High Level Computer Vision: Attacks on Computer Vision Models

Mario Fritz fritz@cispa.saarland

Bernt Schiele schiele@mpi-inf.mpg.de

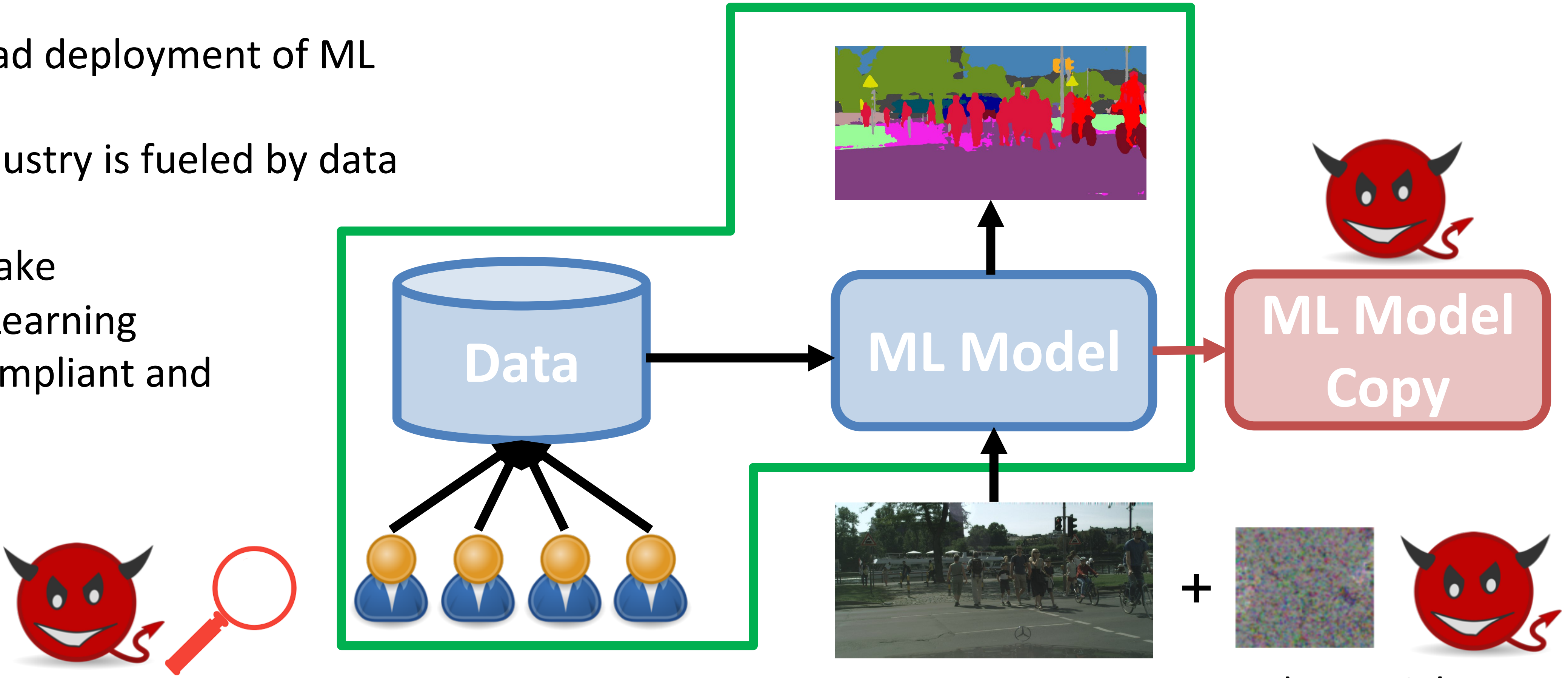
3.7.2019

Outline

- Landscape of attacks on Computer Vision Models
- Adversarial Perturbations
- Data Poisoning
- Membership Inference
- Reverse Engineering and Model Stealing
- Watermarking

Privacy & Security in Machine Learning: Towards Trustworthy AI

- Widespread deployment of ML
- Future industry is fueled by data
- How to make Machine Learning privacy compliant and secure?



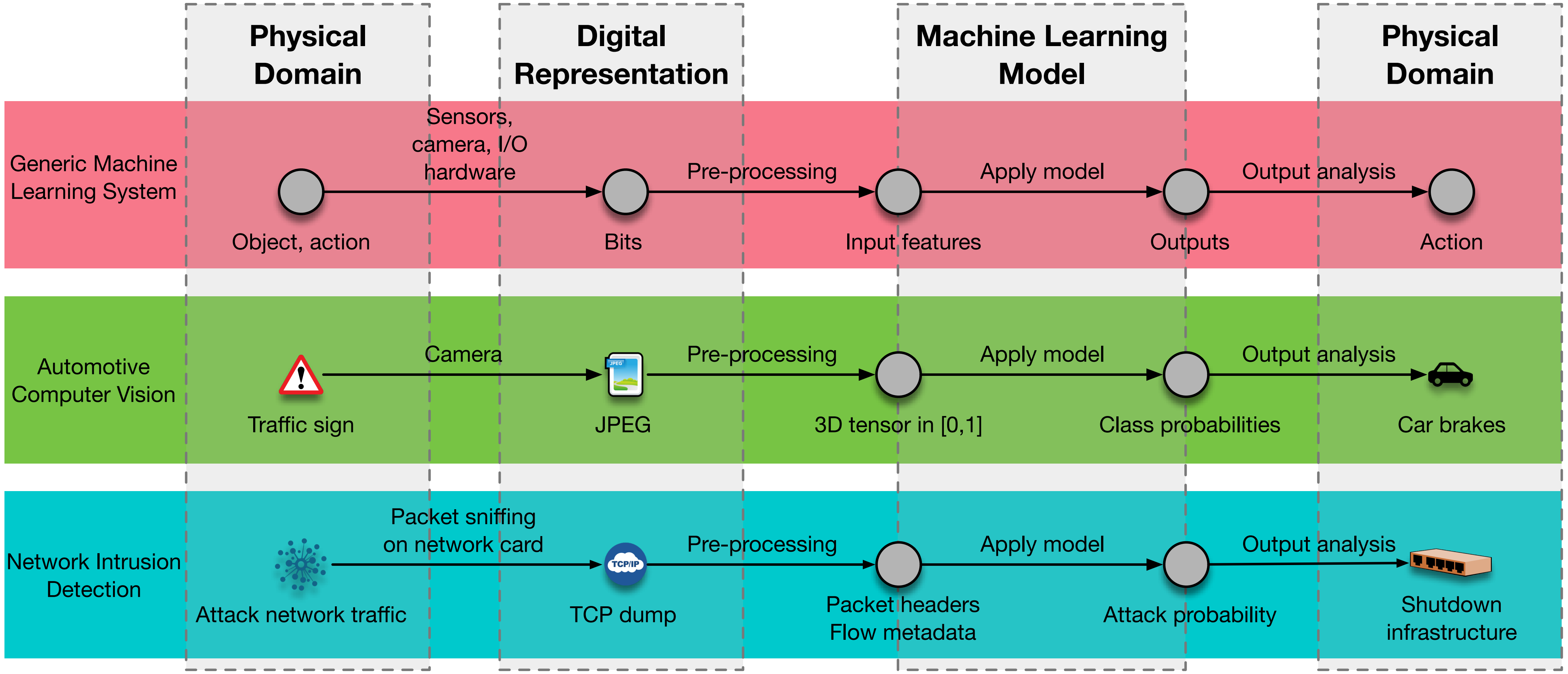
- Membership Inference
- Data Poisoning

Adversarial Perturbations

S. Oh; M. Augustin; B. Schiele; M. Fritz; Towards Reverse-Engineering Black-Box Neural Networks; **ICLR'18**
S. Oh; M. Fritz; B.Schiele; Adversarial Image Perturbation for Privacy Protection -- A Game Theory Perspective **ICCV'17**
A. Salem; Y. Zhang; M. Humbert; M. Fritz; M. Backes; ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models **NDSS'19**

K.Grosse, N. Papernot, P.Manoharan, M. Backes, P. D. McDaniel: Adversarial Examples for Malware Detection. **ESORICS'17**
L. Hanzlik; Y, Zhang; K. Grosse; A. Salem; M. Augustin; M. Backes; M.Fritz; MLCapsule: Guarded Offline Deployment of Machine Learning as a Service; **ArXiv'18**
Tribhuvanesh Orekondy; Bernt Schiele; Mario Fritz; Knockoff Nets: Stealing Functionality of Black-Box Models **CVPR'19**

Machine Learning Systems' attack surface



Papernot'16: SoK: Towards the Science of Security and Privacy in Machine Learning

Goals: Confidentiality & Privacy

- Membership Inference Attacks
  - Trying to infer information on the training data
  - Only observing input/output
  - High capacity models partially memorize the training data

- Model Inference Attacks
  - Trying to infer information about the model
  - Only observing input/output

Goals: Integrity & Availability

- Reduce
  - Quality of model (confidence or consistency)
  - Performance (speed)
  - Access (denial of service)

- Manipulating
  - Training data: Poisoning Attacks
  - Test data: Evasion Attacks

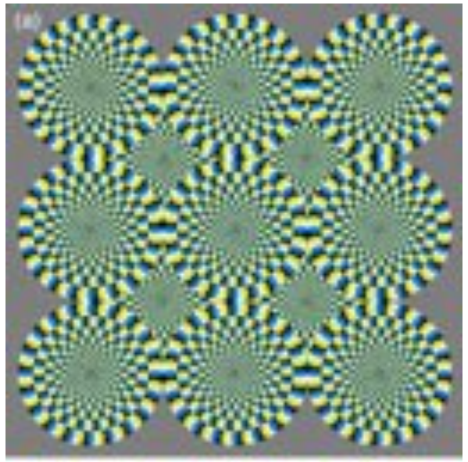
Evasion Attacks

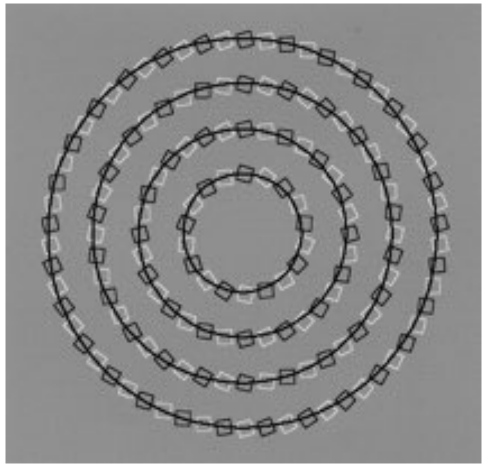
Robustness of Machine Learning – Difficult to test in real-world (i.i.d, tails of the distribution, corner cases, rare examples)

Real World Data

Robustness of Machine Learning – Difficult to test in real-world (i.i.d, tails)

Human crafted/manipulated data

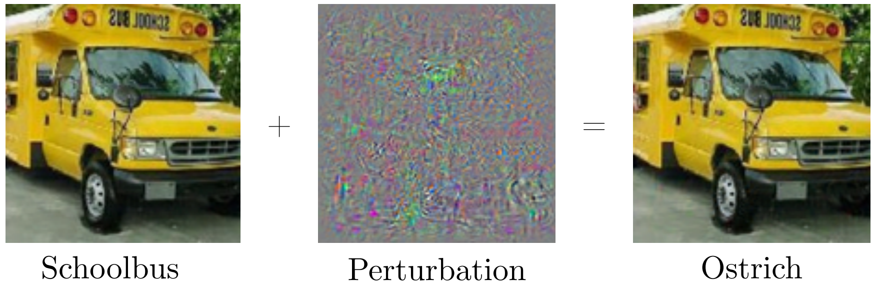




Adversarial Manipulation of Input

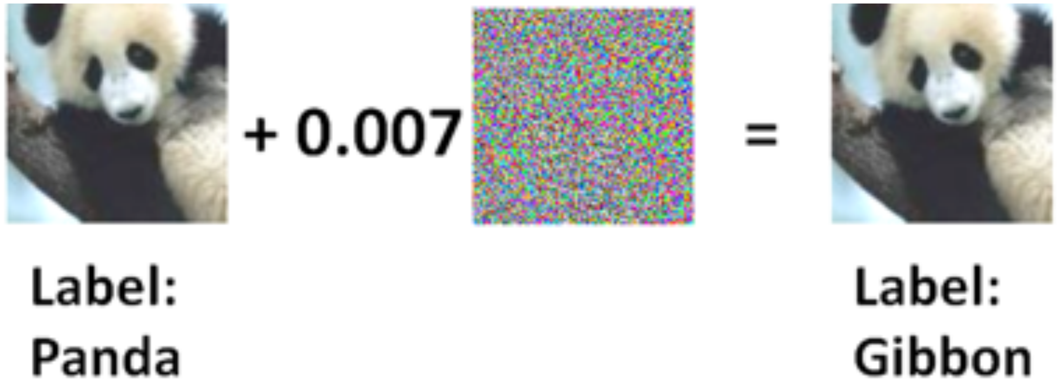
Machine crafted/manipulated data

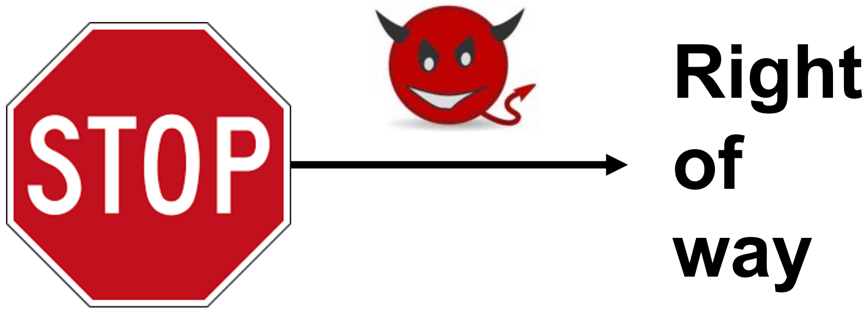
“Adversarial examples”



(rescaled for visualization)

(Szegedy et al, 2013)





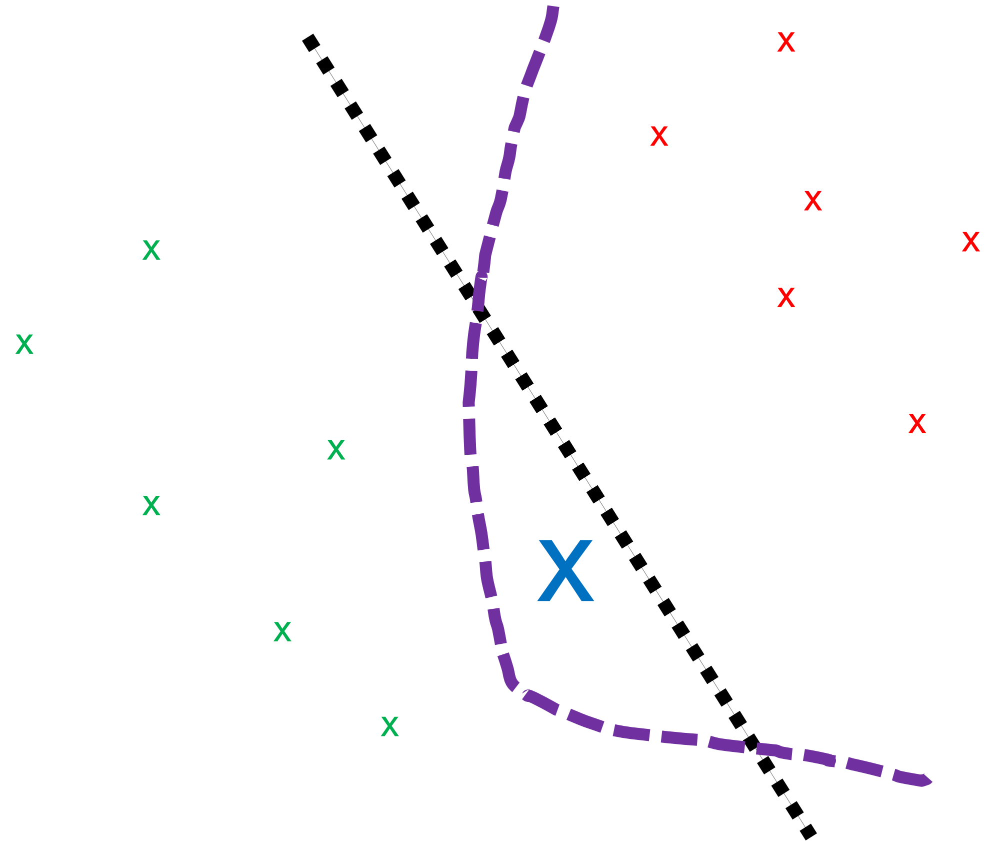
Evasion Attack

Training Data

email
email
email
email
email

email
email
email
email
email

Feature Space



Test Data

email

- Correct classification as non-SPAM
- Misclassification as non-SPAM (true boundary)

Evasion Attack

Training Data

email

email

email

email

email

email

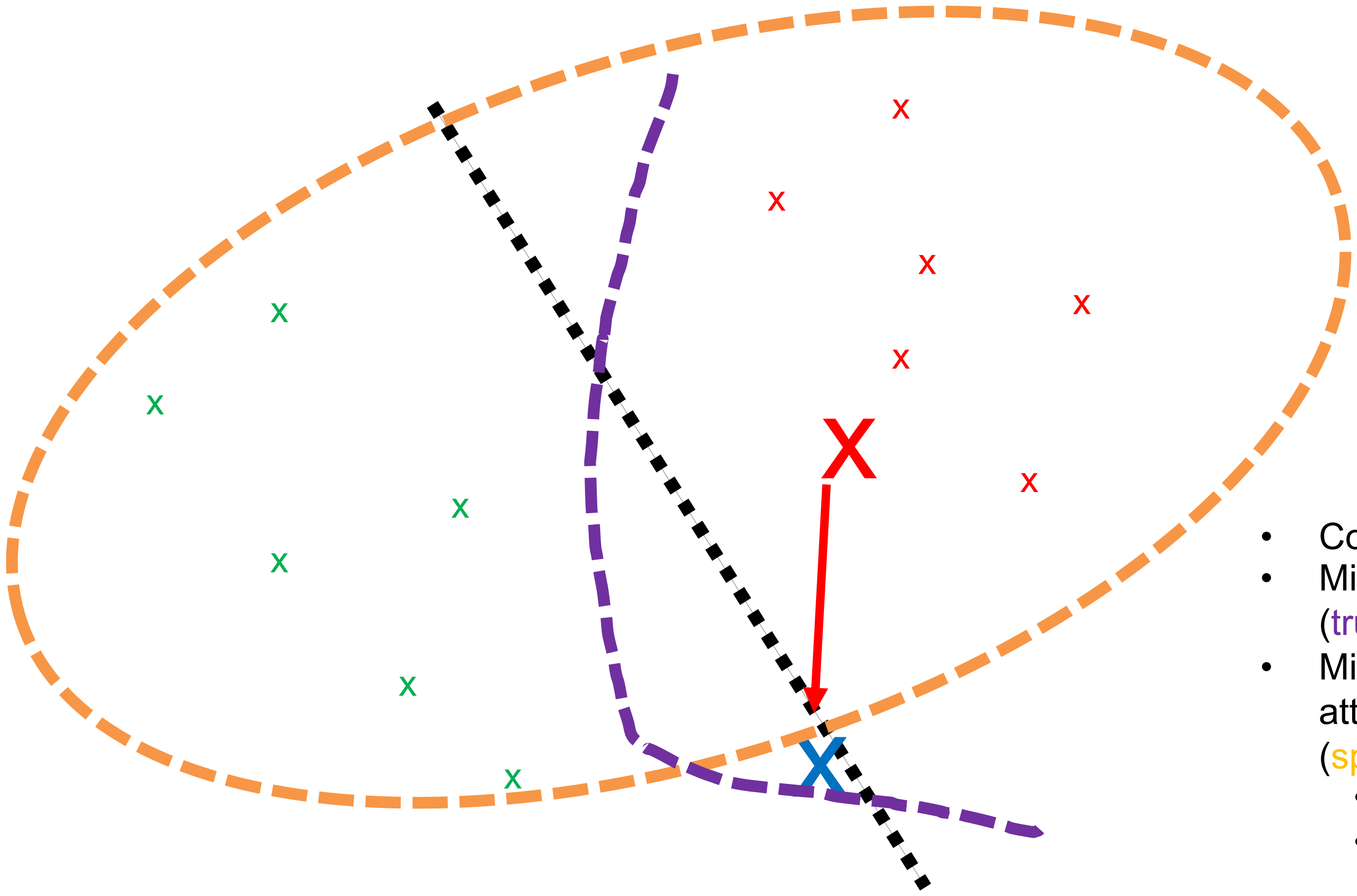
email

email

email

email

Feature Space



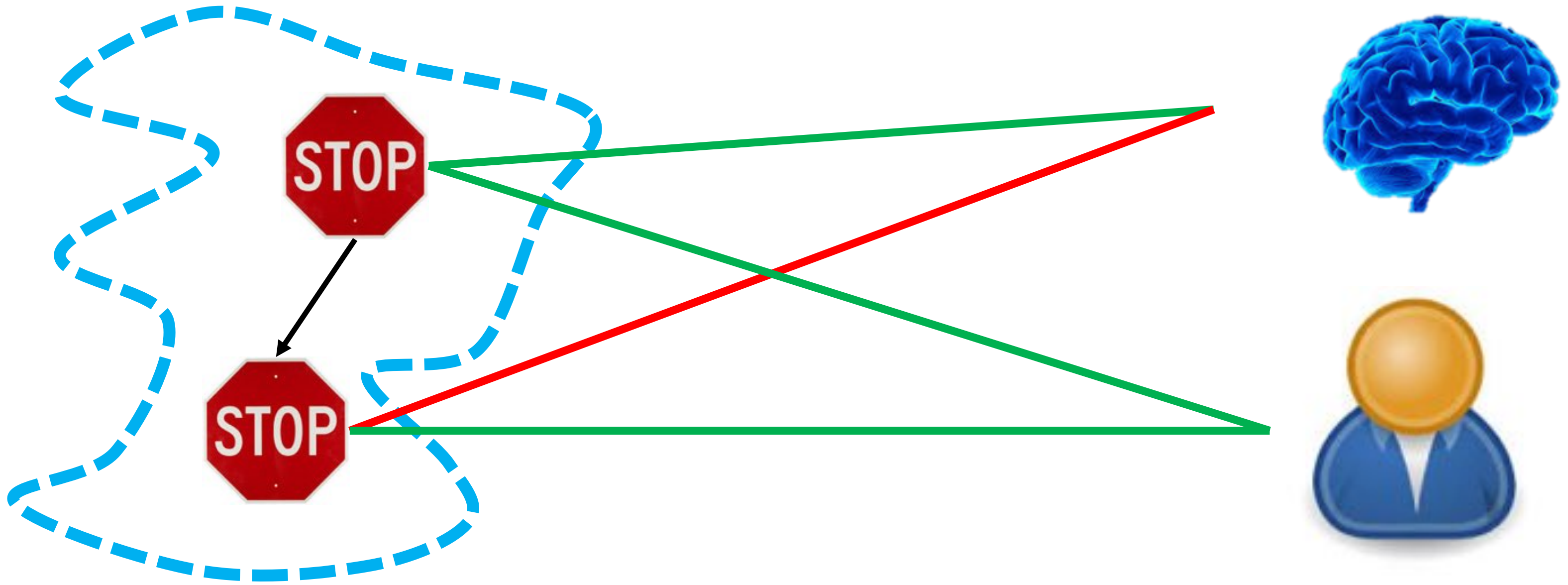
Test Data

email

- Correct classification as non-SPAM
- Misclassification as non-SPAM (true boundary)
- Misclassification due to evasion attack (space of proper emails)
  - Modeling error
  - Out of sample

Evasion Attacks

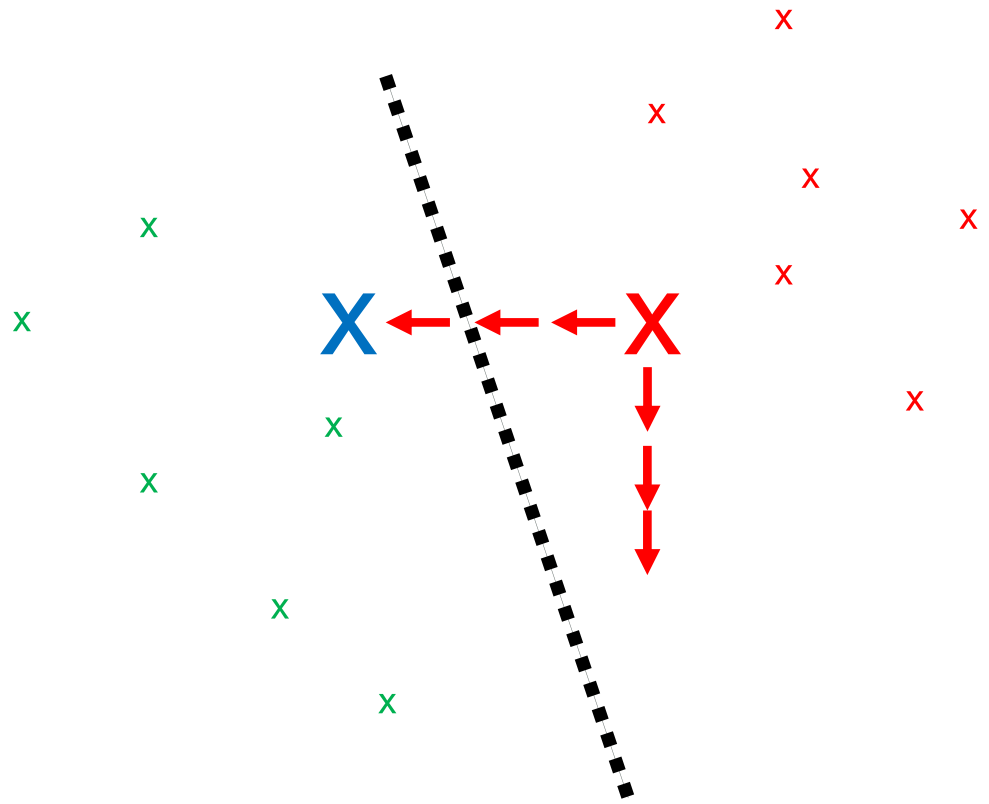
- Has been shown to work for all kind of input data:
  - SPAM, malware, traffic signs, …
- We require some notion that the change is small “small"
  - E.G. L0 (how many dimensions unchanged), L2, L_infinity norm (what is the largest change)
  - Perceptual and domain specific norms are topic of research



Binary Classifier Evasion Attack

- Linear classifier / logistic regression
- Find direction with strongest change
  - Dimension with highest weight
- Move axis parallel until label changes

$$f(w,x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$



Adversarial Machine Learning: Evasion attacks

- Empirical Risk Minimization

$$\min_{\theta} \rho(\theta), \qquad \text{where} \qquad \rho(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[L(x, y, \theta)]$$

- Empirical Risk Minimization in adversarial conditions:
  - Saddle point: $$\min_{\theta} \rho(\theta), \qquad \text{where} \qquad \rho(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\max_{\delta\in\mathcal{S}} L(\theta, x+\delta, y)\right]$$
  - Inner maximization finds adversarial versions with high loss
  - Outer minimization tries to find parameters so that “adversarial loss” of inner attack is minimized

Evasion Attacks: Fast Signed Gradient Method

$$\min_{\theta} \rho(\theta), \qquad \text{where} \qquad \rho(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\max_{\delta\in\mathcal{S}} L(\theta, x+\delta, y)\right]$$

- How to perform inner maximization? (Attack)
- Projected Gradient descent
- One step method: Fast Gradient Sign Method (FGSM)

$$x + \varepsilon\,\mathrm{sgn}(\nabla_x L(\theta, x, y))$$

- Multi-step method: FGSM^k

$$x^{t+1} = \Pi_{x+\mathcal{S}}\left(x^t + \alpha\,\mathrm{sgn}(\nabla_x L(\theta, x, y))\right)$$

Evasion Attack: Data Gradient

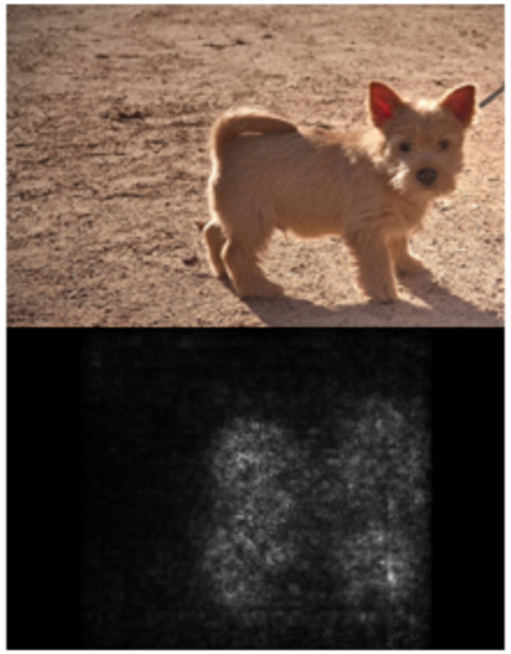
- For Training / Emperical Risk Minimization, we compute using backprop

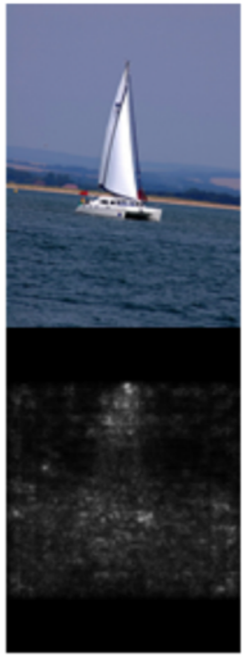
$$\nabla_\theta L(\theta, x, y)$$

- In the same manner we can use backprop to compute

$$\nabla_x L(\theta, x, y)$$

- This can also be used for interpretation:
  - How do I need to change my input to increase/decrease the loss
- However, this needs white box access in order to compute the gradient





Evasion Attack: Carlini/Wagner

$$\min_\theta \rho(\theta), \qquad \text{where} \qquad \rho(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\max_{\delta\in\mathcal{S}} L(\theta, x+\delta, y)\right]$$

- Up to now – untargeted attack: “only” increase loss
  - Targeted attack:

$$\min_\theta \rho(\theta), \qquad \text{where} \qquad \rho(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\max_{\delta\in\mathcal{S}} L_{y}(\theta, x+\delta, y)\right]$$

- Constraint optimization with Lagrange multiplier

$$\min_\theta \rho(\theta), \qquad \text{where} \qquad \rho(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\max_{\delta} L_y(\theta, x+\delta, y) - \lambda d(x, x+\delta)\right]$$

Black-Box Evasion Attacks: Transferability

- Assumption:
  - We A is not white box, B is. Use B to attack A!
  - Gradient of Loss on model A also increases loss of model B
  - Usually not the case
- Can be improved by make a guess / training a classifier to predict model family
  - Difficult
- Generate adversarial examples over ensemble

$$\min_{\theta} \rho(\theta), \qquad \text{where} \qquad \rho(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\max_{\delta\in\mathcal{S}} \underset{\underset{\displaystyle\sum_{k\in\mathcal{M}} \lambda_k \mathcal{L}_k}{\Big\uparrow}}{L}(\theta, x+\delta, y)\right]$$

Black-Box Evasion Attacks: Transferability

- Naïve approach

	RMSD	ResNet-152	ResNet-101	ResNet-50	VGG-16	GoogLeNet
ResNet-152	23.13	100%	2%	1%	1%	1%
ResNet-101	23.16	3%	100%	3%	2%	1%
ResNet-50	23.06	4%	2%	100%	1%	1%
VGG-16	23.59	2%	1%	2%	100%	1%
GoogLeNet	22.87	1%	1%	0%	1%	100%

- Use ensemble to generate adversarial examples!

	RMSD	ResNet-152	ResNet-101	ResNet-50	VGG-16	GoogLeNet
-ResNet-152	30.68	38%	76%	70%	97%	76%
-ResNet-101	30.76	75%	43%	69%	98%	73%
-ResNet-50	30.26	84%	81%	46%	99%	77%
-VGG-16	31.13	74%	78%	68%	24%	63%
-GoogLeNet	29.70	90%	87%	83%	99%	11%

Black-Box Evasion Attacks: Selective Attack

- Model specific attacks:
  - Break model A
  - Leave model B alone

$$\min_{\theta} \rho(\theta), \qquad \text{where} \qquad \rho(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\max_{\delta\in\mathcal{S}} L(\theta, x+\delta, y)\right]$$

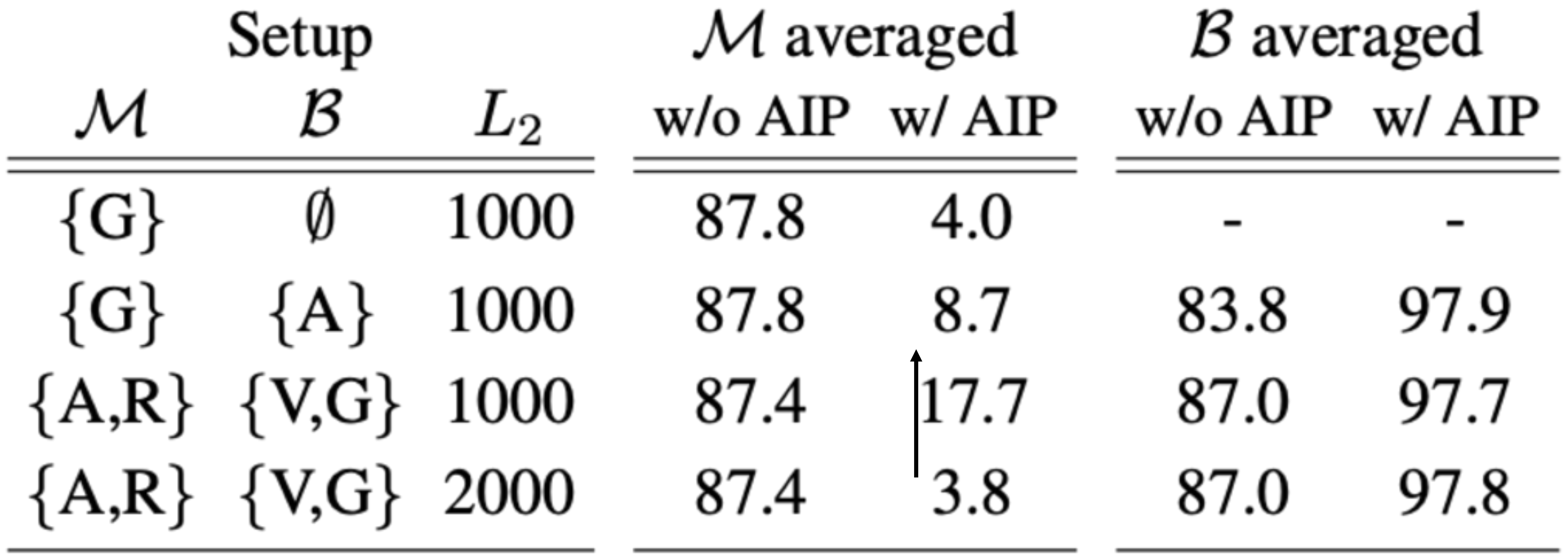
$$\uparrow$$

$$\sum_{k\in\mathcal{M}} \lambda_k \mathcal{L}_k - \sum_{k'\in\mathcal{B}} \lambda_{k'} \mathcal{L}_{k'}$$

Malicious models

Benign models

Black-Box Evasion Attacks: Selective Attack



Setup			$\mathcal{M}$ averaged		$\mathcal{B}$ averaged	
$\mathcal{M}$	$\mathcal{B}$	L_2	w/o AIP	w/ AIP	w/o AIP	w/ AIP
{G}	$\emptyset$	1000	87.8	4.0	-	-
{G}	{A}	1000	87.8	8.7	83.8	97.9
{A,R}	{V,G}	1000	87.4	17.7	87.0	97.7
{A,R}	{V,G}	2000	87.4	3.8	87.0	97.8

$$\sum_{k \in \mathcal{M}} \lambda_k \mathcal{L}_k - \sum_{k' \in \mathcal{B}} \lambda_{k'} \mathcal{L}_{k'}$$

Black Box Evasion Attack: Approximation of the Gradient

- Numerical approximation of gradient

$$\hat{g}_i := \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_i} \approx \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x} - h\mathbf{e}_i)}{2h}$$

- Stochastic coordinate descent

Algorithm 1 Stochastic Coordinate Descent

```
1: while not converged do
2:     Randomly pick a coordinate $i \in \{1, \dots, p\}$
3:     Compute an update $\delta^*$ by approximately minimizing
           $\arg\min_{\delta} f(\mathbf{x} + \delta\mathbf{e}_i)$
4:     Update $\mathbf{x}_i \leftarrow \mathbf{x}_i + \delta^*$
5: end while
```

Defenses against evasion attacks

- Ensembles
- Dimensionality Reduction PCA or auto-encoder
- ”Denoising”
- Transformations (feature squeezing, noise, jpeg, crop)
- Detection

- … unfortunately non of these really work …
- In most cases – including the defense in the attack – there is no strong effectg

- Large body of work – limited progress so far

Topology of Evasion Attacks

- Notion of small perturbation / similar input
  - L0, L2, Linf
- One step vs multi-step method
- Projected Gradient vs Largange Optimiztion
- White box vs. black box
- Black box:
  - Transferability attacks
  - Numerical gradient
- Defenses
  - Adversarial Training

Evasion Attacks Defenses: Adversarial Training

- Adversarial Training:
  - Minimize for a maximizer of inner optimization
  - Iterate

$$\min_{\theta} \rho(\theta), \quad \text{where} \quad \rho(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\max_{\delta\in\mathcal{S}} L(\theta, x+\delta, y)\right]$$

Maximize (attack)

Minimize (defend)

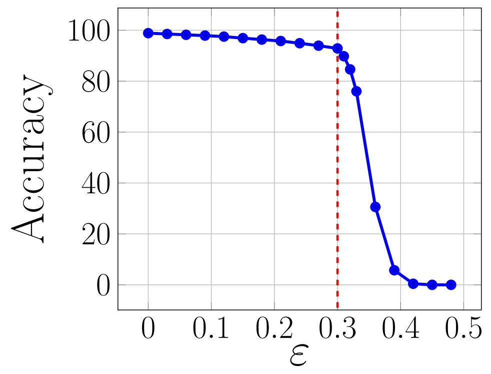
Maximize (attack)

Minimize (defend)

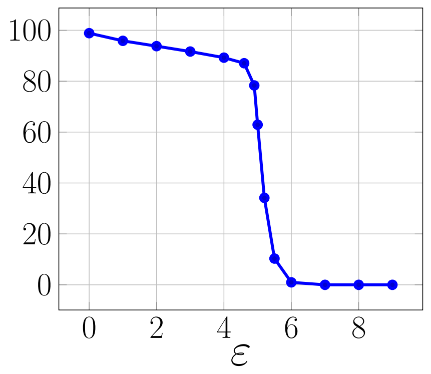
...

...

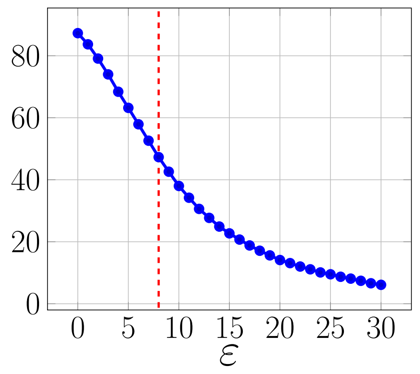
Evasion Attacks Defenses: Adversarial Training



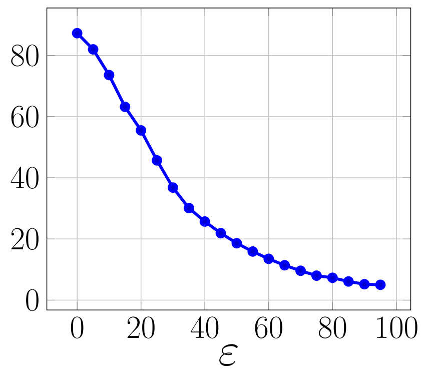
(a) MNIST, ℓ_∞ norm



(b) MNIST, ℓ_2 norm

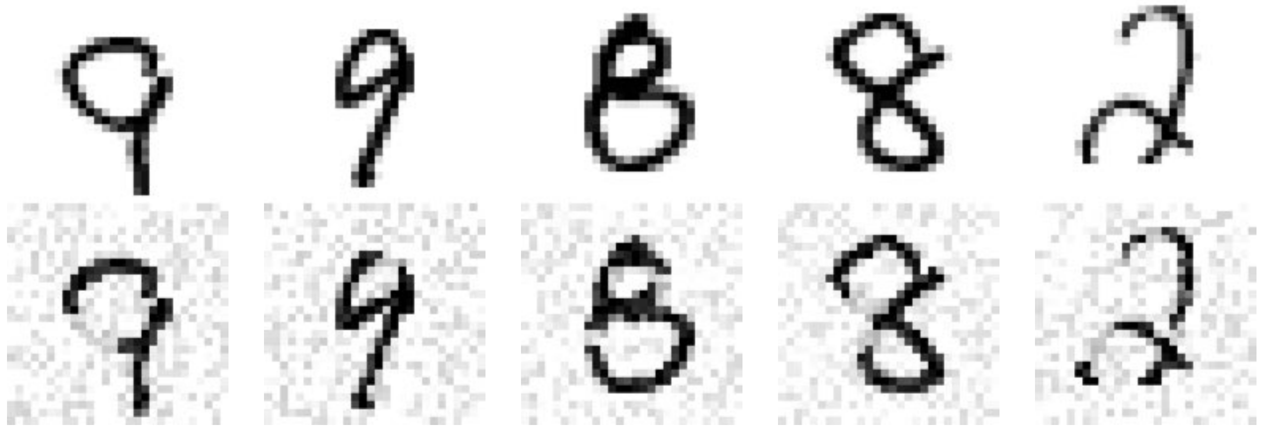


(c) CIFAR10, ℓ_∞ norm



(d) CIFAR10, ℓ_2 norm

- For datasets with lots of training data:
  - Robustness for small norm balls
- No defenses for bigger norm balls
- At some point – semantic shifts



Natural: 9	Natural: 9	Natural: 8	Natural: 8	Natural: 2
Adversarial: 7	Adversarial: 4	Adversarial: 5	Adversarial: 3	Adversarial: 3

Conclusion so far

- We are still in a cat and mouse game

- Small norm balls can be defended (only perturbations with semantic shifts can be found)
- Perturbations are still a problem in large norm balls
- Theoretical guarantees have only been shown for small networks / simplified problems
- Choice of norm is unclear / task dependent. L0-Linf norm is convenience than motivated choice
- Maybe Bayes Deep Learning, Gaussian Processes can provide a solution …

https://github.com/tensorflow/cleverhans

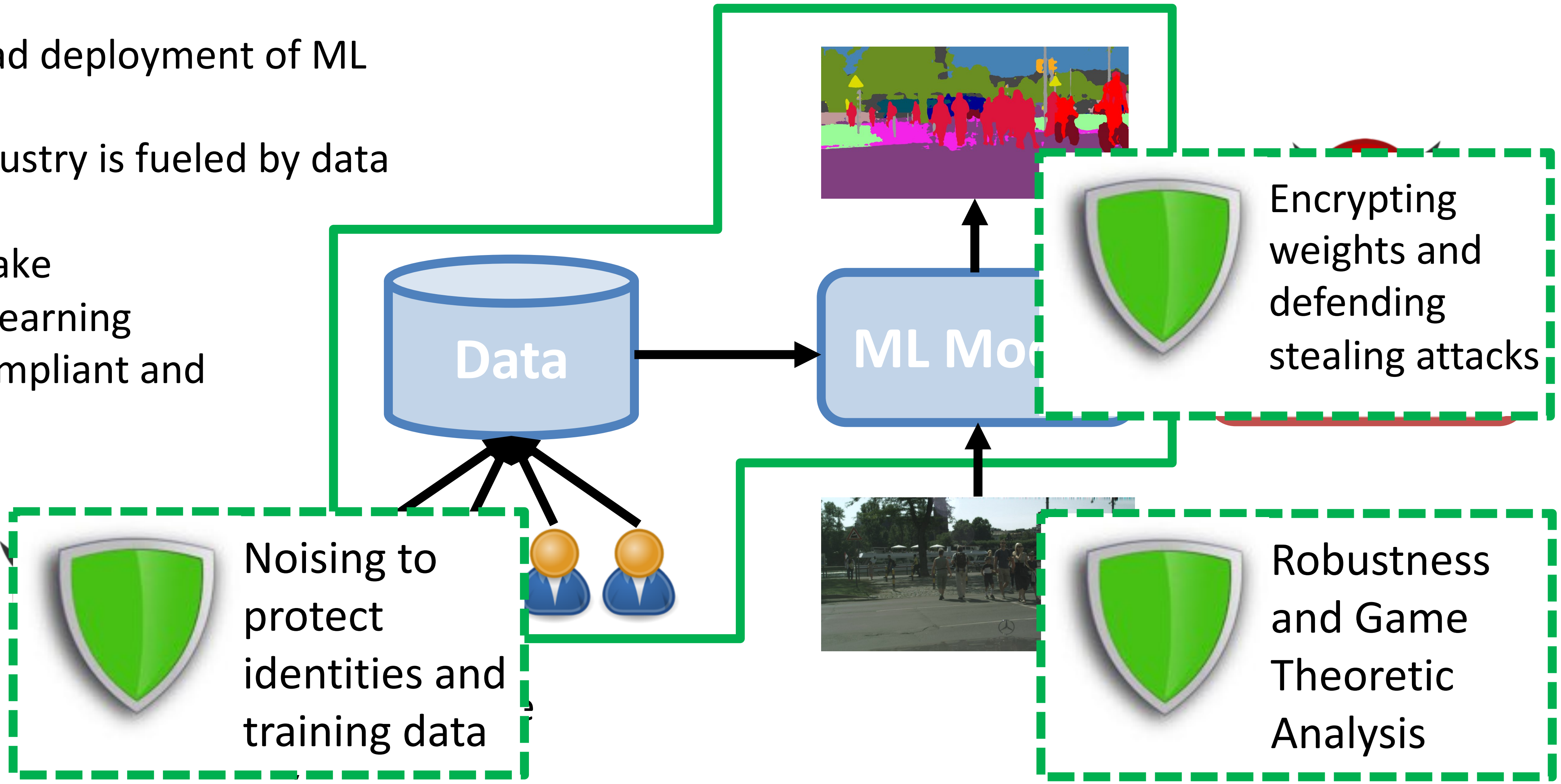
Advanced Attacks on AI/ML: Reverse Engineering and Model Stealing

Seong Joon Oh; Max Augustin; Bernt Schiele; Mario Fritz
Towards Reverse-Engineering Black-Box Neural Networks Inproceedings
ICLR'18

Tribhuvanesh Orekondy; Bernt Schiele; Mario Fritz
Knockoff Nets: Stealing Functionality of Black-Box Models
CVPR'19

Privacy & Security in Machine Learning

- Widespread deployment of ML
- Future industry is fueled by data
- How to make Machine Learning privacy compliant and secure?


Data
ML Mo
Encrypting weights and defending stealing attacks
Noising to protect identities and training data
Robustness and Game Theoretic Analysis

S. Oh; M. Augustin; B. Schiele; M. Fritz; Towards Reverse-Engineering Black-Box Neural Networks; **ICLR'18**
S. Oh; M. Fritz; B.Schiele; Adversarial Image Perturbation for Privacy Protection -- A Game Theory Perspective **ICCV'17**
A. Salem; Y. Zhang; M. Humbert; M. Fritz; M. Backes; ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models **ArXiv'18**

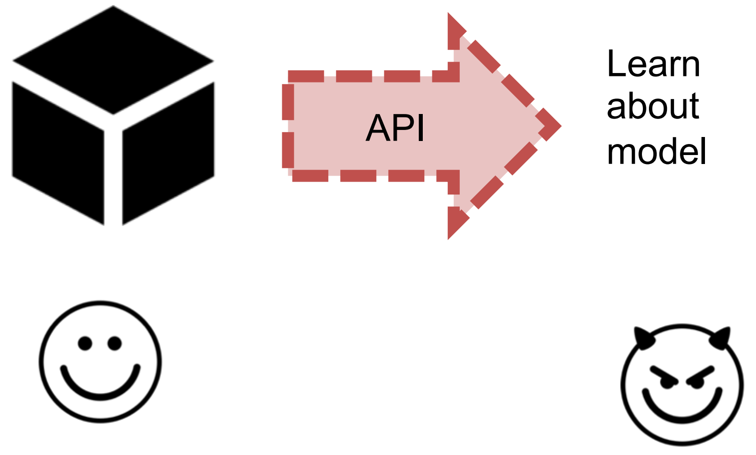
K.Grosse, N. Papernot, P.Manoharan, M. Backes, P. D. McDaniel: Adversarial Examples for Malware Detection. **ESORICS'17**
L. Hanzlik; Y, Zhang; K. Grosse; A. Salem; M. Augustin; M. Backes; M.Fritz; MLCapsule: Guarded Offline Deployment of Machine Learning as a Service; **ArXiv'18**

Reverse Engineering & Model Stealing: Problem & Motivation

- Many deployed models are black boxes, APIs (given input, returns output).

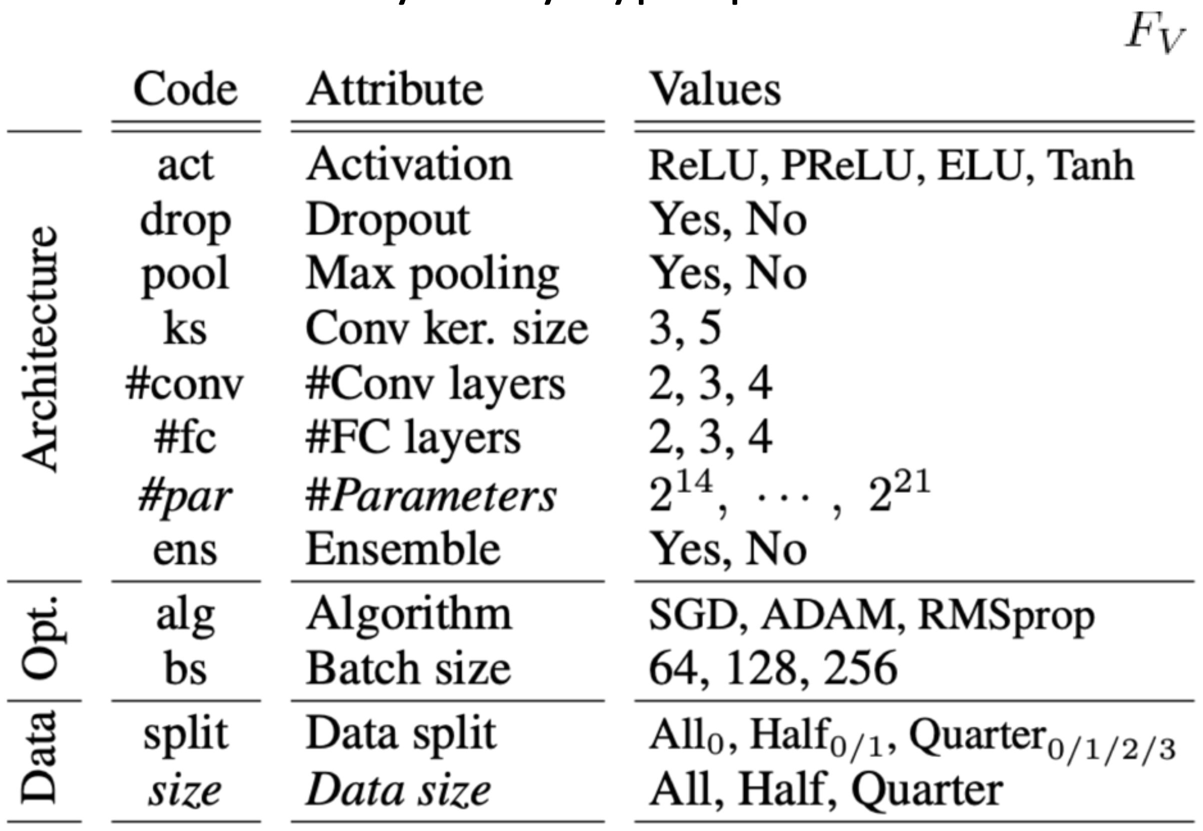
- Can black-box accesses reveal model internals? e.g.
  - Architecture
  - training procedure
  - Data
  - Functionality

- Why does it matter? Key intellectual property, monetization and increased vulnerability to other attacks.



Reverse Engineering Neural Networks (ICLR’18)

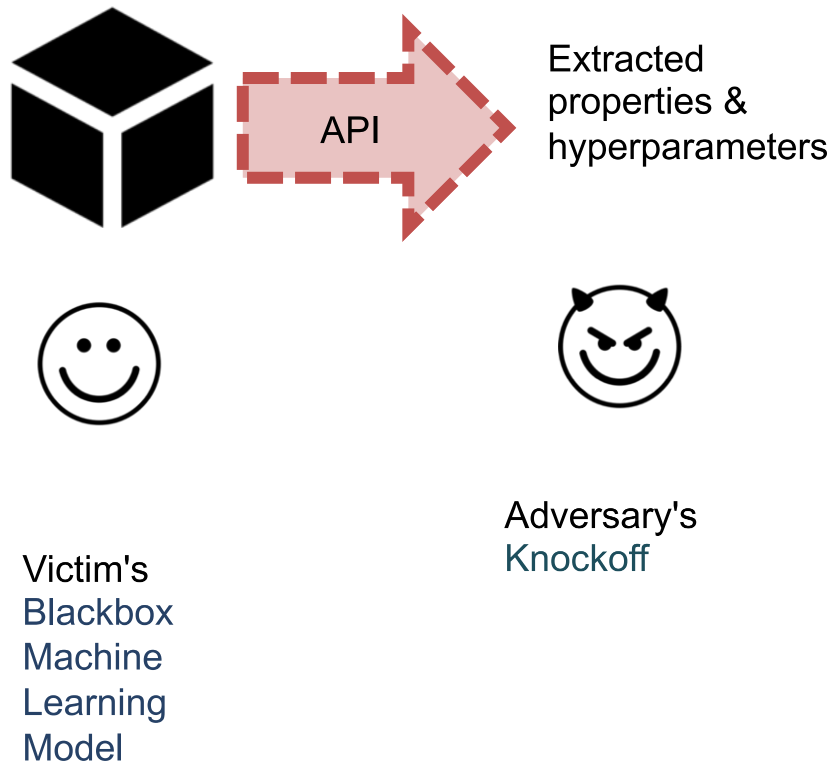
- State of the art deep learning architectures are defined by many hyper parameters



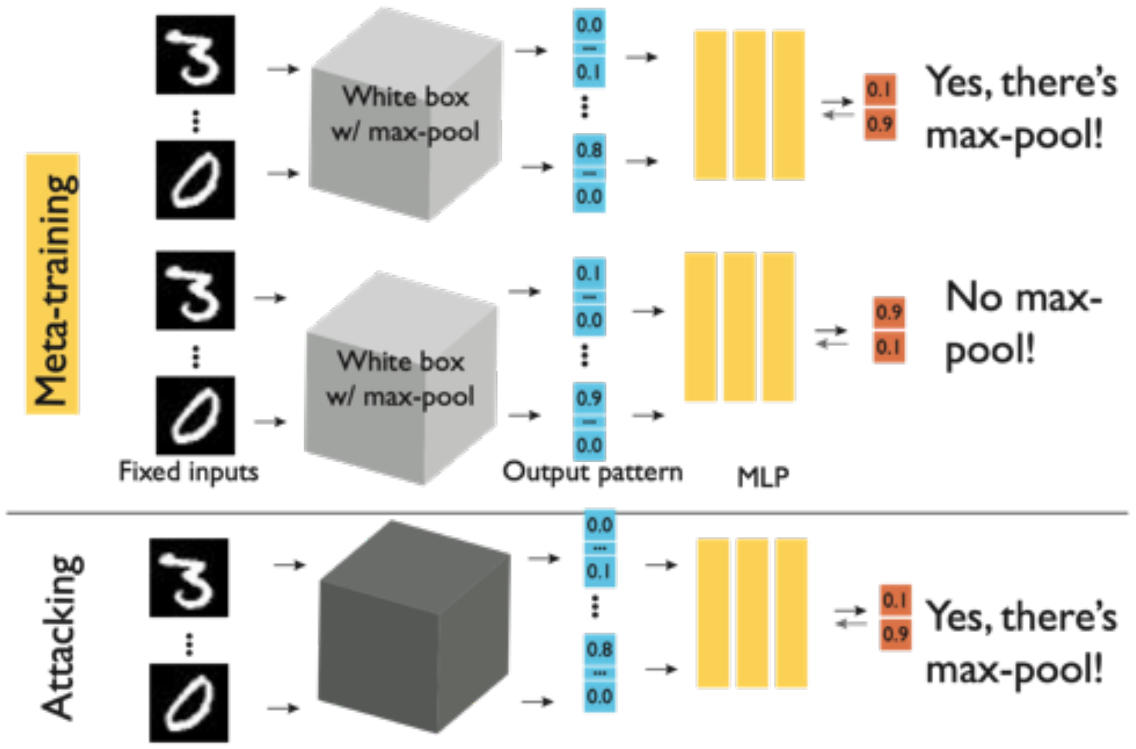
F_V

	Code	Attribute	Values
Architecture	act	Activation	ReLU, PReLU, ELU, Tanh
	drop	Dropout	Yes, No
	pool	Max pooling	Yes, No
	ks	Conv ker. size	3, 5
	#conv	#Conv layers	2, 3, 4
	#fc	#FC layers	2, 3, 4
	#par	*#Parameters*	$2^{14}, \cdots, 2^{21}$
	ens	Ensemble	Yes, No
Opt.	alg	Algorithm	SGD, ADAM, RMSprop
	bs	Batch size	64, 128, 256
Data	split	Data split	$\text{All}_0, \text{Half}_{0/1}, \text{Quarter}_{0/1/2/3}$
	size	*Data size*	All, Half, Quarter

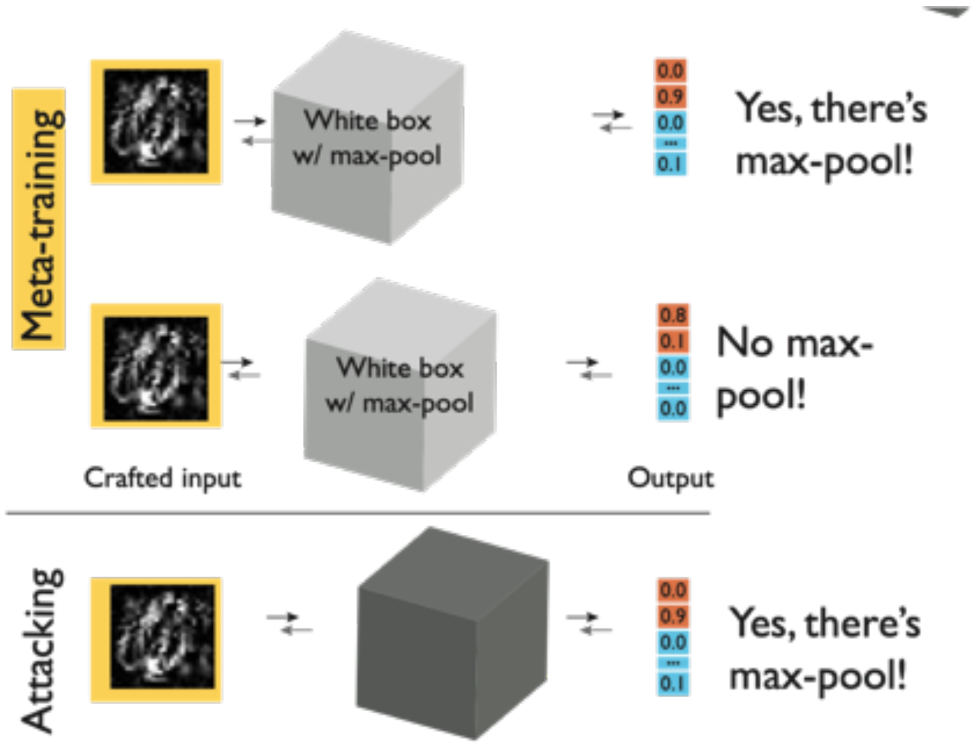
- Can those be inferred from black box access?


API
Extracted properties & hyperparameters
Victim's Blackbox Machine Learning Model
Adversary's Knockoff

Reverse Engineering Neural Networks (ICLR’18)



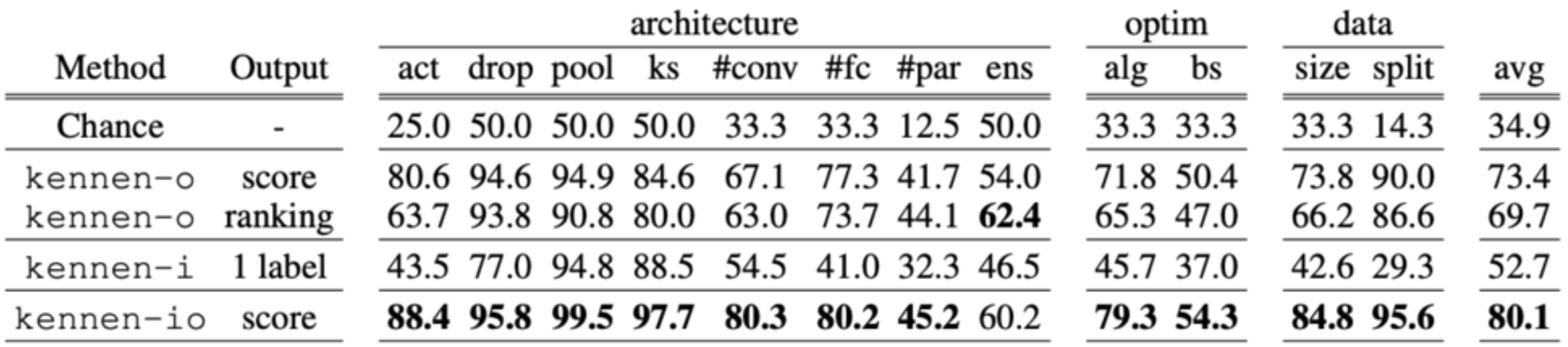
Method 1. kennen-o : Learn to read-off the existence of max-pool from the output pattern.



Method 2. kennen-i : Craft a single “adversarial” input that looks like “1” with a max-pool layer and “0” without.

Method 3. kennen-io: attribute prediction + input crafting

Reverse Engineering Neural Networks (ICLR'18)



		architecture								optim		data		
Method	Output	act	drop	pool	ks	#conv	#fc	#par	ens	alg	bs	size	split	avg
Chance	-	25.0	50.0	50.0	50.0	33.3	33.3	12.5	50.0	33.3	33.3	33.3	14.3	34.9
kennen-o	score	80.6	94.6	94.9	84.6	67.1	77.3	41.7	54.0	71.8	50.4	73.8	90.0	73.4
kennen-o	ranking	63.7	93.8	90.8	80.0	63.0	73.7	44.1	**62.4**	65.3	47.0	66.2	86.6	69.7
kennen-i	1 label	43.5	77.0	94.8	88.5	54.5	41.0	32.3	46.5	45.7	37.0	42.6	29.3	52.7
kennen-io	score	**88.4**	**95.8**	**99.5**	**97.7**	**80.3**	**80.2**	**45.2**	60.2	**79.3**	**54.3**	**84.8**	**95.6**	**80.1**

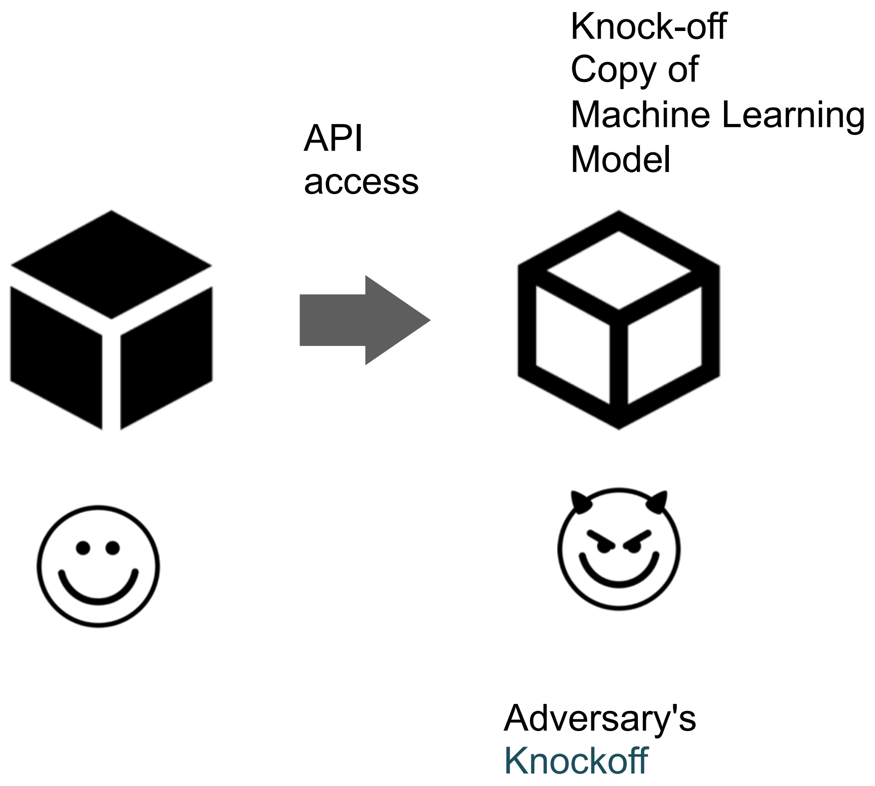
… but does adversary really want to know all those details to steal or attack a model?

Functionality Stealing / Knock-off Nets (CVPR'19)

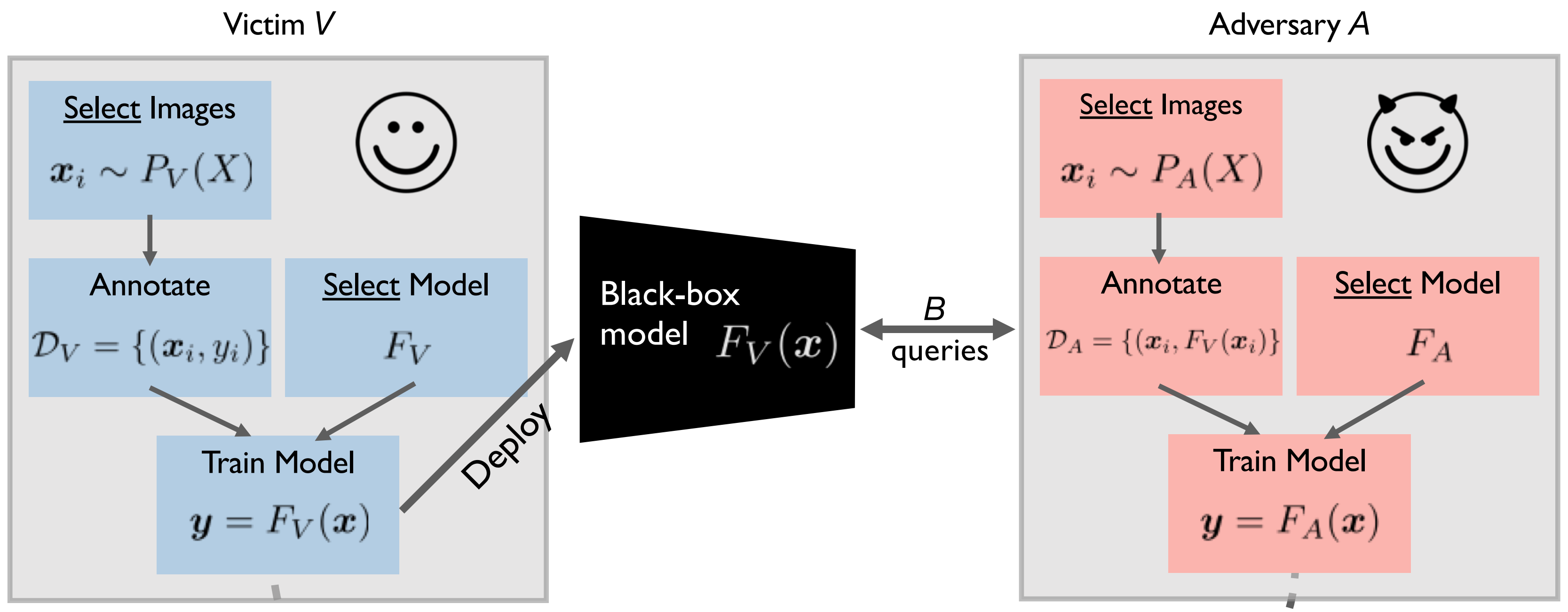
- Functionality stealing generates copy
- Copy might differ internally – should be indistinguishable from the outside

- Facilitates stronger attacks
- Threat to intellectual property and monetization models

- What does adversary need to know?
  - Model (does not matter much)
  - Data (does not matter much)
- What about defenses?



Functionality Stealing: Knock-Off Nets (CVPR'19)



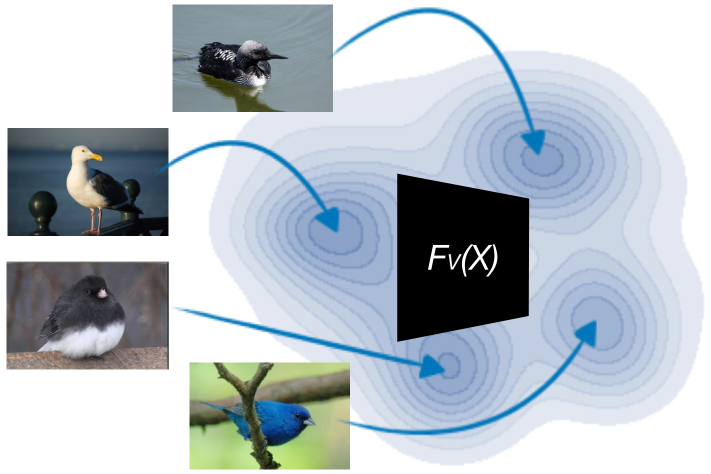
Can *A* approximate *F_V*:

1. when P_V and F_V are unknown?
2. using minimum queries *B*?

Resembles Model Distillation … but under weaker assumptions

Query Set Selection: Challenge

$$P_V(X)$$



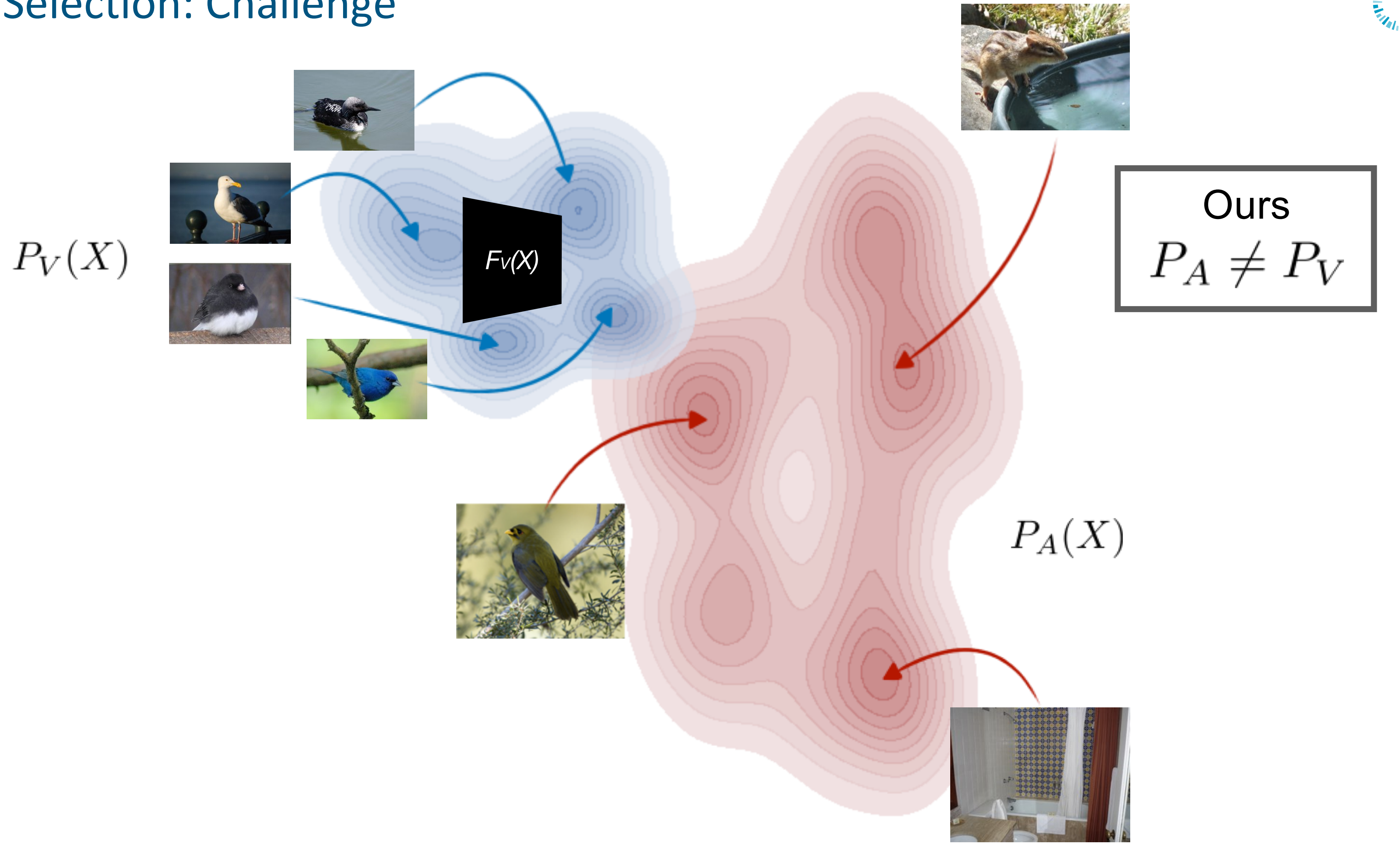
Active Learning

Distillation

Student-Teacher

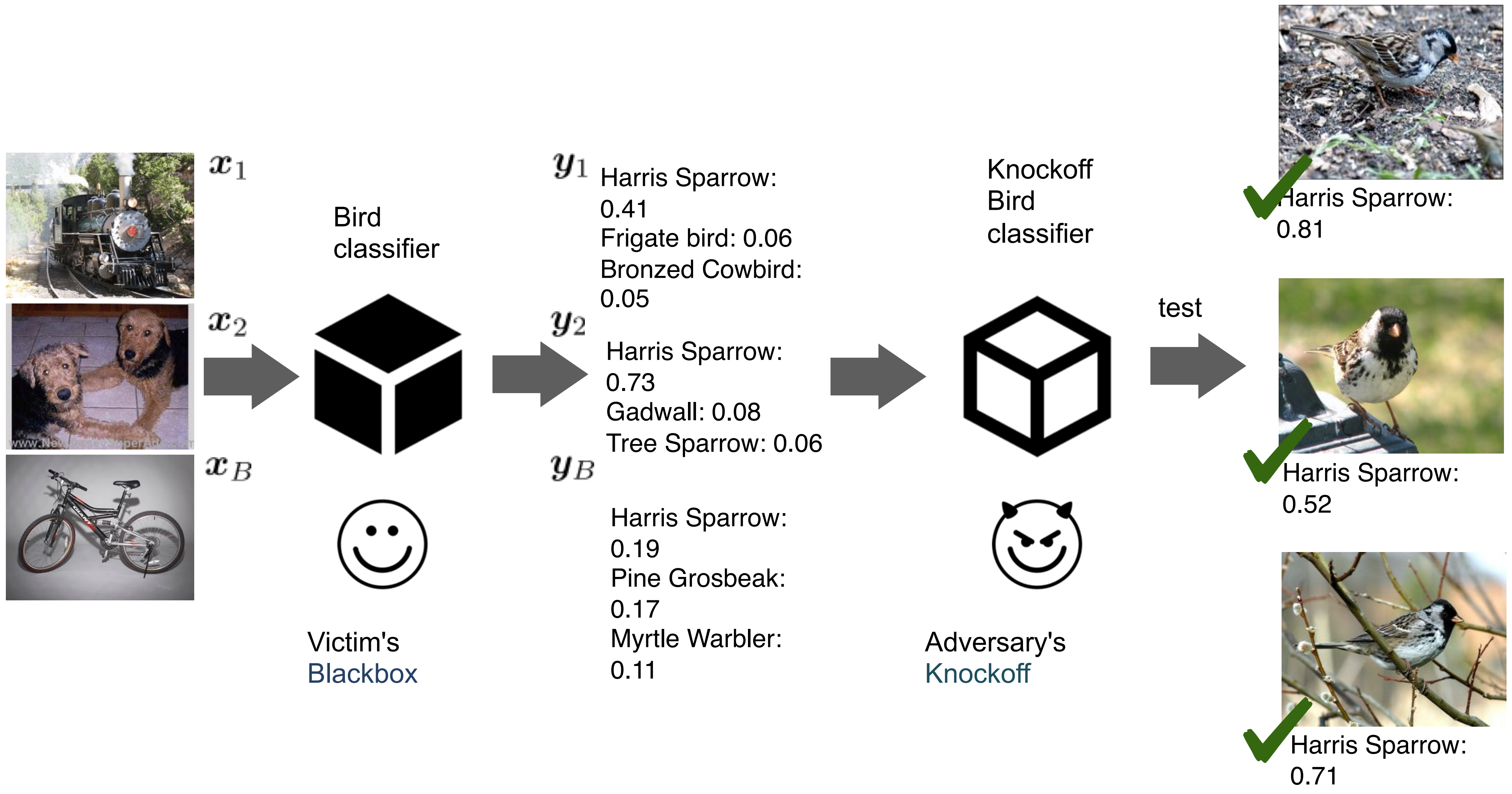
$$P_V = P_A$$

Query Set Selection: Challenge



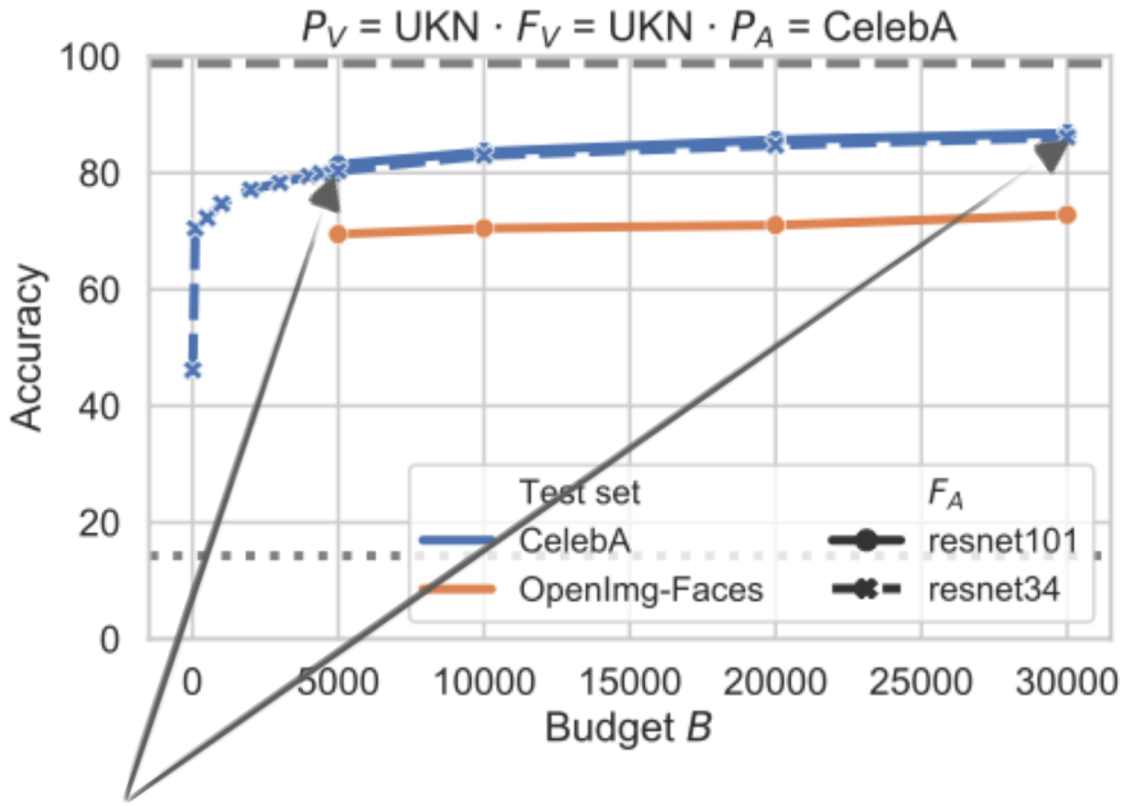
- Improved query efficiency by Reinforcement Learning

Functionality Stealing: Knock-Off Nets (CVPR’19)



Results on Real-World Attacks

Train: CelebA
Test: CelebA, OpenImg-Faces



80% accuracy @ B=5k ($0)
86% @ B=30k ($30)

- Strong copy from a few 1000 queries
- Unfortunately difficult to defend
  - Noising
  - Top-k, argmax
  - Rounding
  - Watermarking only post-hoc attribution
  - MLCapsule – SGX-based deployment