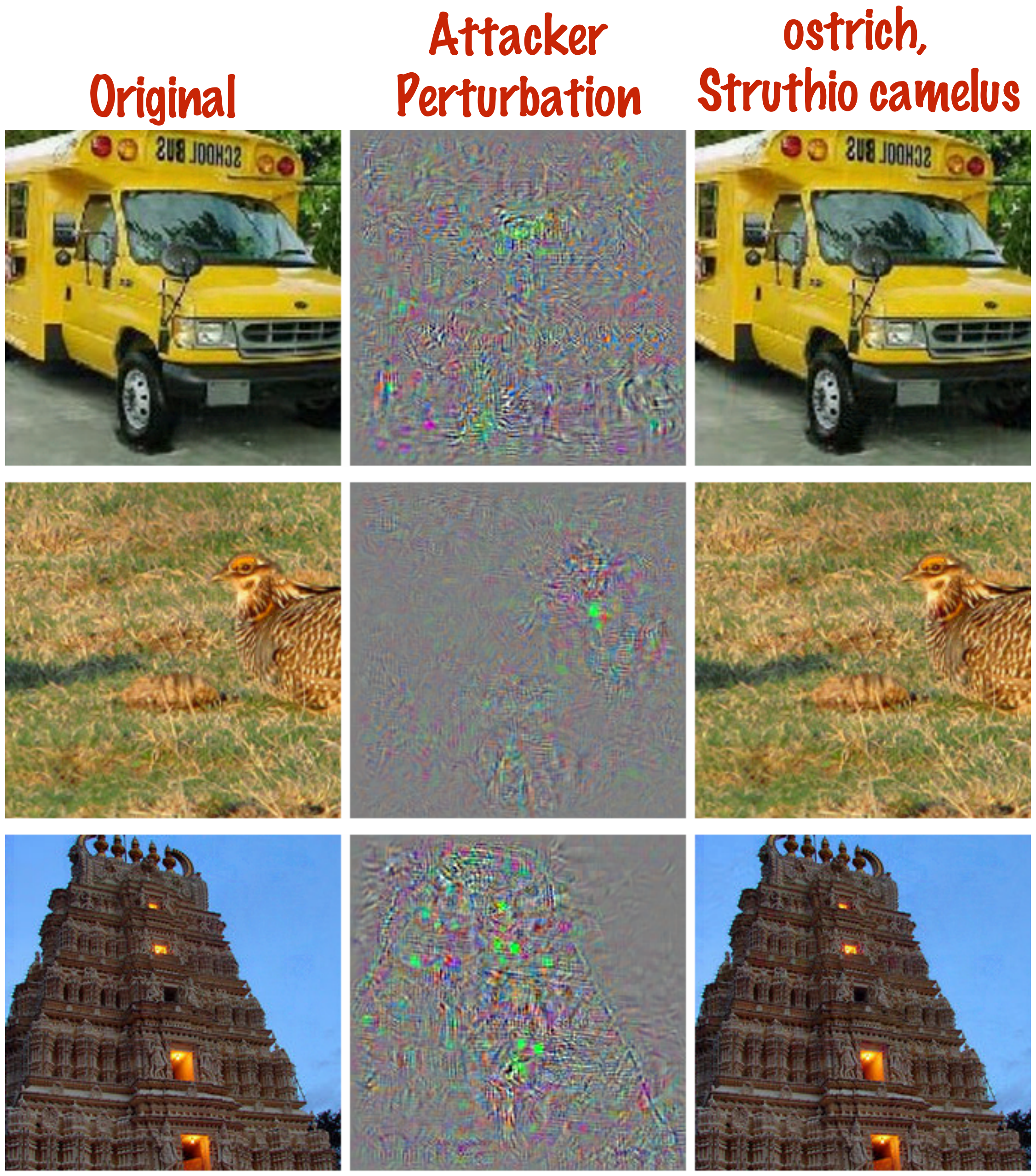


Image credit: Szegedy et al., *Intriguing Properties of Neural Networks*, 2014

Jacobian-based **S**aliency **M**ap **A**pproach

Papernot et al. 2016, *The Limitations of DL in Adversarial Settings*

Basic approach: understand how inputs affect outputs

1. Compute forward derivative for each feature
2. Construct *saliency map*: **input perturbations → output variations**
3. Modify sample, focusing on most influential feature
4. Iterate, until output label changes

Adversarial Saliency Maps

For a softmax classifier: $label(\mathbf{X}) = \arg\max_j \mathbf{F}_j(\mathbf{X})$

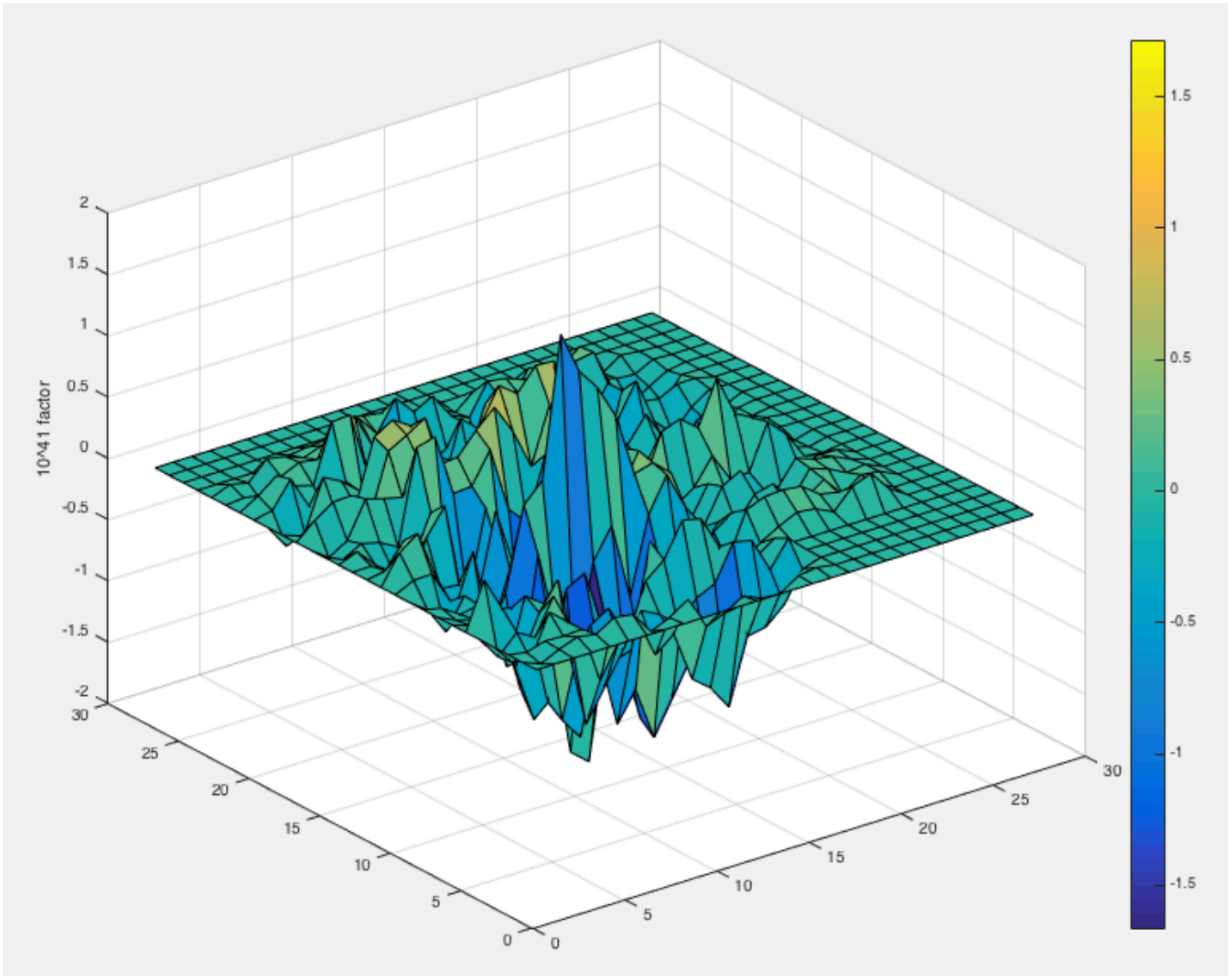
Feature moves away from target, or towards other labels

$$S(\mathbf{X}, t)[i] = \begin{cases} 0 \text{ if } \frac{\partial \mathbf{F}_t(\mathbf{X})}{\partial \mathbf{X}_i} < 0 \text{ or } \sum_{j \neq t} \frac{\partial \mathbf{F}_j(\mathbf{X})}{\partial \mathbf{X}_i} > 0 \\ \left(\frac{\partial \mathbf{F}_t(\mathbf{X})}{\partial \mathbf{X}_i}\right) \left|\sum_{j \neq t} \frac{\partial \mathbf{F}_j(\mathbf{X})}{\partial \mathbf{X}_i}\right| \text{ otherwise} \end{cases}$$

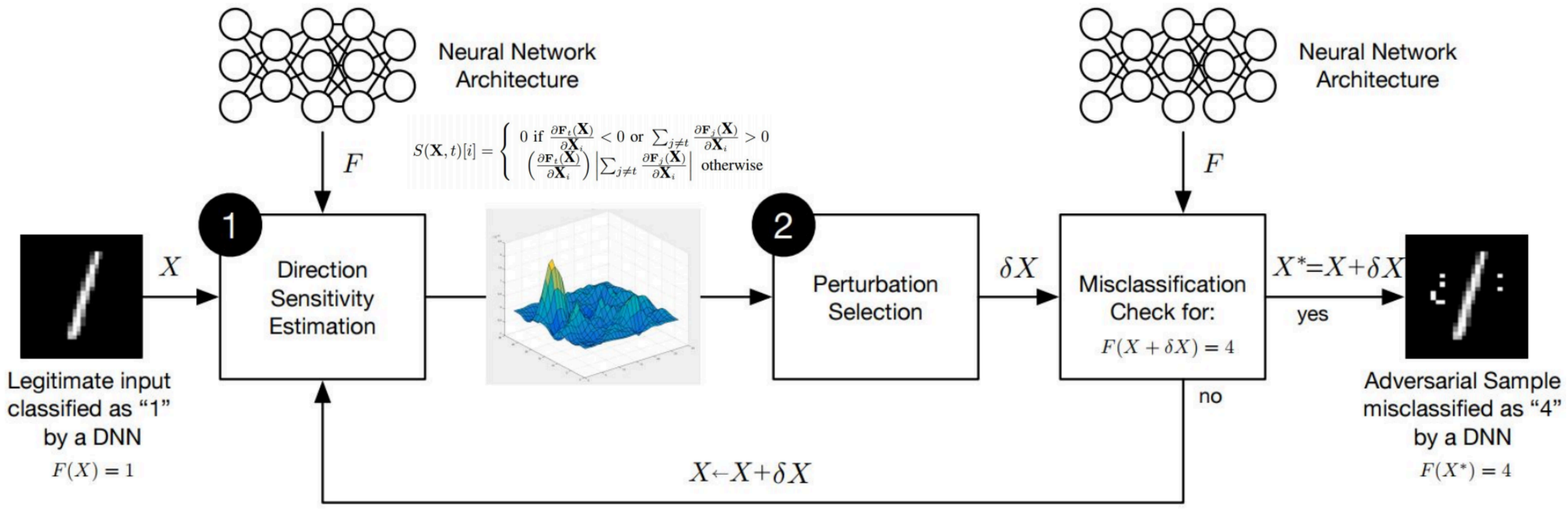
Attacker's target class

Measure how much output moves toward target

Adversarial Saliency Maps: MNIST



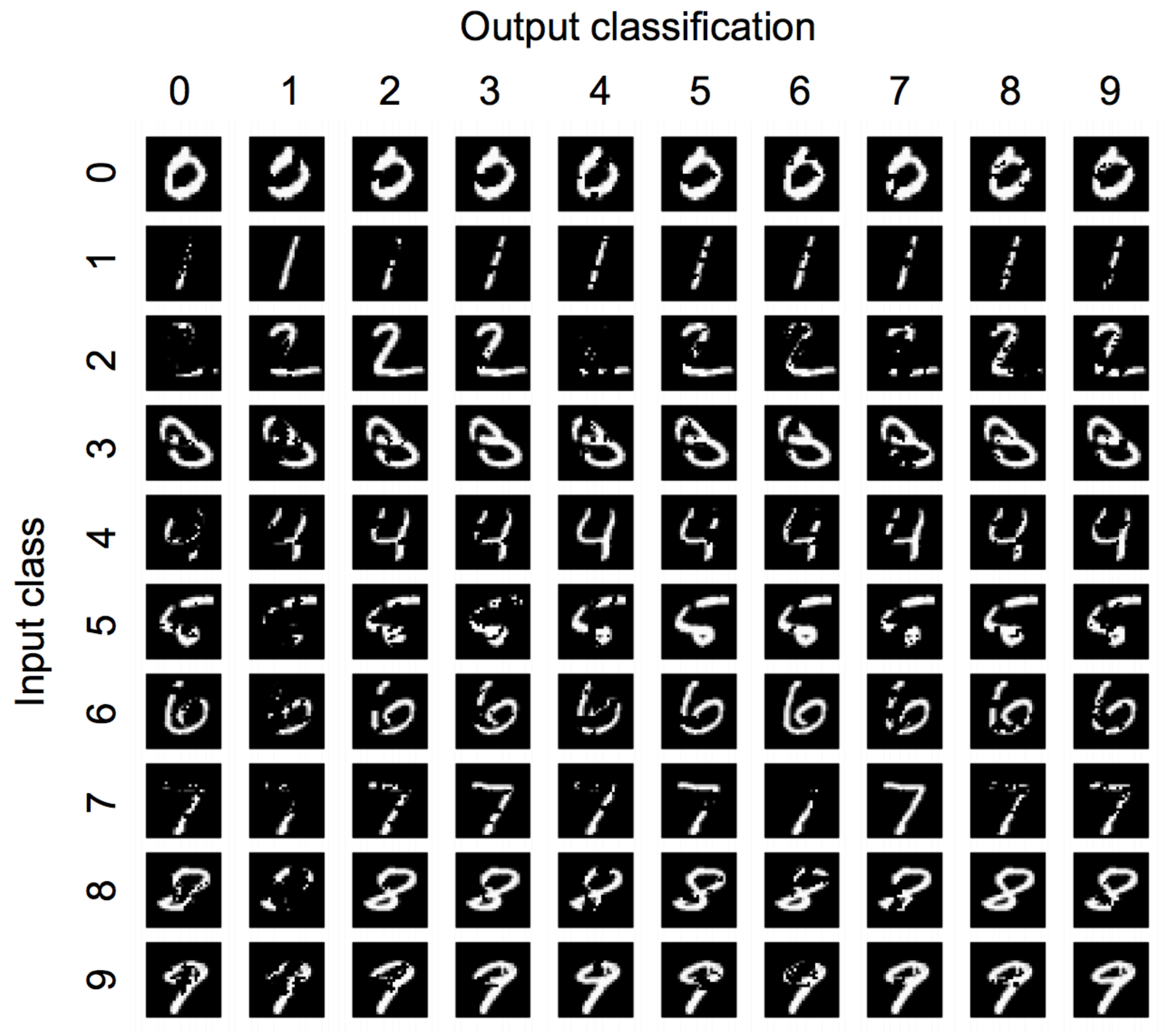
JSMA Greedy Search



$$S(\mathbf{X}, t)[i] = \begin{cases} 0 \text{ if } \frac{\partial \mathbf{F}_t(\mathbf{X})}{\partial \mathbf{X}_i} < 0 \text{ or } \sum_{j \neq t} \frac{\partial \mathbf{F}_j(\mathbf{X})}{\partial \mathbf{X}_i} > 0 \\ \left(\frac{\partial \mathbf{F}_t(\mathbf{X})}{\partial \mathbf{X}_i}\right) \left|\sum_{j \neq t} \frac{\partial \mathbf{F}_j(\mathbf{X})}{\partial \mathbf{X}_i}\right| \text{ otherwise} \end{cases}$$

Slide credit: adapted from Nicolas Papernot

JSMA on MNIST



JSMA on Malware Classifiers

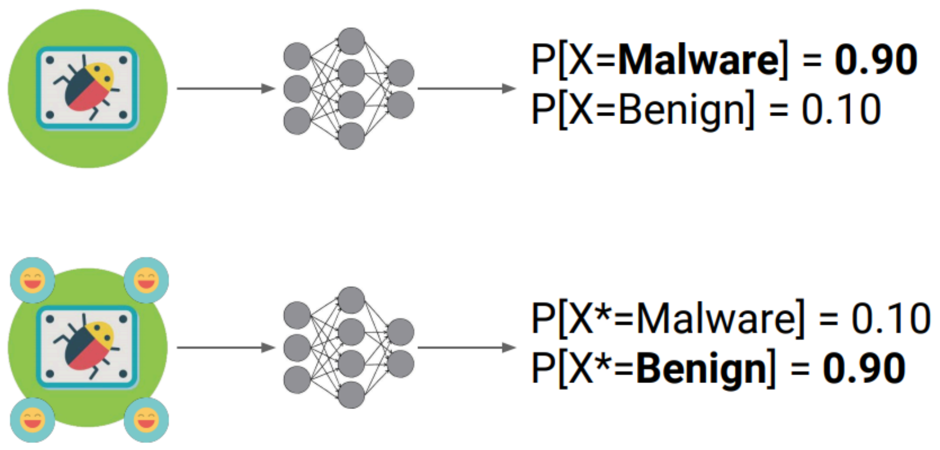
Grosse et al. 2016, *Adversarial Perturbations Against DNNs for Malware*

Add constraints to JSMA

- Only **add** features, i.e. don't remove malicious behavior
- Use **manifest** features, i.e. easy to modify malware

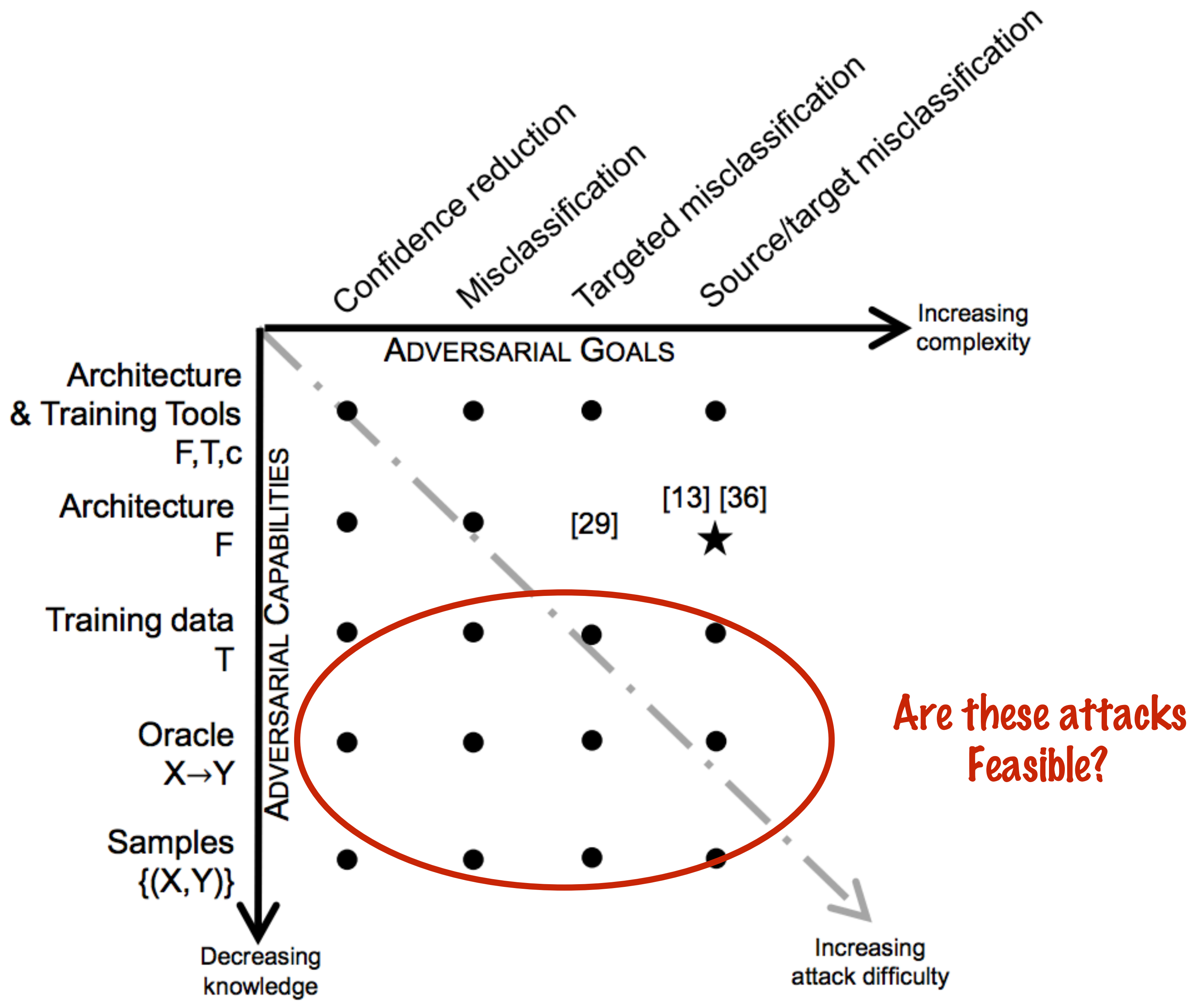
Works well in practice

- Classifier: **98%** accuracy
- Evasion successful in **63% of attempts**



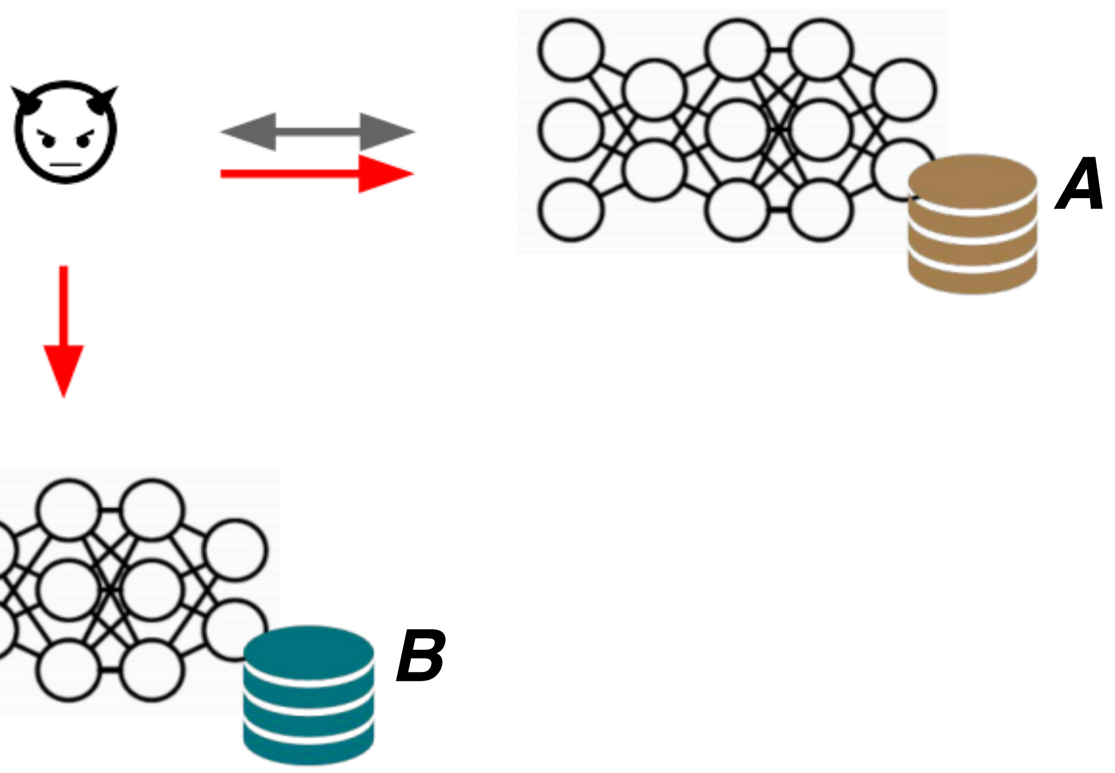
Slide credit: adapted from Nicolas Papernot

Threat taxonomy



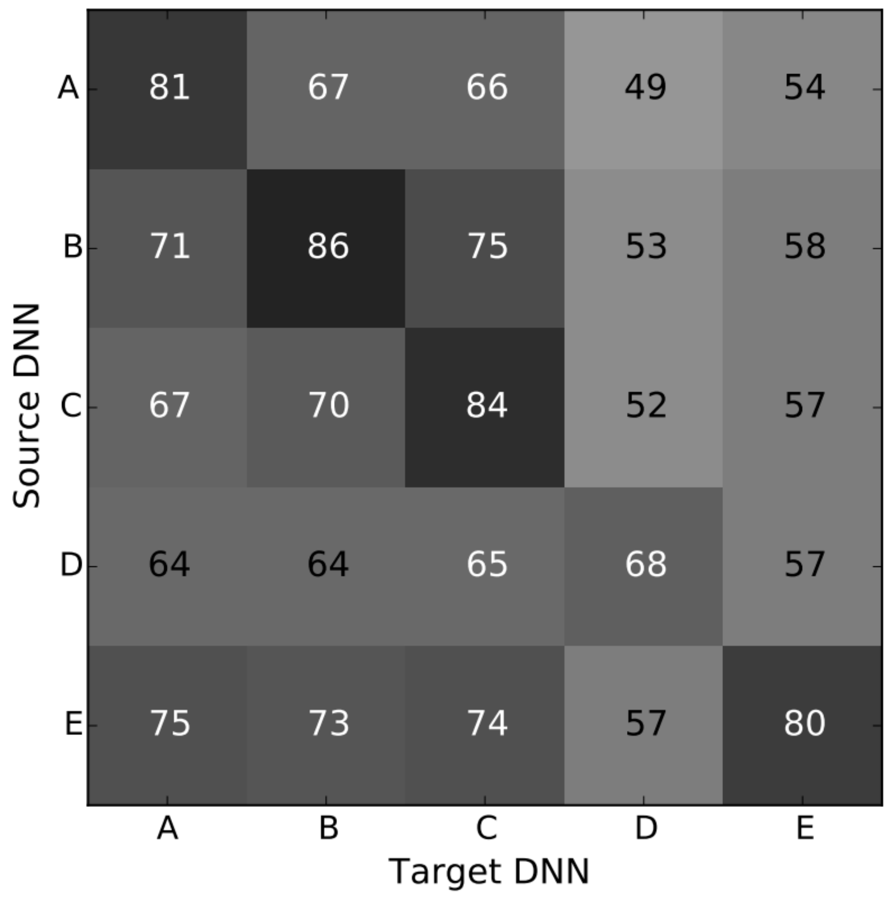
Transferability

Samples that evade model *A* are likely* to evade model *B* as well

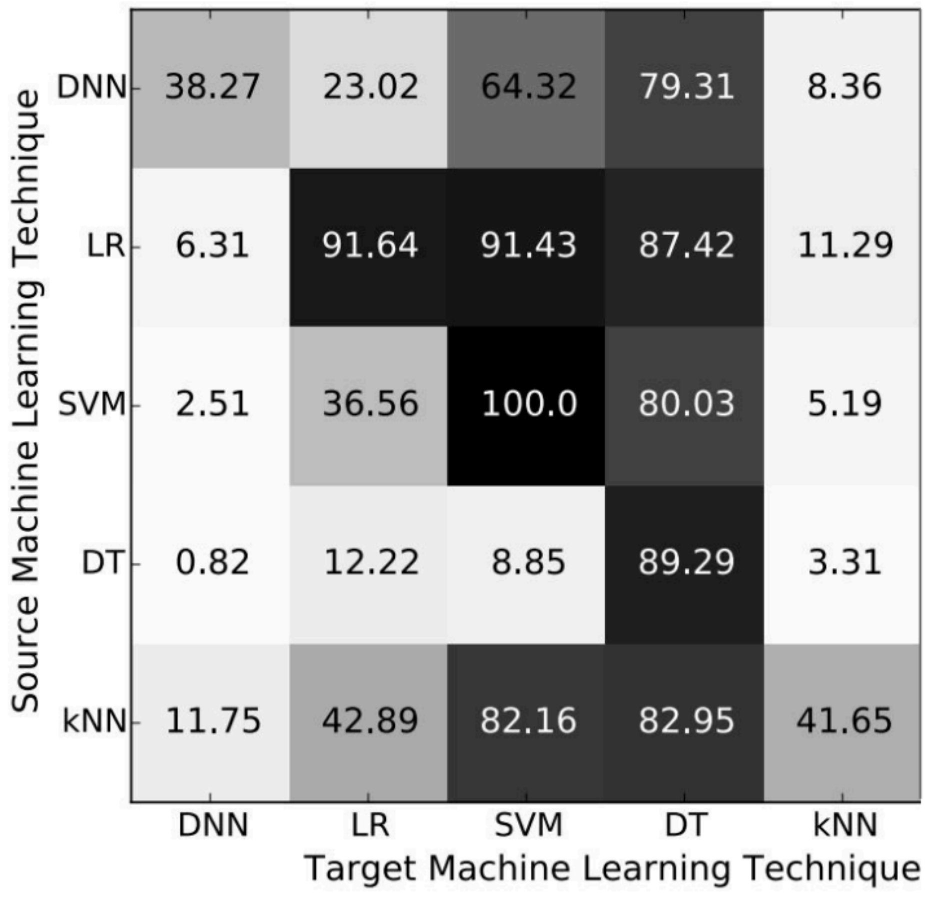


Slide credit: adapted from Nicolas Papernot

Cross-model transferability



Source DNN \ Target DNN	A	B	C	D	E
A	81	67	66	49	54
B	71	86	75	53	58
C	67	70	84	52	57
D	64	64	65	68	57
E	75	73	74	57	80

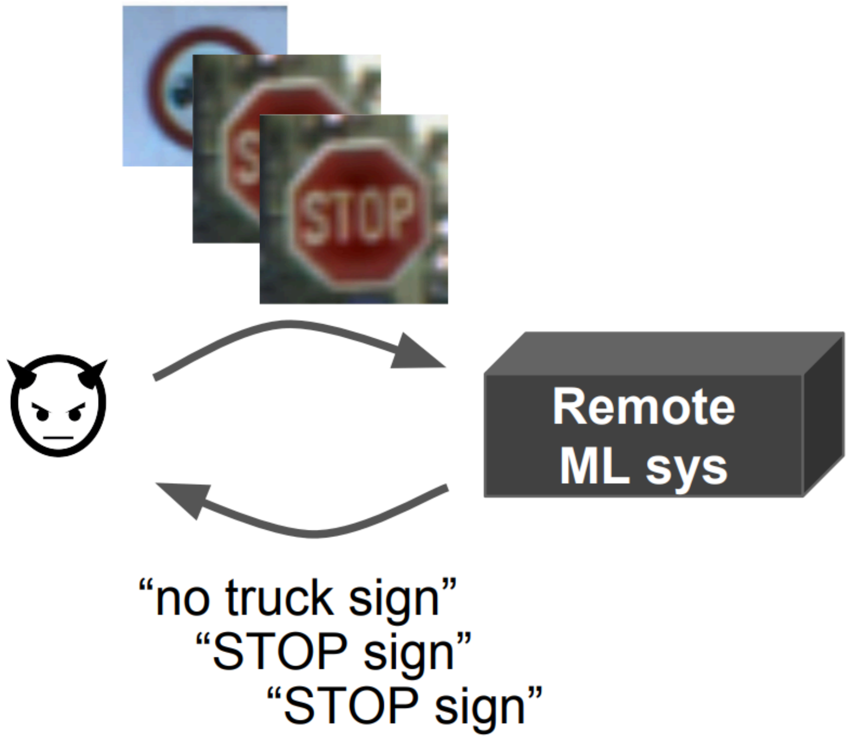


Source Machine Learning Technique \ Target Machine Learning Technique	DNN	LR	SVM	DT	kNN
DNN	38.27	23.02	64.32	79.31	8.36
LR	6.31	91.64	91.43	87.42	11.29
SVM	2.51	36.56	100.0	80.03	5.19
DT	0.82	12.22	8.85	89.29	3.31
kNN	11.75	42.89	82.16	82.95	41.65

Image credit: Nicolas Papernot

Black-box attacks

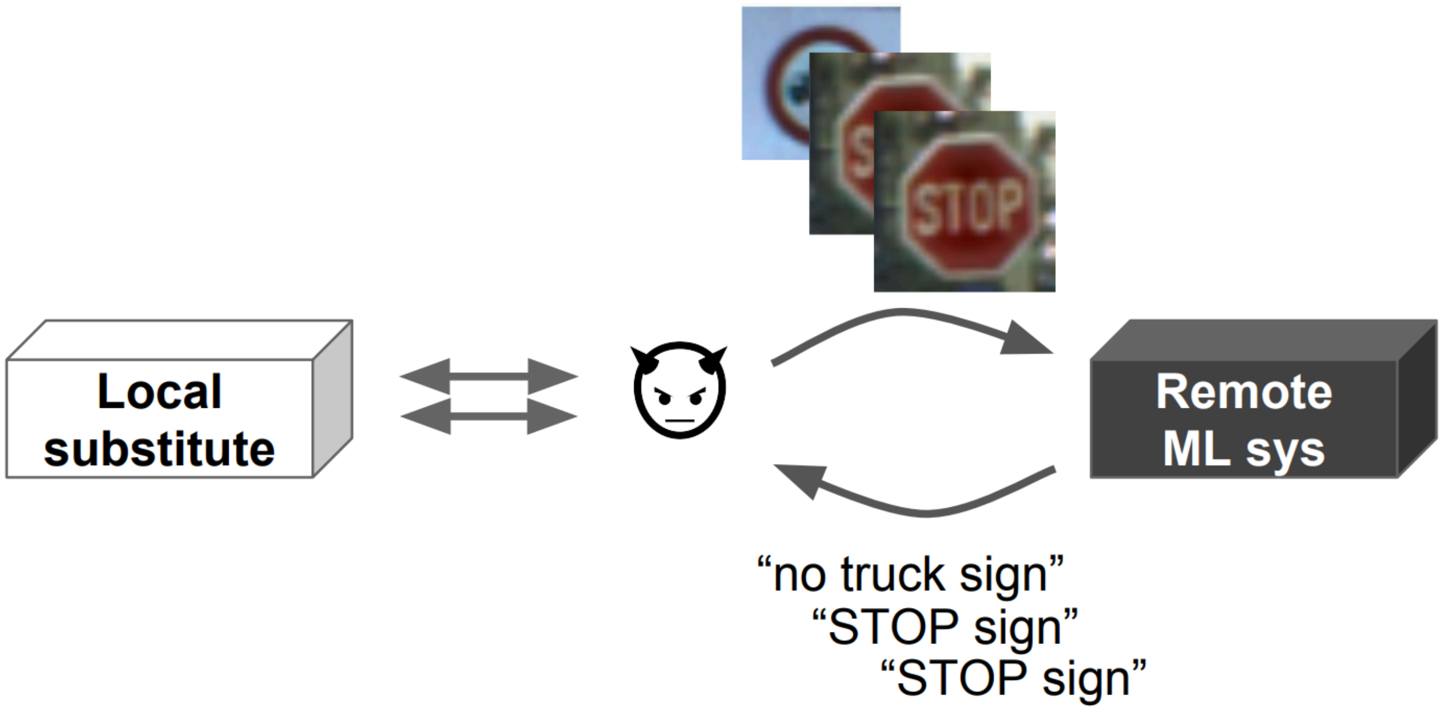
Step 1: Query black-box models on inputs of adversary's choice



Slide credit: adapted from Nicolas Papernot

Black-box attacks

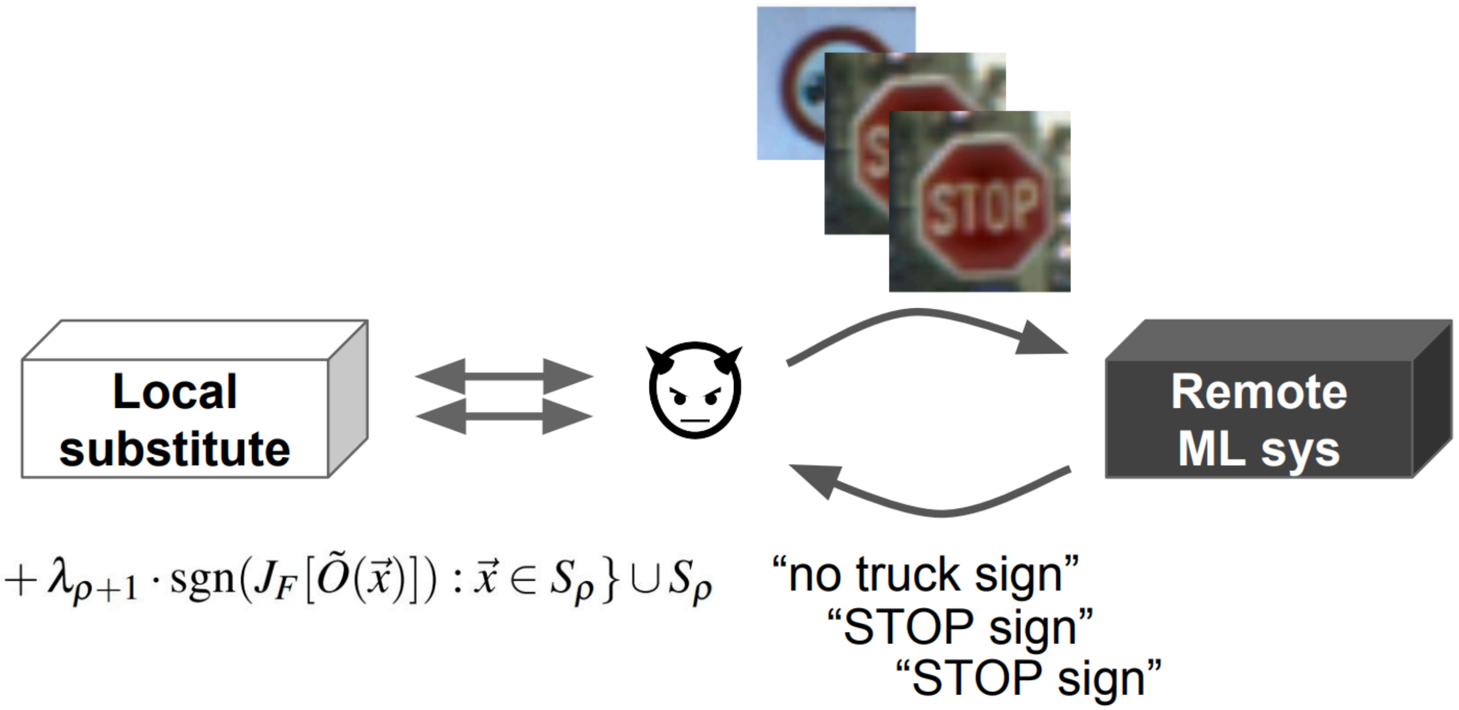
Step 2: Train a local substitute from black-box model's labels



Slide credit: adapted from Nicolas Papernot

Black-box attacks

Step 3: Augment dataset with samples that approach the local model's decision boundary

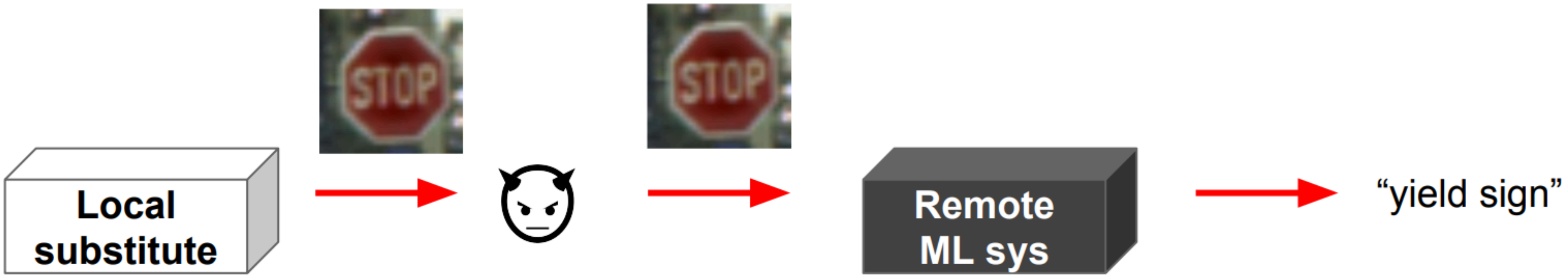


$$S_{\rho+1} = \{\vec{x} + \lambda_{\rho+1} \cdot \mathrm{sgn}(J_F[\tilde{O}(\vec{x})]) : \vec{x} \in S_\rho\} \cup S_\rho$$

Slide credit: adapted from Nicolas Papernot

Black-box attacks

Step 4: Transfer attacks from local model to black-box remote



Slide credit: adapted from Nicolas Papernot

Black-box results

Remote Platform	ML technique	Number of queries	Adversarial examples misclassified (after querying)
MetaMind	Deep Learning	6,400	84.24%
amazon webservices™	Logistic Regression	800	96.19%
Google Cloud Platform	Unknown	2,000	97.72%

All remote classifiers are trained on the MNIST dataset (10 classes, 60,000 training samples)

Image credit: Nicolas Papernot

What causes attacks?

Hypothesis: Overfitting

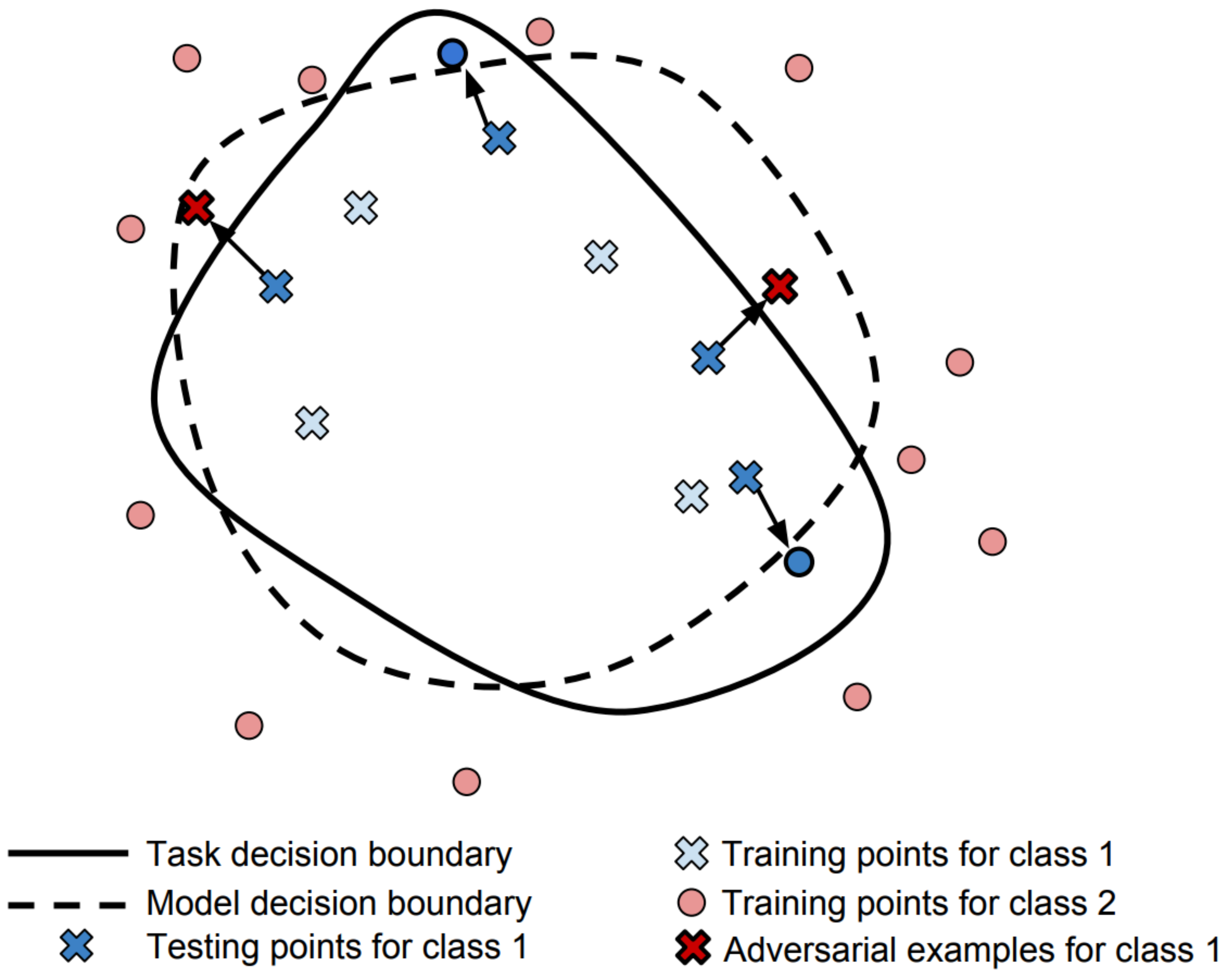
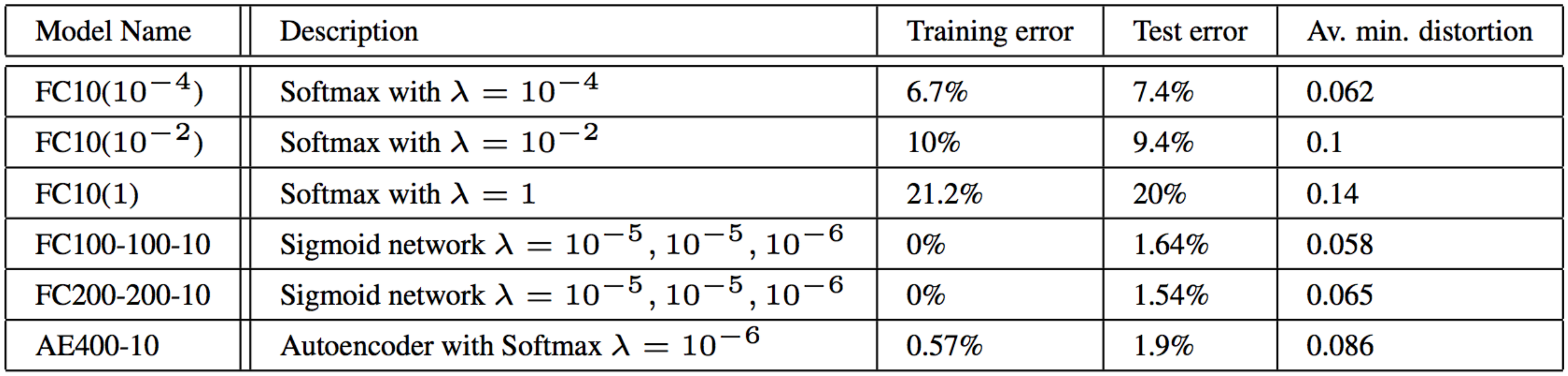

Task decision boundary
Model decision boundary
Testing points for class 1
Training points for class 1
Training points for class 2
Adversarial examples for class 1

Image credit: Nicolas Papernot

Attacks not explained by overfitting



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

Szegedy et al., *Intriguing Properties of Neural Networks*

What causes vulnerability?

Hypothesis: Linearity

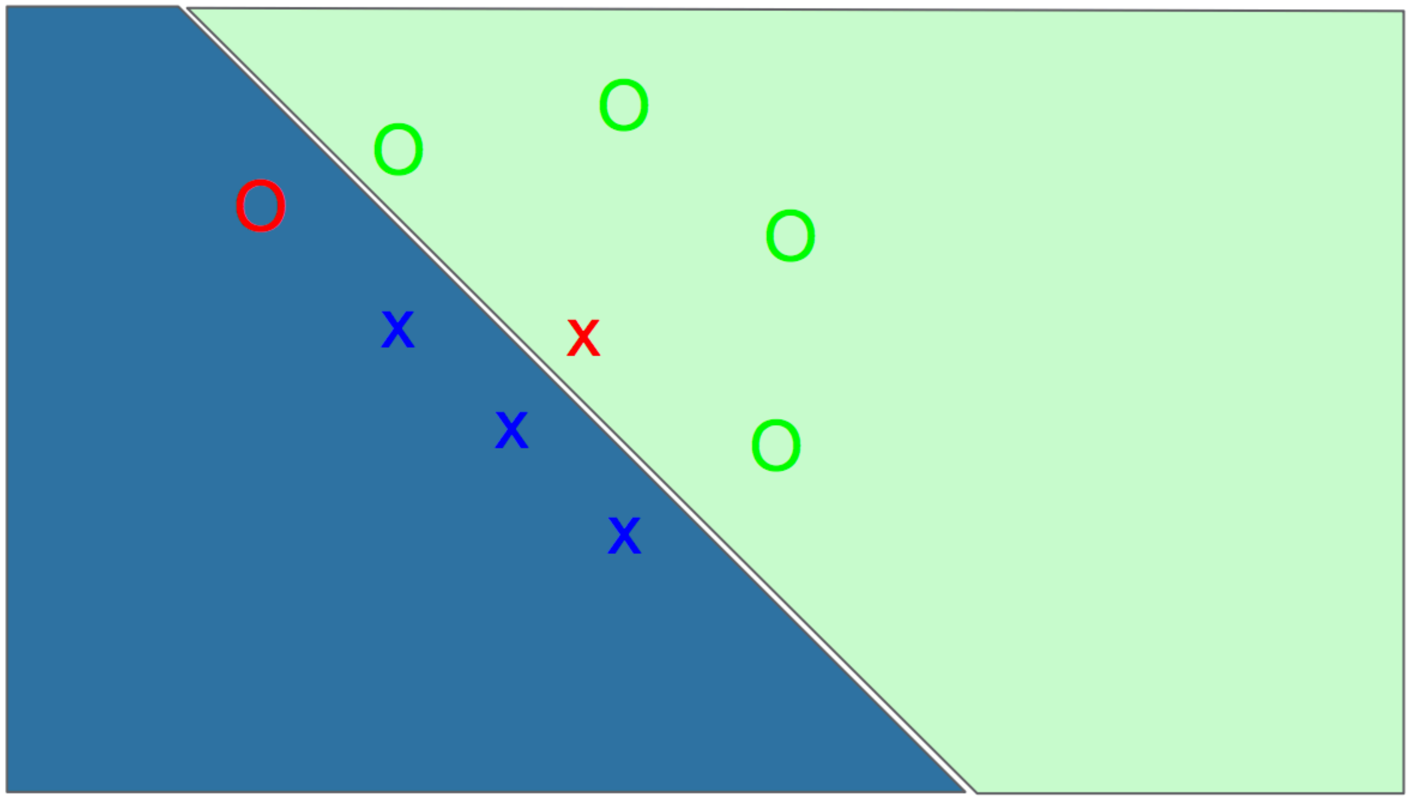
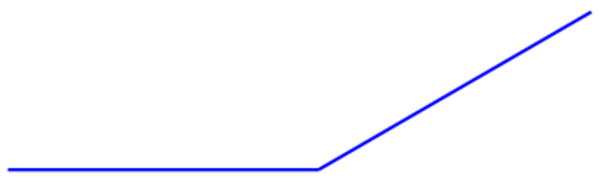


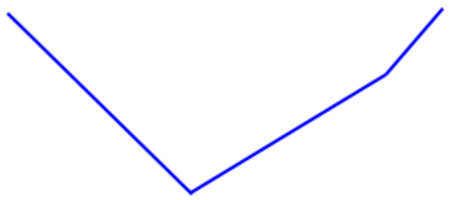
Image credit: Ian Goodfellow

Deep nets are piecewise linear

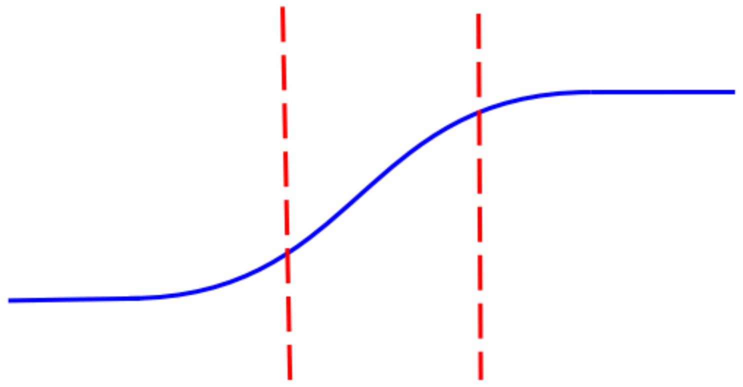
Rectified linear unit



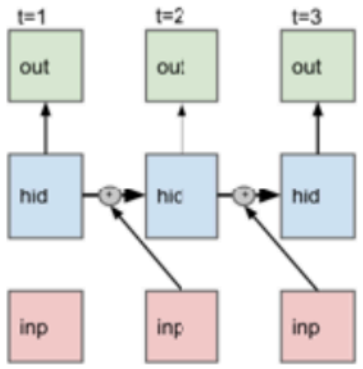
Maxout



Carefully tuned sigmoid



LSTM



Slide credit: Ian Goodfellow

Fast Gradient Sign Method

Attack parameter

Training cost function

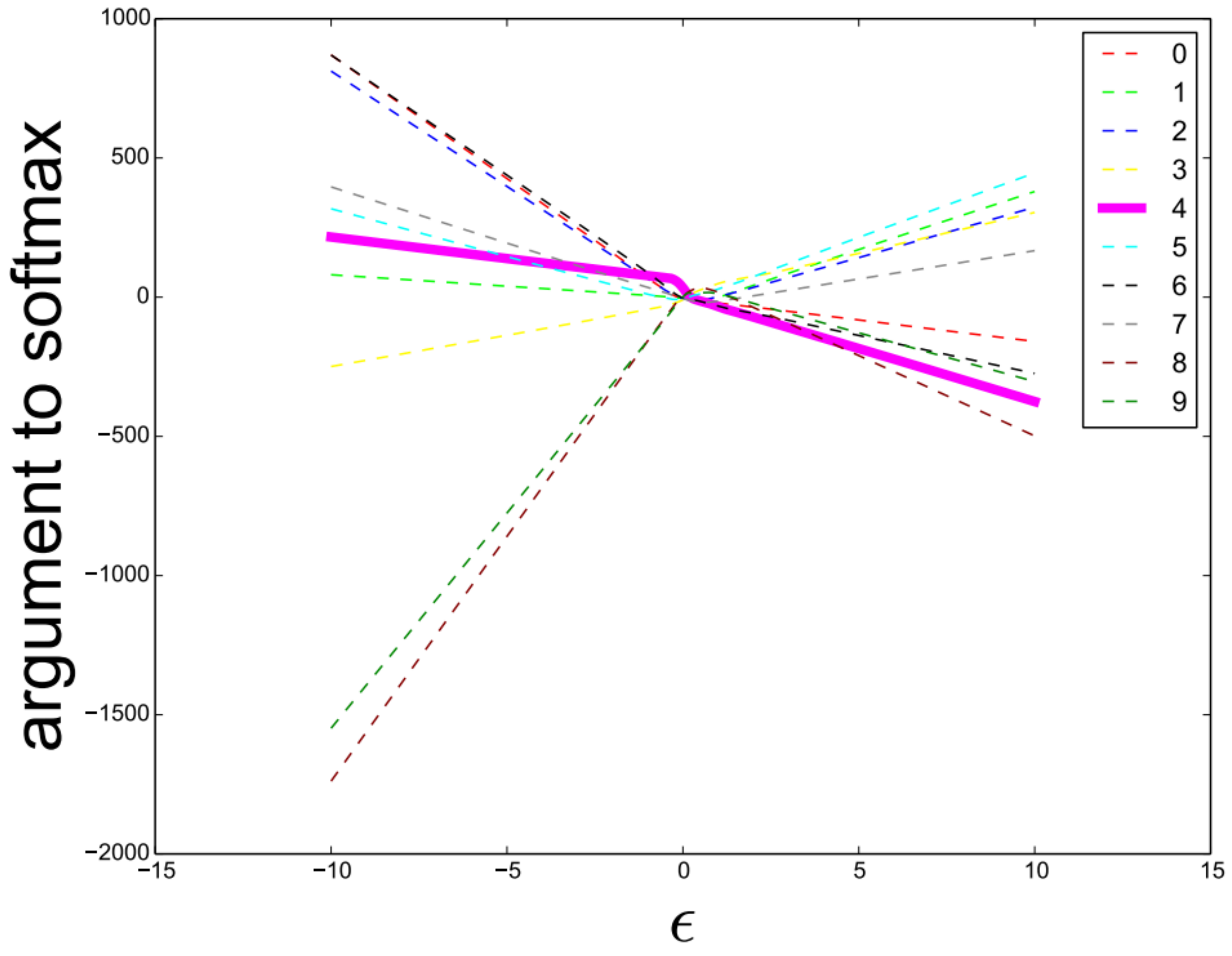
$$\boldsymbol{\eta} = \epsilon \operatorname{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y)\right)$$

Attacker's perturbation

Input to the model

Attacker's target class

Excessive linearity



Goodfellow et al., *Explaining and Harnessing Adversarial Examples*

Defenses

Hot research topic: prevent evasion attacks

1. Train on adversarial samples with correct labels
2. Classify using ensembles
3. Compress/de-noise images
4. Train classifiers to detect attacks
5. Smooth gradients around training points

None of these work against novel attacks!

Provable Defenses

Goal: Given classifier ***f***, prove that for all ***x*** there are no ***x'*** "near" ***x*** where ***f(x) ≠ f(x')***.

Many challenges

1. How to define "near" precisely enough for proof?
2. State space is large; verification is expensive
3. Evidence so far is that this isn't ever true
4. …so how to build (and then prove) classifiers with this property?

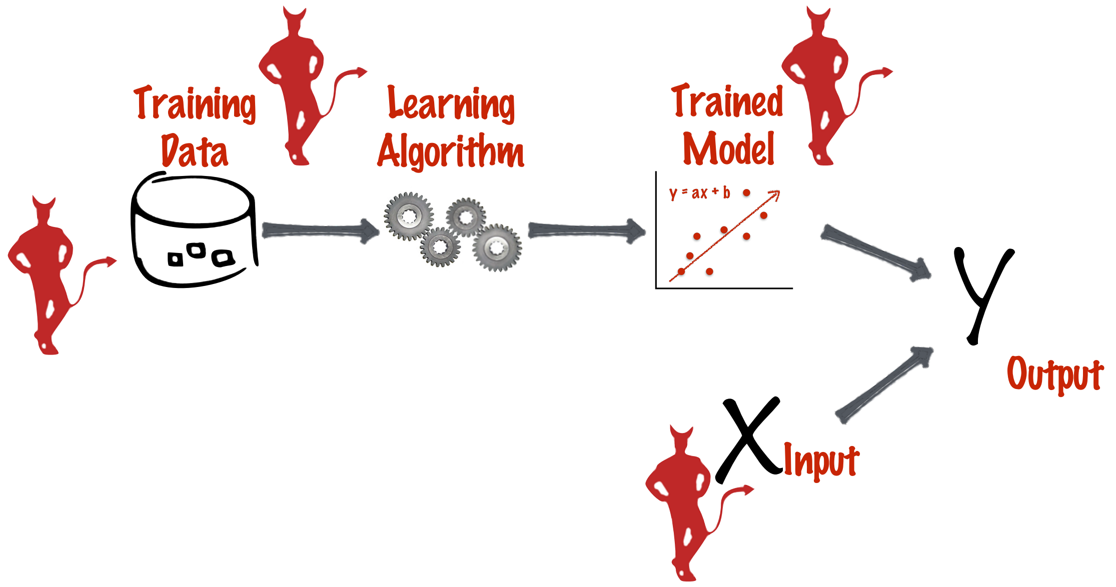
Hardening models: a compromise

New goal: Given classifier ***f***, prove that for all ***x*** in the training data there are no **x'** "near" **x** where **f(x) ≠ f(x')**.

Still challenging

1. How to define "near" precisely enough for proof?
2. State space is (still) large
3. Recent progress on training *robust* models with this property
4. But what about points outside the training data?

Summary



Attacks exist at each stage of the pipeline

- ML techniques need assumptions to perform well
- When assumptions don't hold, behavior is often surprising
- Opacity of Deep Learning techniques compounds the problem
- Addressing the gap between attacker capability and needed assumptions is an active research topic

Further reading

- Dalvi et al, “Adversarial classification”. KDD 2004
- Biggio et al, “Poisoning Attacks Against Support Vector Machines”. ICML 2012
- Koh et al, “Understanding Black-Box Predictions via Influence Functions”. ICML 2017
- Szegedy et al, “Intriguing properties of neural networks”. arXiv TR, 2013
- Goodfellow et al, “Explaining and harnessing adversarial examples”. arXiv TR, 2014
- Tramèr et al, “The Space of Transferable Adversarial Examples”. arXiv TR, 2017
- Papernot et al, “Practical Black-Box Attacks against Machine Learning”. ASIACCS 2017
- Papernot et al, “The Limitations of Deep Learning in Adversarial Settings”, EuroS&P 2016
- Carlini and Wagner, “Towards Evaluating the Robustness of Neural Networks”, Oakland 2017
- Carlini and Wagner, “Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods”. AISec 2017.