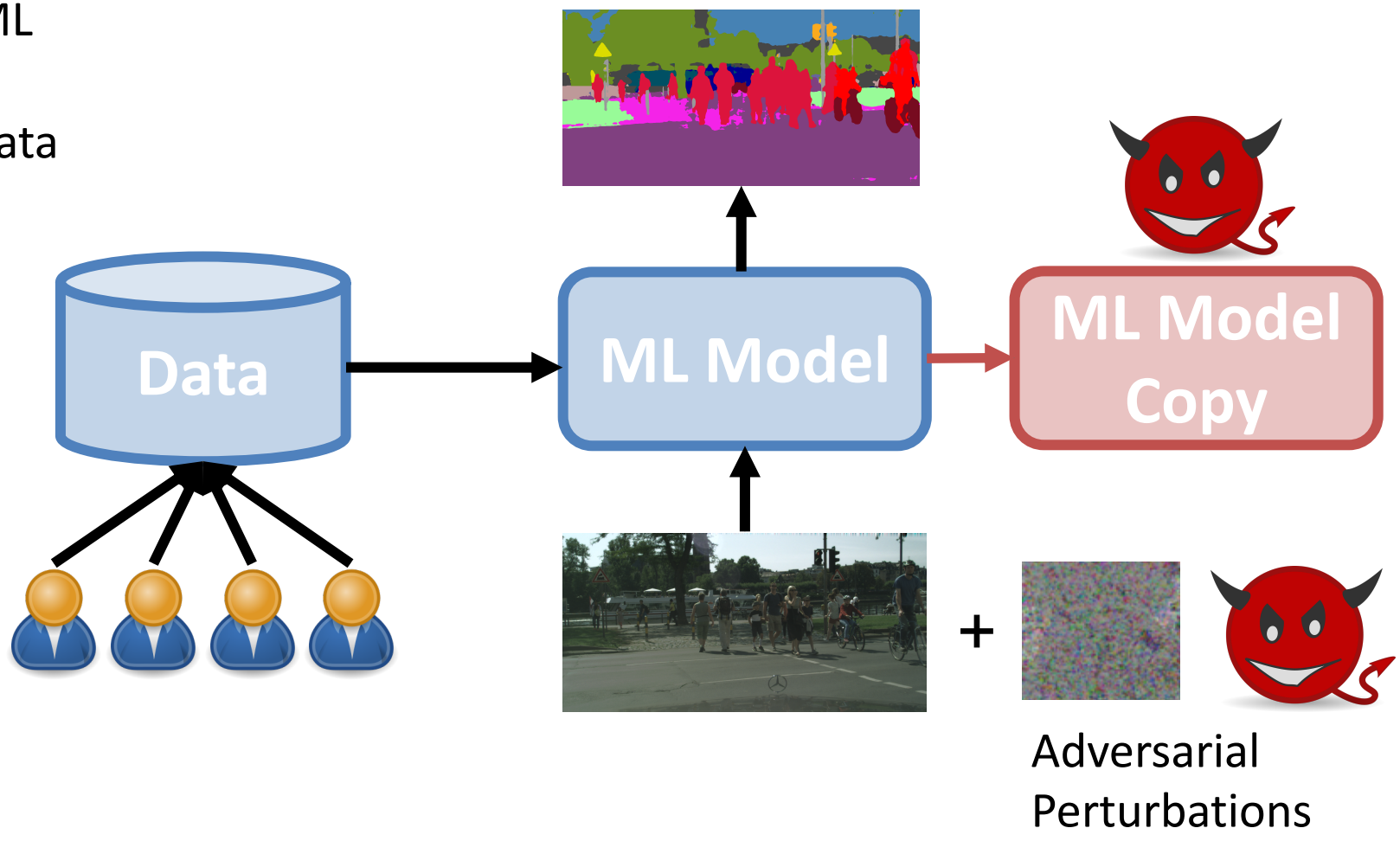
Machine Learning in Cybersecurity

- Model Stealing & Defense
- Watermarking

Prof. Dr. Mario Fritz | 10.12.2020

Privacy & Security in Machine Learning

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



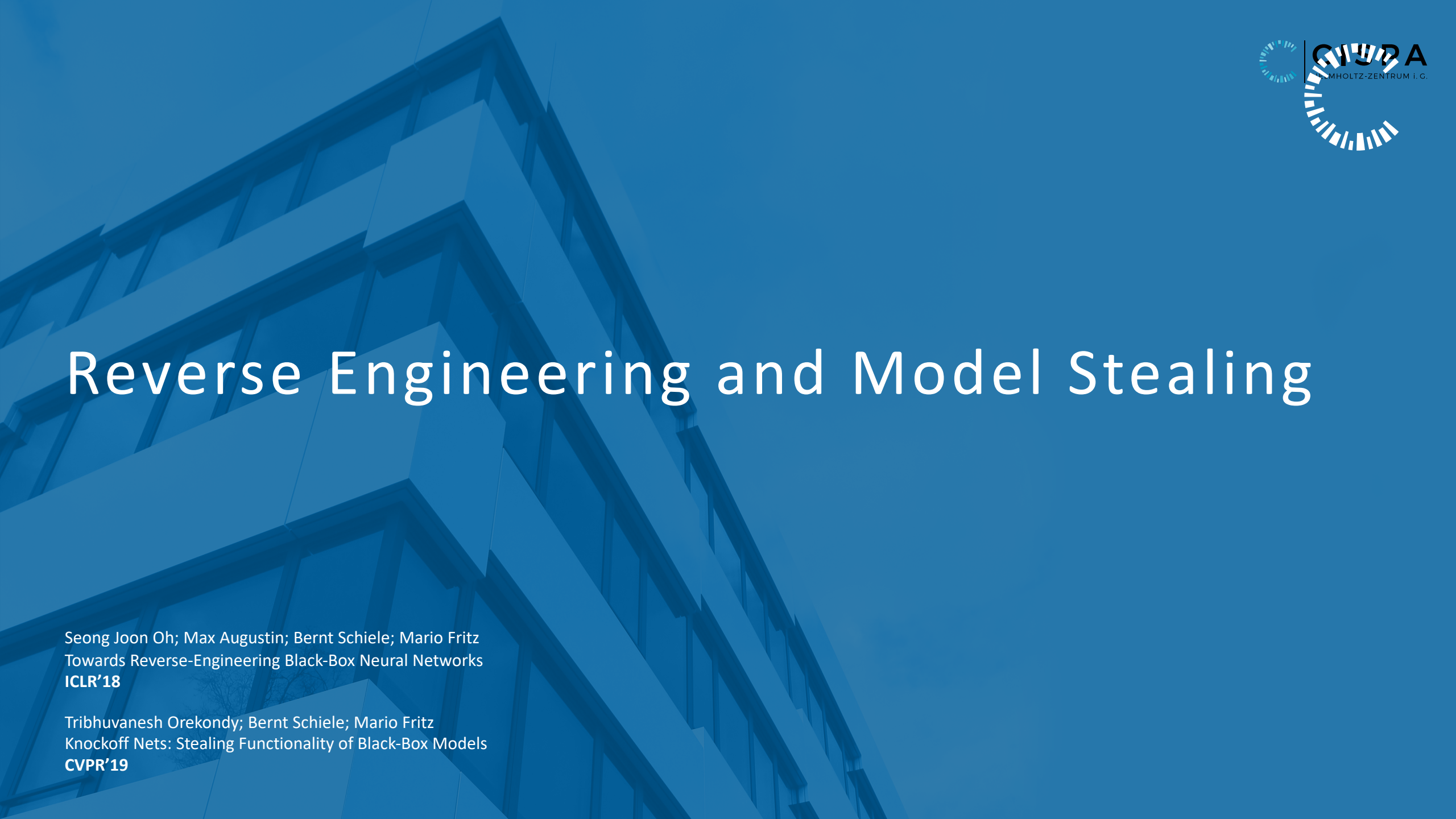
S. Oh; M. Augustin; B. Schiele; M. Fritz; Towards Reverse-Engineering Black-Box Neural Networks; **ICLR'18**

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**



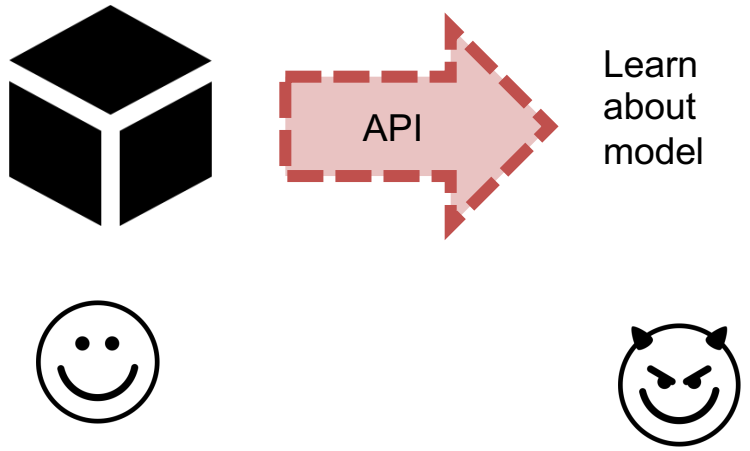
Reverse Engineering and Model Stealing

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

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

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.



- Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, Thomas Ristenpart: Stealing Machine Learning Models via Prediction APIs. USENIX Security Symposium 2016

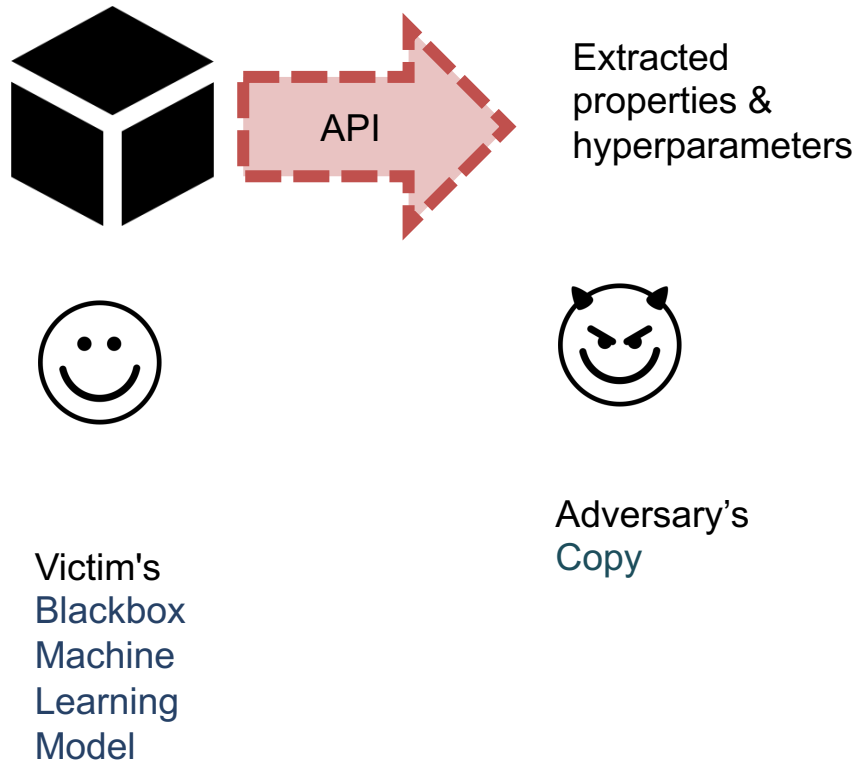
Reverse Engineering Neural Networks (ICLR'18)

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

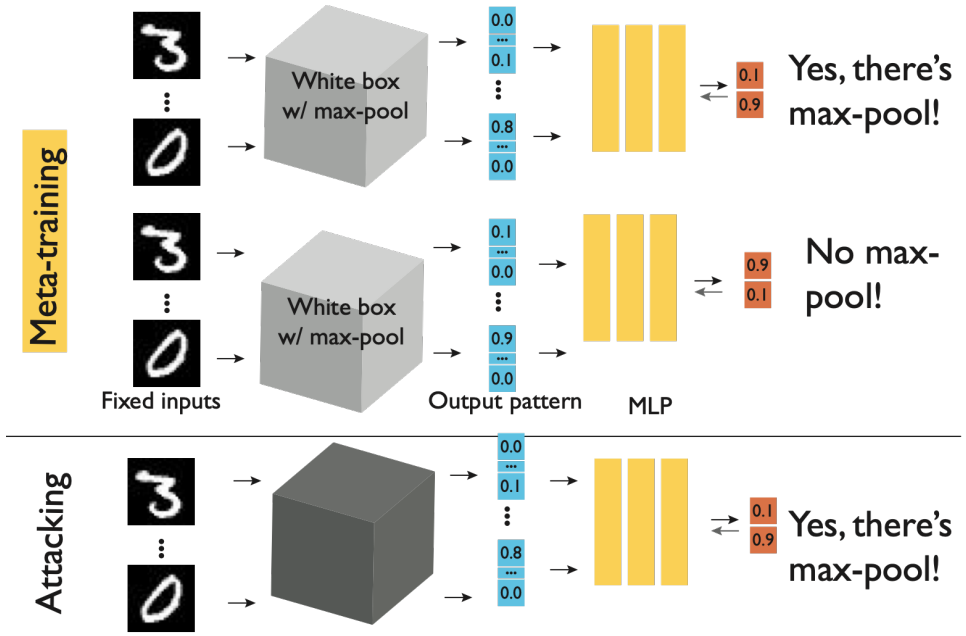
F_V

	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	All_0, $\text{Half}_{0/1}$, $\text{Quarter}_{0/1/2/3}$
	size	*Data size*	All, Half, Quarter

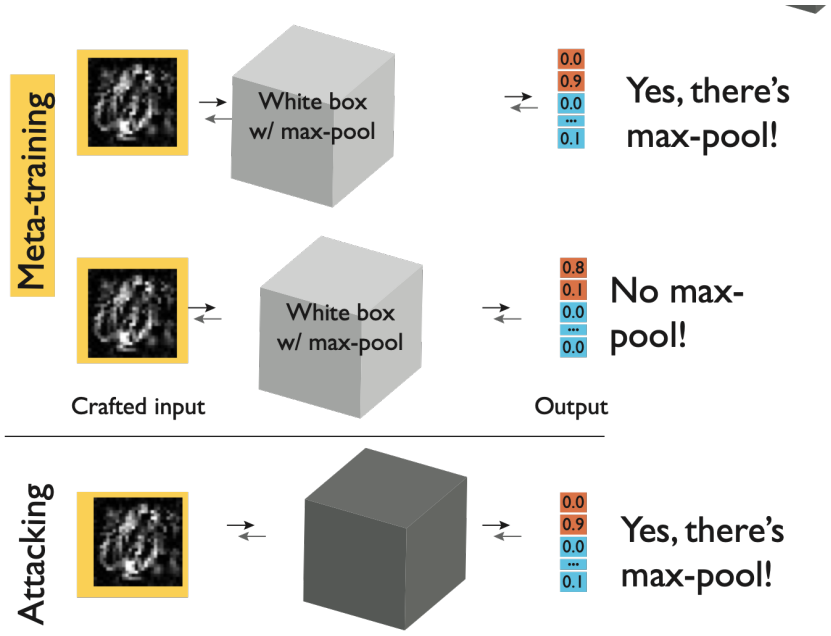
- Can those be inferred from black box access? What does the prediction leak about the **model**?



Method



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

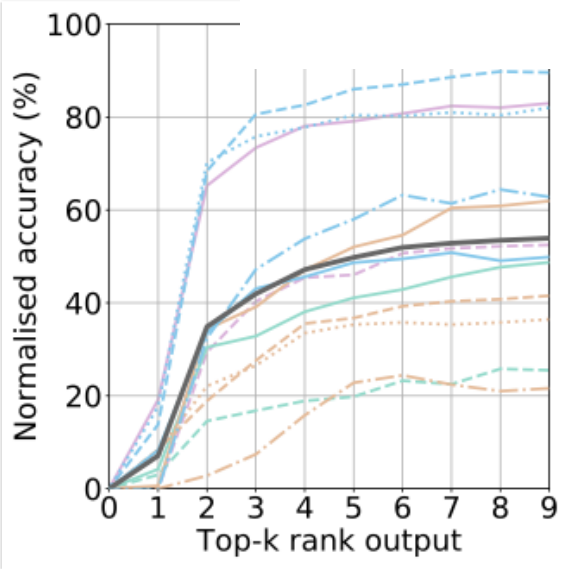
Results

Method	Output	architecture								optim		data		avg
		act	drop	pool	ks	#conv	#fc	#par	ens	alg	bs	size	split	
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**

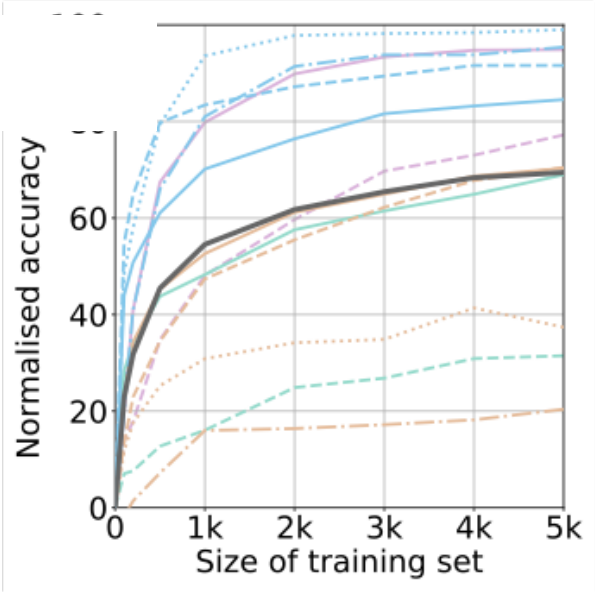
- Always far above chance.
- `kennen-o` is stealthy and effective.
- Easy: act, drop, pool, ks, split.
- #layers can be exposed.
- Optimisation hyperparams can be exposed.

- Don't need full score output; top-k is good enough.
- `kennen-i` is query-efficient and effective.
- `kennen-io` is not stealthy but works best in 11/12 attr.

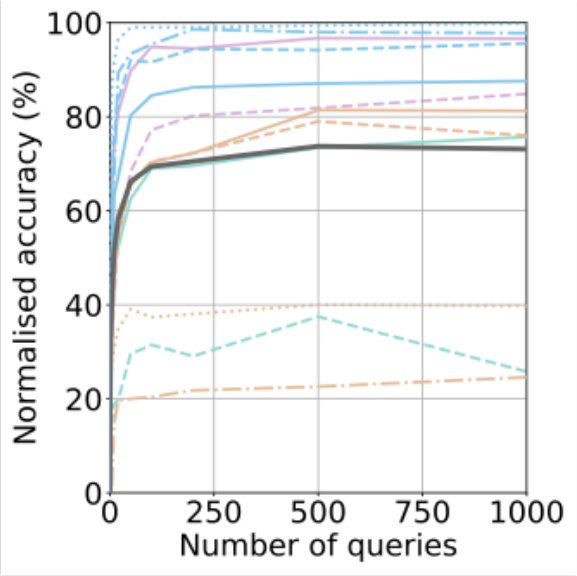
Factor analysis



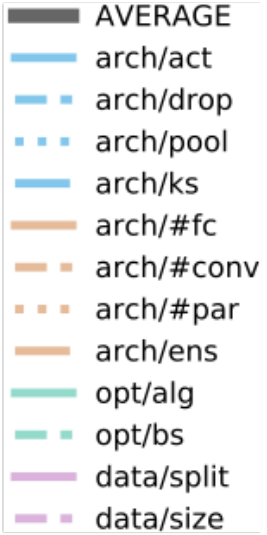
Output richness: top-2 and below are informative.



#training models: hasn't saturated @ 5k.



#queries: saturates @ 100s.



Reverse Engineering for Improve Black Box Attacks

Family	Variants	#layers	Top-5 error
SqueezeNet	v1.0,v1.1	[26,26]	[19.4,19.6]
VGG	11,13,16,19	[11,19]	[9.1,11.4]
VGG-**B**N	11,13,16,19	[11,19]	[8.2,10.2]
ResNet	18,34,50,101,152	[21,156]	[5.9,10.9]
DenseNet	121,161,169,201	[121,201]	[6.2,7.8]

- 19 classifiers from 5 families: high intr-diversity and inter-similarity.
- Family prediction task (cross-validation).

ImageNet Results

Scenario	Generating nets	MC(%)
White box	Single white box	100.0
Family black box	GT family	86.2
Black box whitened	**Predicted family**	**85.7**
Black box	Multiple families	82.2

- Internal exposure can make the transfer based attack more targetted and effective (82.2% → 85.7%).

Conclusion

- Black-box access reveals a lot (e.g. whether SGD or ADAM has trained it).

- Use meta-models; training models don’t need to be very close to test model.

- Exposed internals
  - Future challenges to protect IP
  - Greater vulnerability when turning black box into white box

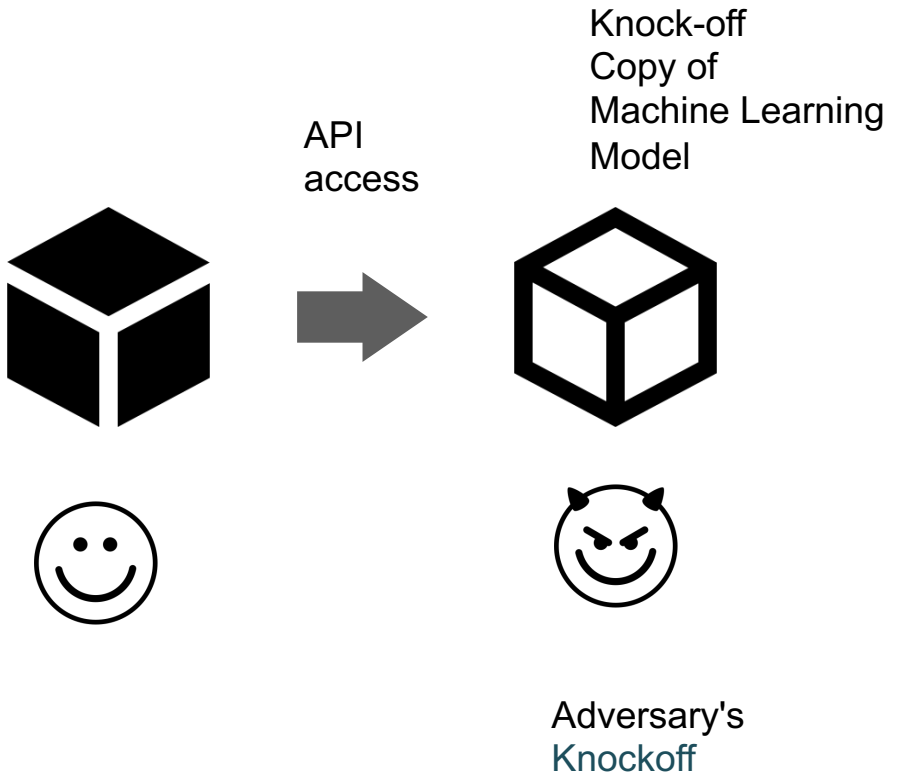
Reverse Engineering Neural Networks (ICLR'18)

Method	Output	architecture								optim		data		avg
		act	drop	pool	ks	#conv	#fc	#par	ens	alg	bs	size	split	
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)

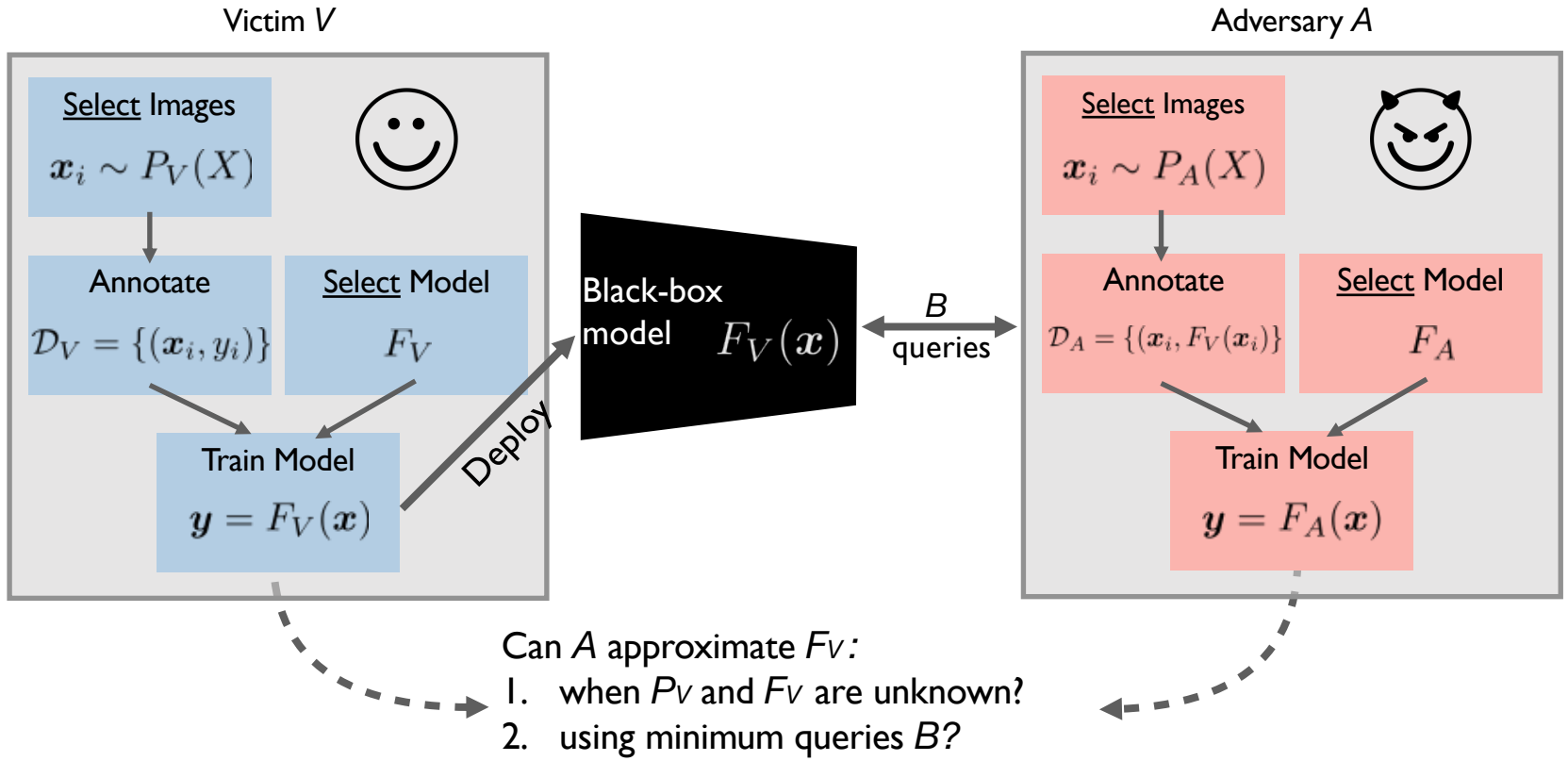
- **Functionality** stealing generates copy that behaves similarily
- Copy might differ internally – should be indistinguishable from the outside



- 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?

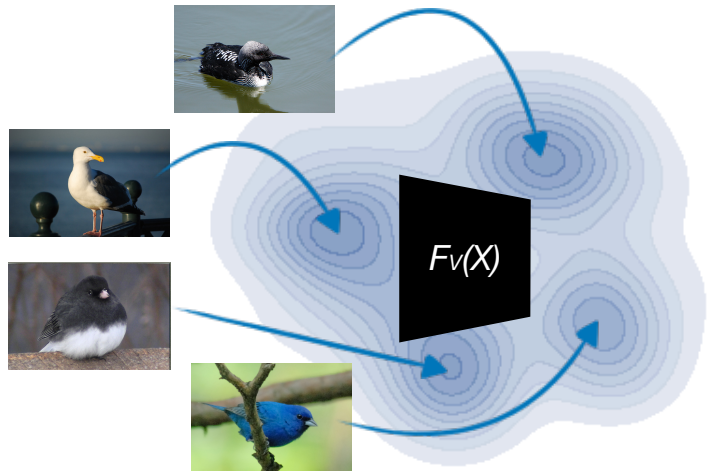
Functionality Stealing: Knock-Off Nets (CVPR'19)



Resembles Model Distillation … but under weaker assumptions

Query Set Selection: Challenge

$P_V(X)$



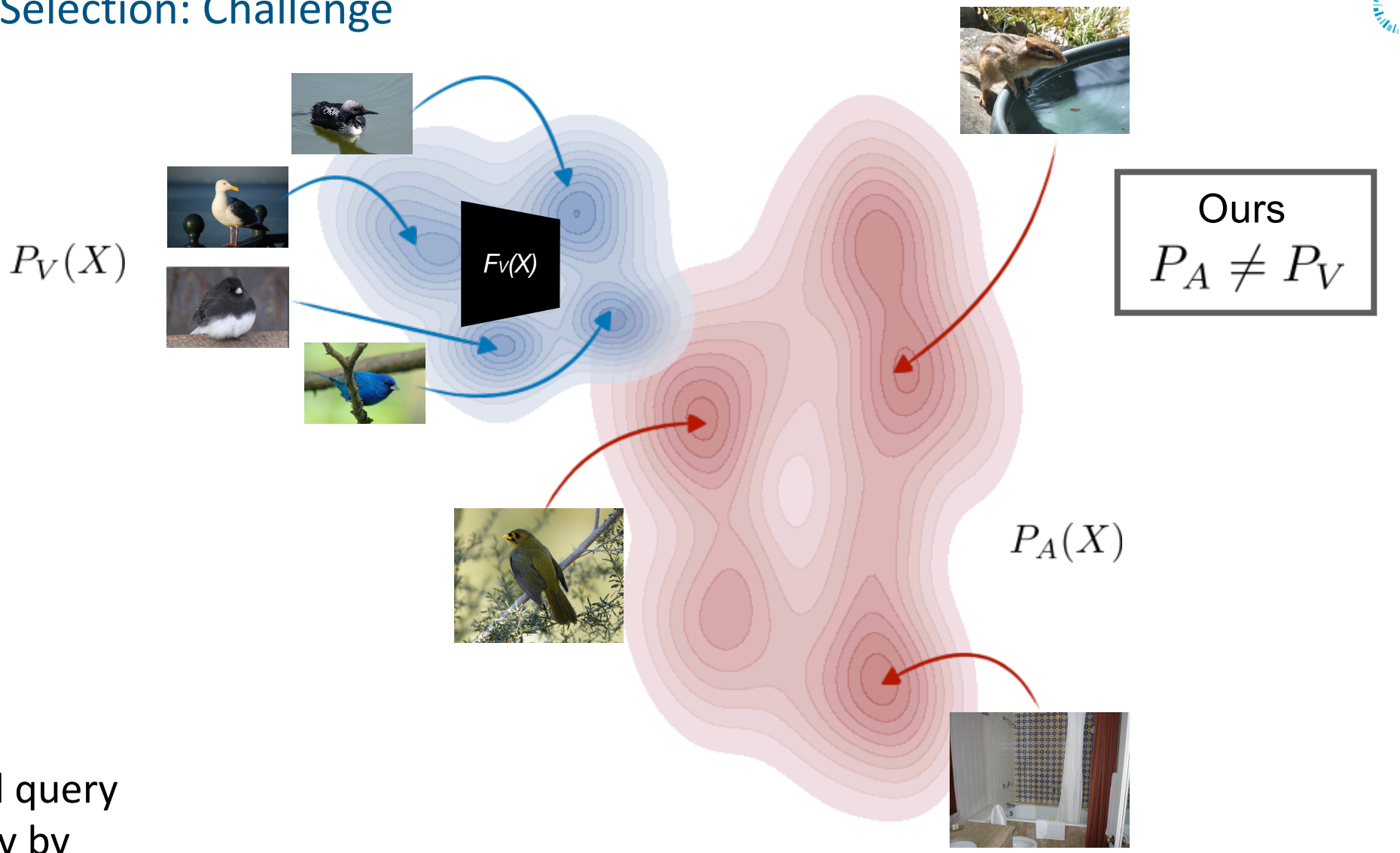
Active Learning

Distillation

Student-Teacher

$$P_V = P_A$$

Query Set Selection: Challenge



- Improved query efficientcy by Reinforcement Learning

Setup

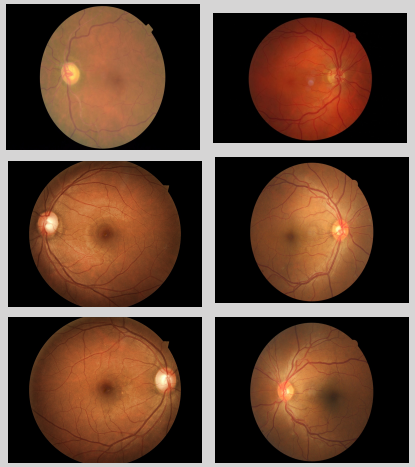
4 Blackbox Models *Fv(X)*

Caltech256

CUB200

Indoor67

Diabetic5



$x \overset{\pi}{\sim} P_A(X)$

CUB200

$y = F_V(x)$

.01	Bobolink
.13	Indigo Bunting
.42	Gray Catbird
⋮	200 bird species
.03	Winter Wren

Transfer Set

$$\{(x_t,\ F_V(x_t))\}_{t=1}^{B}$$

Train (Knockoff)

F_A

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

x_1

x_2

x_B

Bird classifier

Victim's Blackbox

y_1 Harris Sparrow: 0.41
Frigate bird: 0.06
Bronzed Cowbird: 0.05

y_2 Harris Sparrow: 0.73
Gadwall: 0.08
Tree Sparrow: 0.06

y_B Harris Sparrow: 0.19
Pine Grosbeak: 0.17
Myrtle Warbler: 0.11

Knockoff Bird classifier

Adversary's Knockoff

test

Harris Sparrow: 0.81

Harris Sparrow: 0.52

Harris Sparrow: 0.71

Can we learn with π = random? Yes!

CISPA HELMHOLTZ-ZENTRUM i. G.

Transfer Set

Gadwall: 0.65
Nighthawk: 0.06
Horned Lark: 0.05

Gadwall: 0.31
B. Swallow: 0.14
N. Flicker: 0.12

Gadwall: 0.14
Chuck. Widow: 0.13
Swain. Warbler: 0.1

Lin. Sparrow: 0.82
Ovenbird: 0.07
House Sparrow: 0.03

Lin. Sparrow: 0.49
Mockingbird: 0.18
N. Waterthrush: 0.07

Lin. Sparrow: 0.32
Song Sparrow: 0.06
Tree Sparrow: 0.05

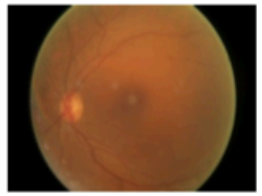
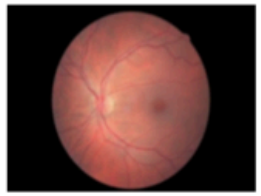
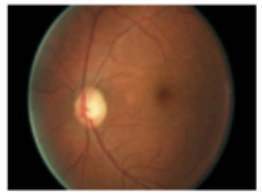
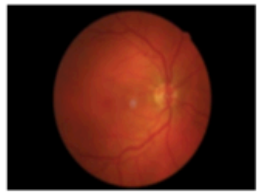
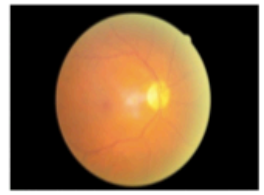
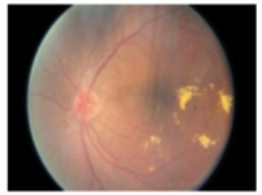
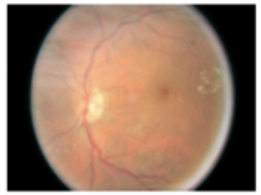
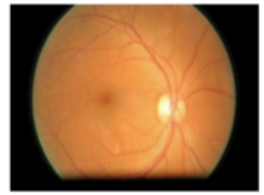
Cactus Wren: 0.95
W. Meadowlark: 0.02
Lin. Sparrow: 0.00

Cactus Wren: 0.88
Geococcyx: 0.04
W. Meadowlark: 0.03

Cactus Wren: 0.33
N. Flicker: 0.28
Lin. Sparrow: 0.07

Can we learn with π = random? Yes!

CISPA HELMHOLTZ-ZENTRUM i. G.

Transfer Set			Test Set		
					
No DR: 0.73 Proliferative: 0.12 Moderate: 0.08	No DR: 0.48 Mild: 0.36 Moderate: 0.16	No DR: 0.30 Moderate: 0.29 Proliferative: 0.28	No DR: 0.50 Mild: 0.33 Moderate: 0.13	No DR: 0.36 Mild: 0.33 Moderate: 0.28	Mild: 0.53 No DR: 0.43 Moderate: 0.03
					
Moderate: 0.69 No DR: 0.31 Mild: 0.01	Moderate: 0.63 No DR: 0.15 Severe: 0.13	Moderate: 0.35 Mild: 0.32 No DR: 0.22	Moderate: 0.48 Mild: 0.316 No DR: 0.21	Moderate: 0.35 Mild: 0.31 No DR: 0.23	No DR: 0.36 Mild: 0.33 Moderate: 0.26
					
Severe: 0.73 Proliferative: 0.23 Moderate: 0.04	Severe: 0.70 Proliferative: 0.30 Moderate: 0.00	Severe: 0.53 Mild: 0.16 Moderate: 0.15	Severe: 0.57 Moderate: 0.23 Proliferative: 0.19	Severe: 0.41 Proliferative: 0.29 Moderate: 0.24	Moderate: 0.62 Severe: 0.16 Mild: 0.13

Can we learn with π = random? Yes!

		random			
	P_A	Caltech256	CUBS200	Indoor67	Diabetic5
Closed	P_V (F_V)	78.8 (1×)	76.5 (1×)	74.9 (1×)	58.1 (1×)
	P_V (KD)	82.6 (1.05×)	70.3 (0.92×)	74.4 (0.99×)	54.3 (0.93×)
	D^2	76.6 (0.97×)	68.3 (0.89×)	68.3 (0.91×)	48.9 (0.84×)
Open	ILSVRC	75.4 (0.96×)	68.0 (0.89×)	66.5 (0.89×)	47.7 (0.82×)
	OpenImg	73.6 (0.93×)	65.6 (0.86×)	69.9 (0.93×)	47.0 (0.81×)

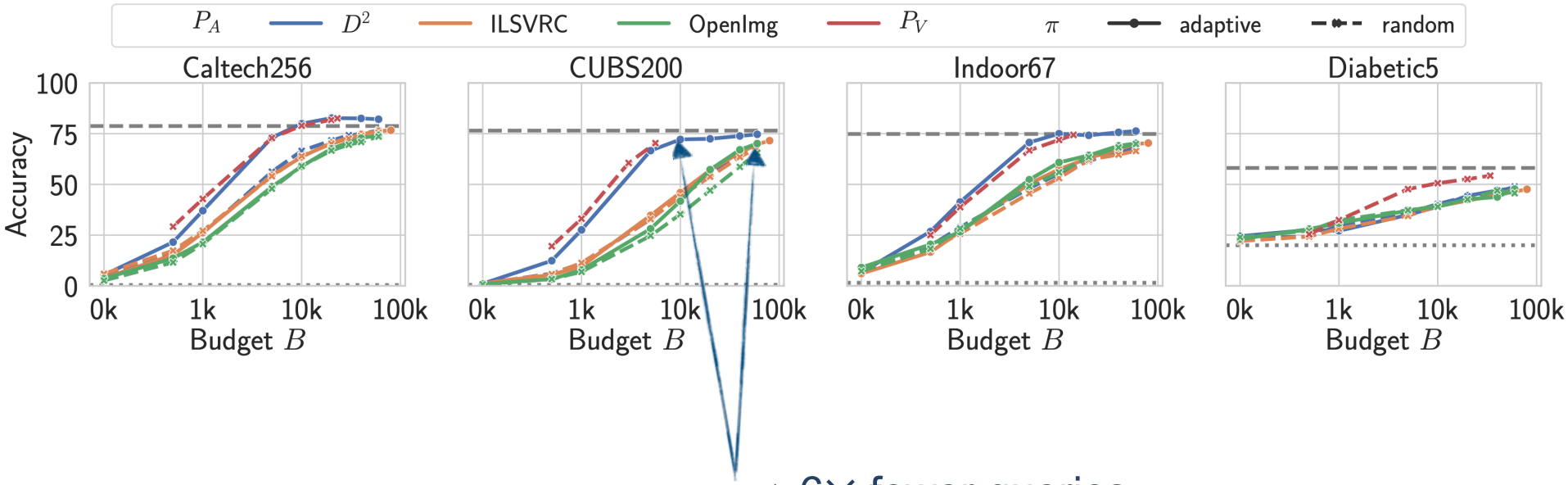
accuracy(victim blackbox)

accuracy(knockoff)

⇒ > 0.81× accuracy of blackbox recovered

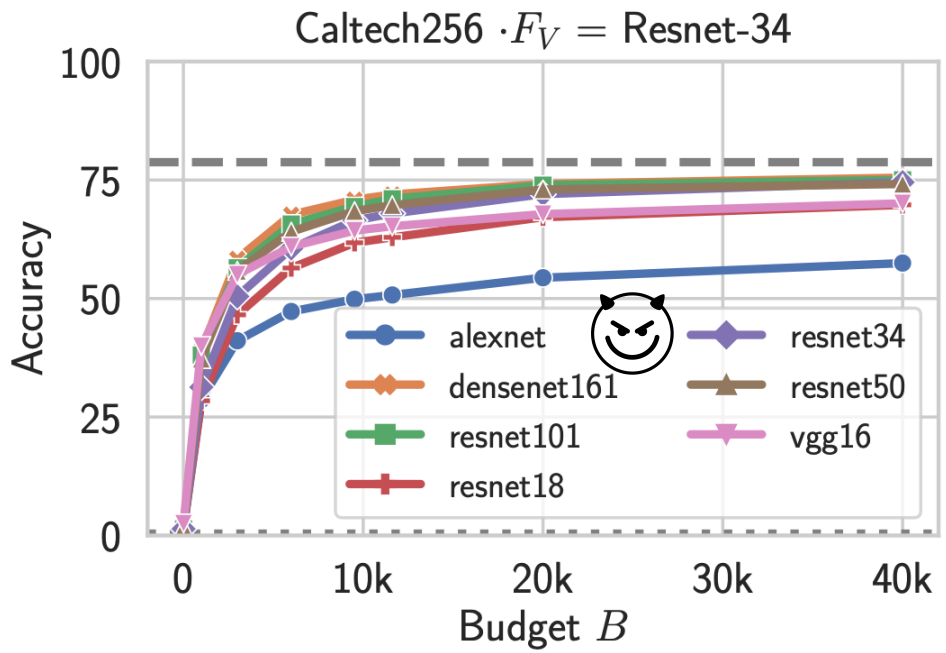
Can make it sample-efficient? Yes*!

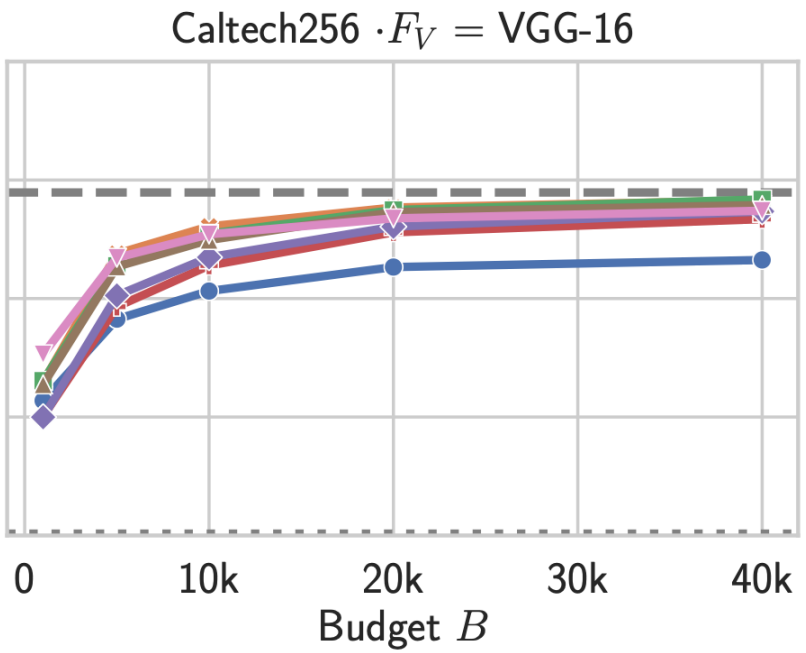
		random				adaptive			
	P_A	Caltech256	CUBS200	Indoor67	Diabetic5	Caltech256	CUBS200	Indoor67	Diabetic5
	P_V (F_V)	78.8 (1×)	76.5 (1×)	74.9 (1×)	58.1 (1×)	-	-	-	-
	P_V (KD)	82.6 (1.05×)	70.3 (0.92×)	74.4 (0.99×)	54.3 (0.93×)	-	-	-	-
Closed	D^2	76.6 (0.97×)	68.3 (0.89×)	68.3 (0.91×)	48.9 (0.84×)	82.7 (1.05×)	74.7 (0.98×)	76.3 (1.02×)	48.3 (0.83×)
Open	ILSVRC	75.4 (0.96×)	68.0 (0.89×)	66.5 (0.89×)	47.7 (0.82×)	76.2 (0.97×)	69.7 (0.91×)	69.9 (0.93×)	44.6 (0.77×)
	OpenImg	73.6 (0.93×)	65.6 (0.86×)	69.9 (0.93×)	47.0 (0.81×)	74.2 (0.94×)	70.1 (0.92×)	70.2 (0.94×)	47.7 (0.82×)



⇒ 6× fewer queries

Can we transfer across architectures? Yes!



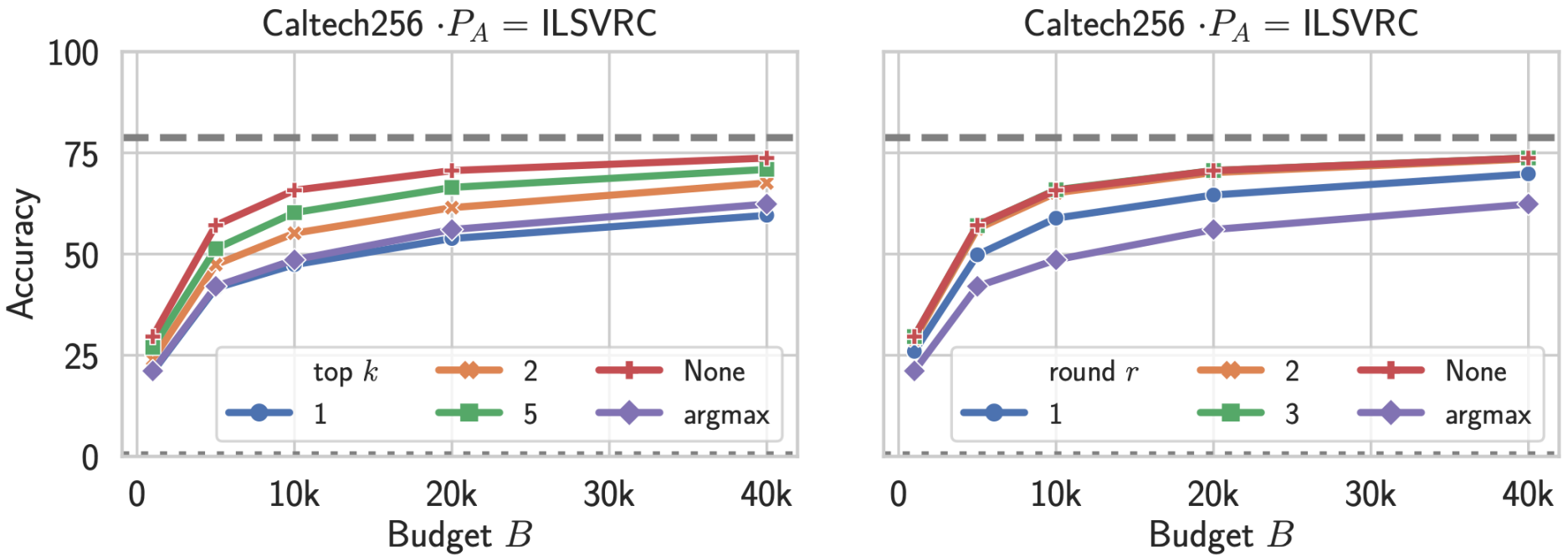


= Resnet-34

= VGG-16

⇒ Robust to choices of blackbox architectures

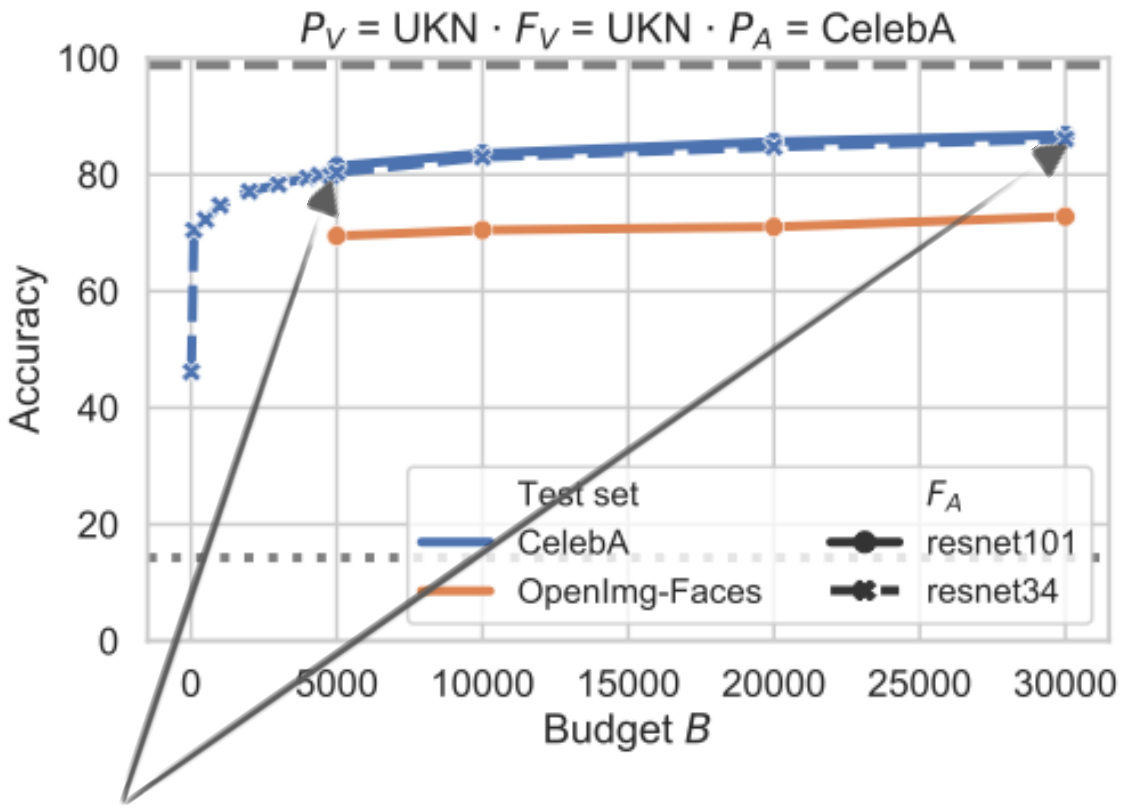
Learning with lesser information? Yes!



⇒ Robust to argmax-preserving defenses
Passive defenses do not really work!

Cost of Attack & Conclusions

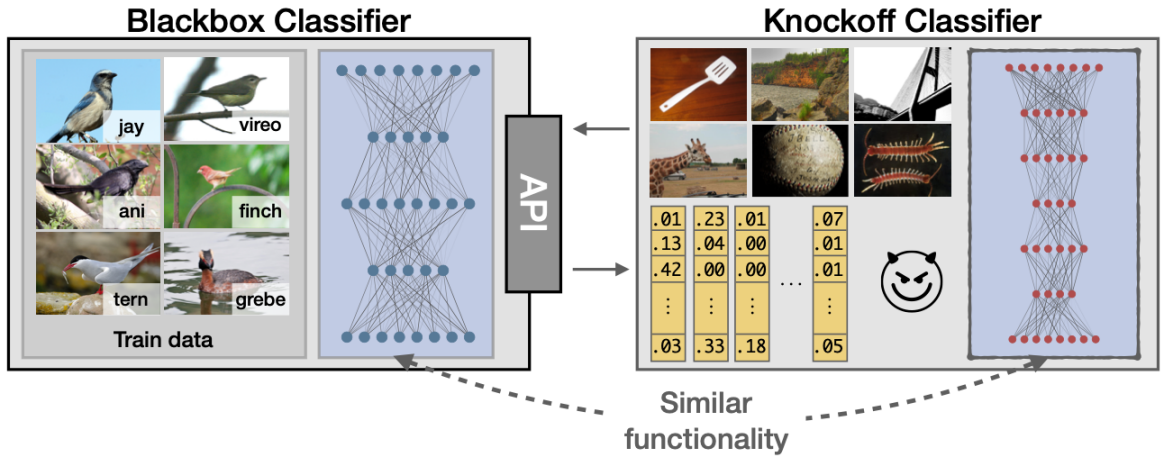
Train: CelebA
Test: CelebA, OpenImg-Faces



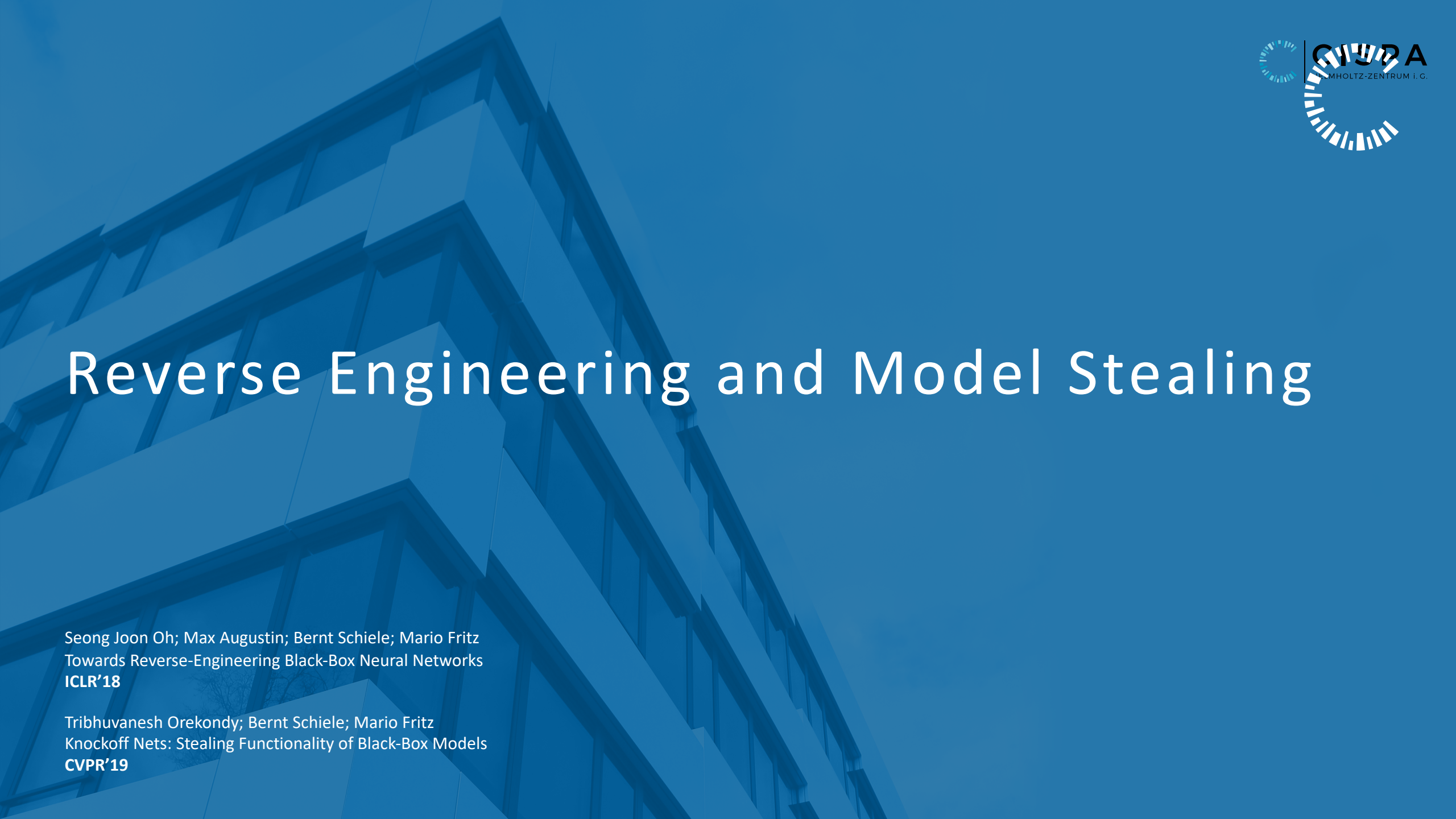
80% accuracy @ B=5k ($0)
86% @ B=30k ($30)

- Strong copy from a few 1000 queries
- Unfortunately difficult to defend
  - Noising
  - Top-k, argmax
  - Rounding
  - MLCapsule – SGX-based deployment [Fritz, ArXiv]
  - Poisoning [Fritz, ICLR'20]
  - Watermarking only post-hoc attribution [Fritz, ArXiv]

Take-aways



- DNN models contain intellectual property: model, parameters, data
- Threat: Can an adversary steal the functionality of a black-box DNN model?
- Model's functionality can be stolen reliably: As few as 10-100K queries ($10-100) are sufficient
- Difficult to defend



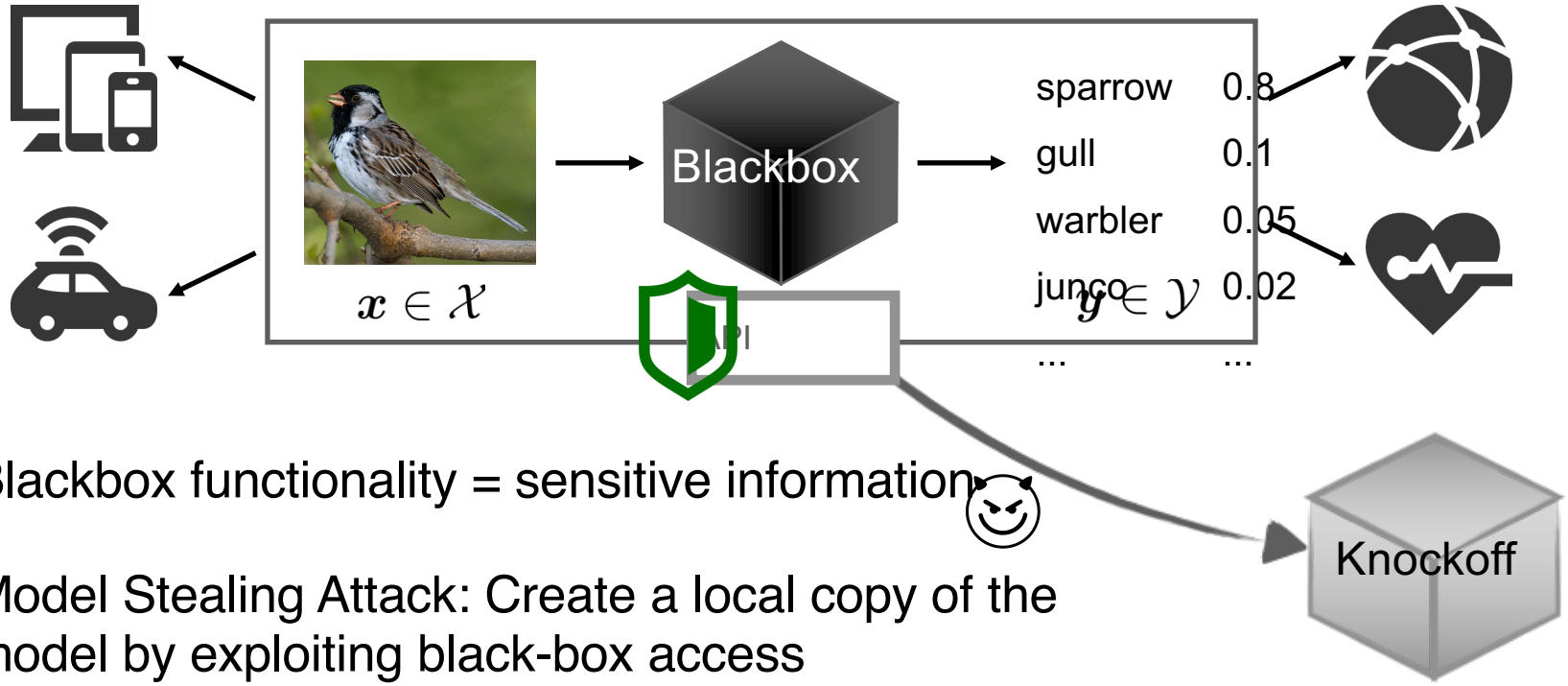
Reverse Engineering and Model Stealing

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

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

Problem: Model Stealing Attacks

- Significant advances in ML
- Deployed into production systems as blackboxes

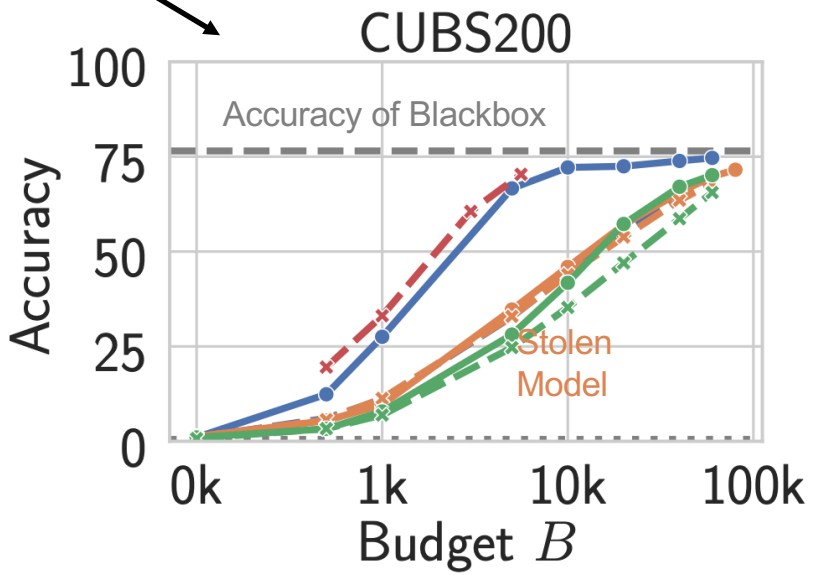


- Blackbox functionality = sensitive information
- Model Stealing Attack: Create a local copy of the model by exploiting black-box access
- Now: Defenses against model stealing

Background: Model Stealing Attacks

- Active research topic
- Challenge: Knowledge-limited Adversary
- Key-idea: Use the blackbox to pseudo-label a set of images → train on (images, pseudo-labels)
- Which images? (a) Any unlabelled set e.g., ImageNet (b) Synthesized data
- Attacks are effective: able to recover 82-96% of black-box accuracy
- Next: How to defend?

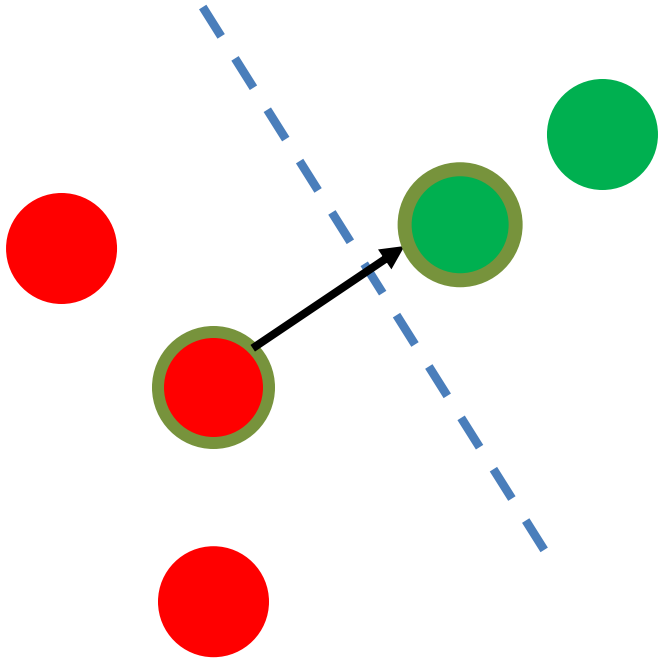
	Black-box type	Proposed Attack	
		Input Query Data	Adapt.?
1. Lowd & Meek (2005)	Linear	Random Noise	✓
2. Nelson et al. (2009)	Linear	Labeled Data	✗
3. Nelson et al. (2010)	Linear	Random Noise	✓
4. Alabdulmohsin et al. (2014)	Linear	Random Noise	✓
5. Tramèr et al. (2016)	Linear, NN	Random Noise	†
6. Milli et al. (2018)	Linear, NN	Random Noise	✓
7. Kesarwani et al. (2018)	Decision Tree	-	-
8. Chandrasekaran et al. (2019)	Linear	Random Noise	✓
9. Papernot et al. (2017b)	CNN	Synth. Data	✓
10. Correia-Silva et al. (2018)	CNN	Unlabeled Data	✗
11. Pal et al. (2019)	CNN	Unlabeled Data	†
12. Orekondy et al. (2019)	CNN*	Unlabeled Data	†
13. Jagielski et al. (2019)	CNN*	Unlabeled Data	✓



Poisoning vs Evasion Attacks

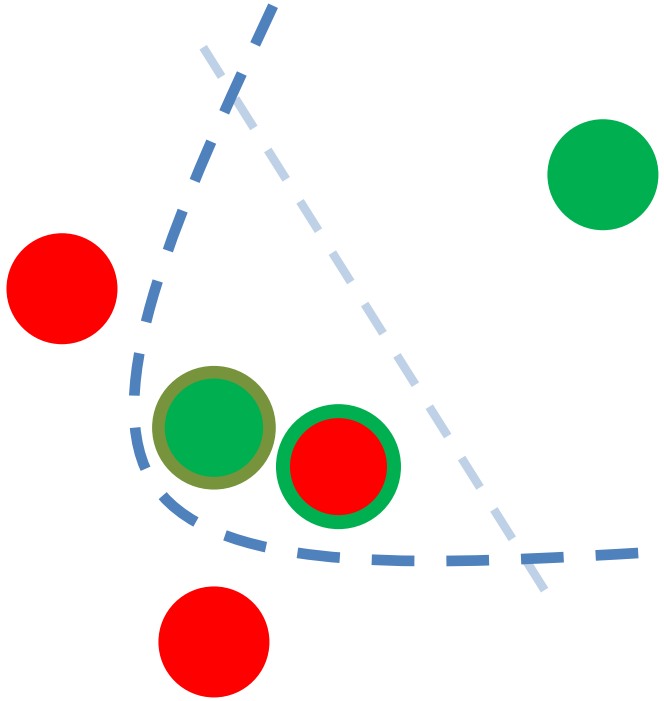
CISPA HELMHOLTZ-ZENTRUM i. G.

Evasion Attack (Adversarial Perturbation)



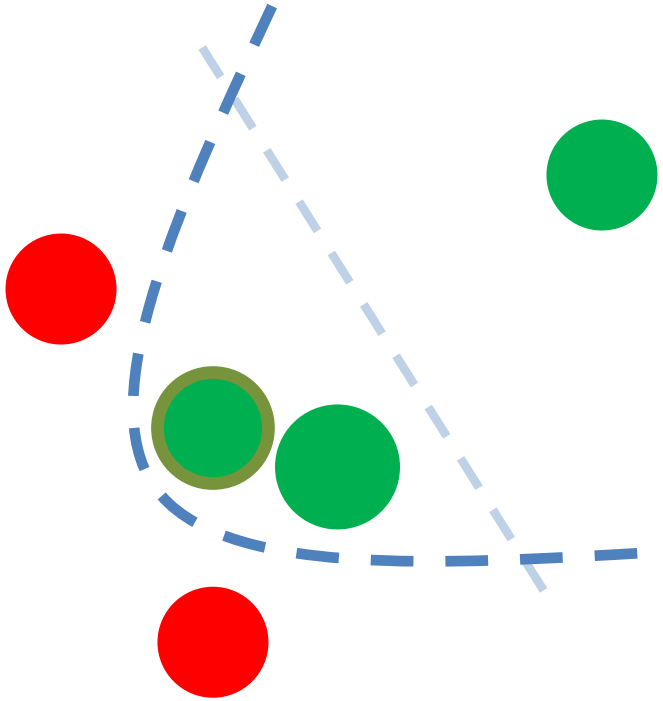
Manipulation of test data

Poisoning Attack



Inject data and label into training set; often wrong label

Clean Data Poisoning Attack

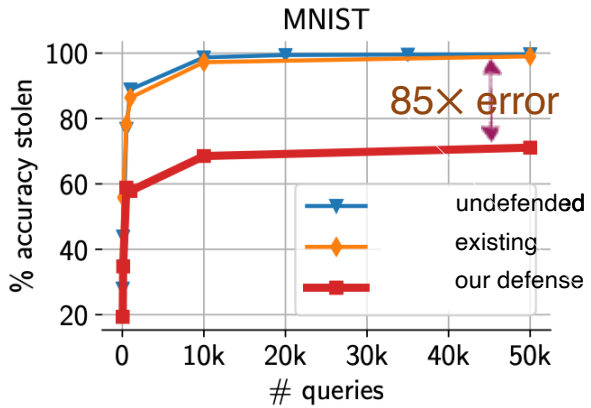


Inject data training set; labeling is correct – can also be done by the victim

Related Work and Limitations

- Defenses
  - Lacking
  - Existing (passive) strategies truncation-based e.g., top-k

- Limitations of Truncation-based defenses
  - Limitation 1: Ineffective
    - Negligible impact of defenses (in simpler settings)
  - Limitation 2: Unbounded utility
    - Cannot bound impact to benign users



Ours: Up to 85× amplification in attacker's error rate

Ours: Bound utility

- Approach: Defense by Adversarial Perturbation/Poisoning
  - Introduce utility-constrained perturbations: $\tilde{\boldsymbol{y}} = \boldsymbol{y} + \boldsymbol{\delta}$
  - Goal of DefenseObjective: Poison attacker's gradient signal via δ

$$\arg\max_{\boldsymbol{\delta}} \ \text{DefenseObjective}(\boldsymbol{\delta}, \cdot) \ \text{ s.t } \ ||\boldsymbol{\delta}|| \leq \epsilon$$

Approach: Intuition

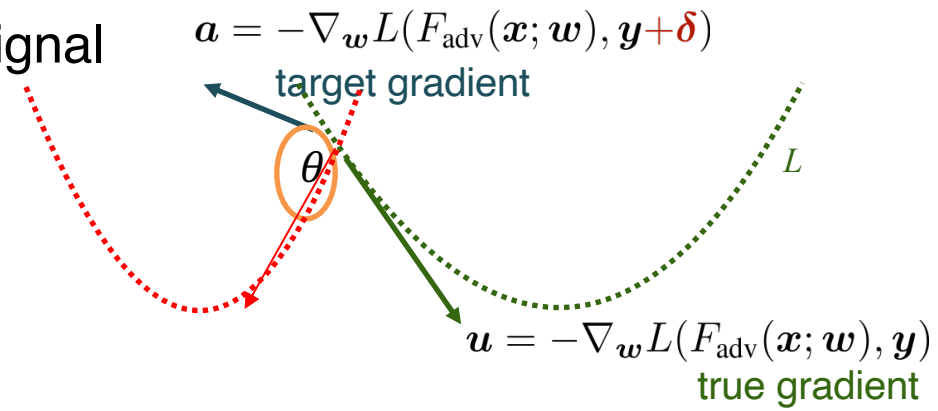
- Intuition: Target attacker's gradient signal
  - Undefended prediction:

$$\boldsymbol{y} = F_{\text{vic}}(\boldsymbol{x})$$

  - Defended prediction:

$$\tilde{\boldsymbol{y}} = F_{\text{vic}}(\boldsymbol{x}) + \boldsymbol{\delta}$$

  - Maximize angular deviation between gradients
  - … subject to utility constraint: L1 distance or preserve argmax



$$\arg\max_{\boldsymbol{\delta}} \text{DefenseObjective}(\boldsymbol{\delta}, \cdot)$$
$$\text{s.t} \quad ||\boldsymbol{\delta}|| \leq \epsilon$$

$$\arg\max_{\boldsymbol{\delta}} \theta$$
$$\text{s.t} \quad ||\boldsymbol{\delta}|| \leq \epsilon$$

Approach: Closer look

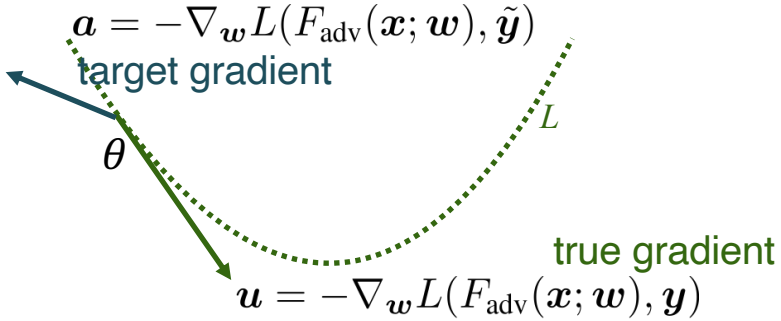
- Intuition: Target attacker's gradient signal

$$\max_{\tilde{\boldsymbol{y}}} \left\| \frac{\boldsymbol{G}^\top \tilde{\boldsymbol{y}}}{||\boldsymbol{G}^\top \tilde{\boldsymbol{y}}||_2} - \frac{\boldsymbol{G}^\top \boldsymbol{y}}{||\boldsymbol{G}^\top \boldsymbol{y}||_2} \right\|_2^2$$

$$\text{where} \quad \boldsymbol{G} = \nabla_{\boldsymbol{w}} \log F_{\text{adv}}(\boldsymbol{x}; \boldsymbol{w})$$

$$\text{s.t} \quad \tilde{\boldsymbol{y}} \in \Delta^K$$

$$\text{dist}(\boldsymbol{y}, \tilde{\boldsymbol{y}}) \leq \epsilon$$

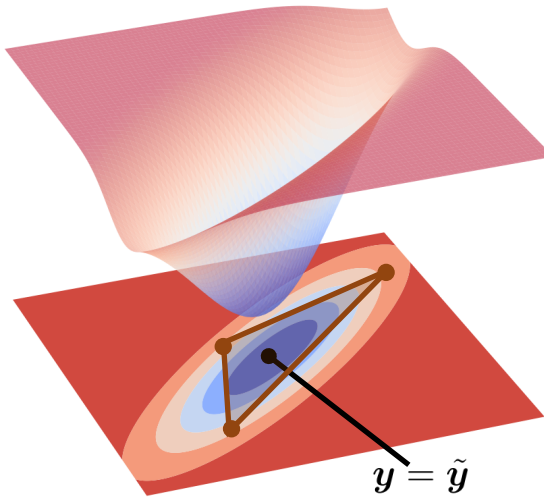


$$\arg\max_{\tilde{\boldsymbol{y}}} \theta \quad \text{s.t} \quad \text{dist}(\boldsymbol{y}, \tilde{\boldsymbol{y}}) \leq \epsilon$$

- Estimating G
  - Unknown adversary model F_{adv}
  - Estimate via F_{sur}
  - Determined empirically; transfers effectively

- Performing Optimization
  - Gradient-based optimization: problematic
  - Heuristic solver

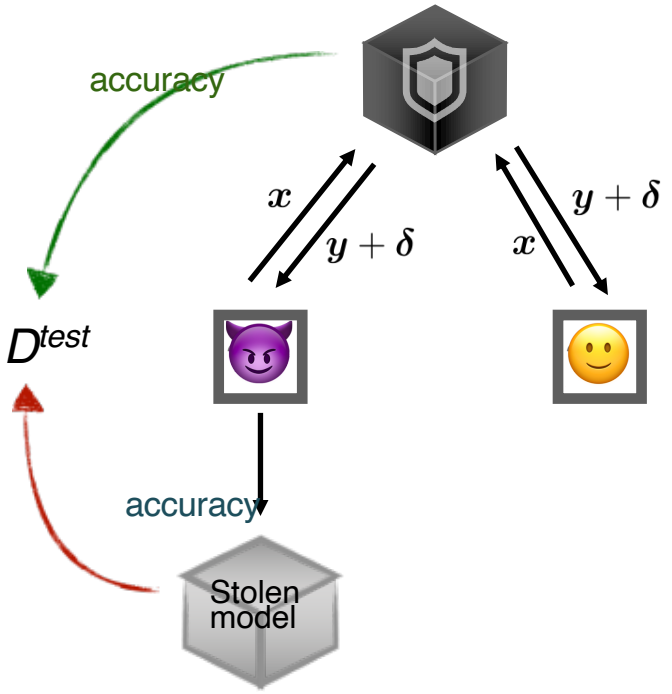
  - Idea: Search extremes of convex set



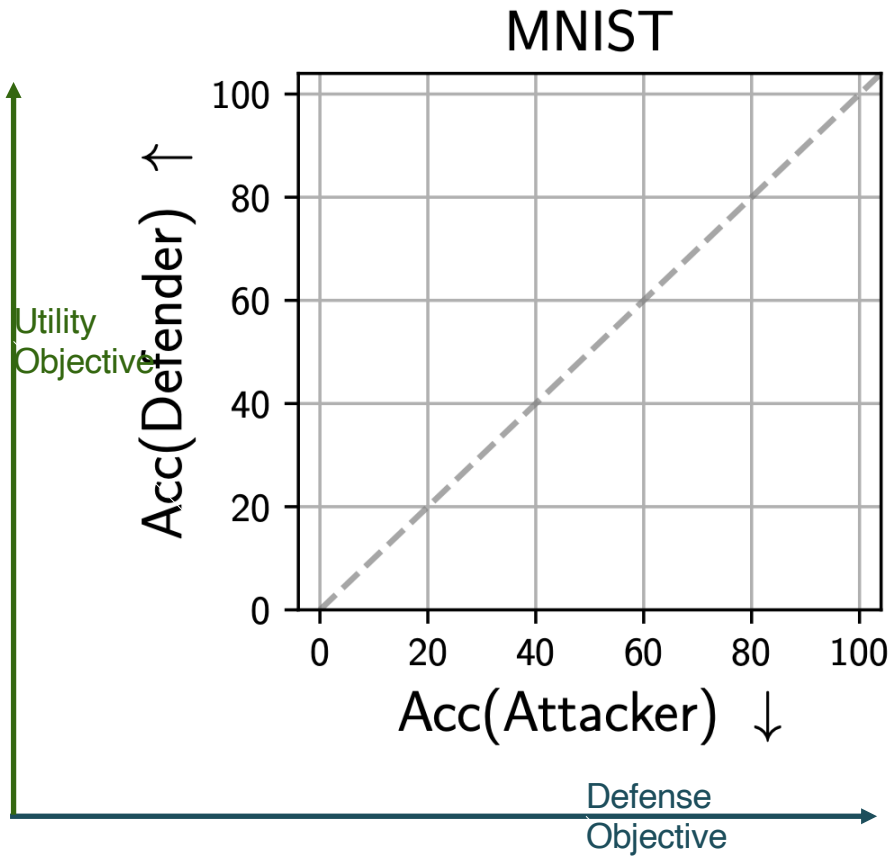
Evaluation: Setup

- Threat Model
  - Defender does not know
    - Whether a query *x* is malicious/benign
    - Attacker's strategy
  - Perturbation to each query
    - Applied independently
    - Using an identical strategy

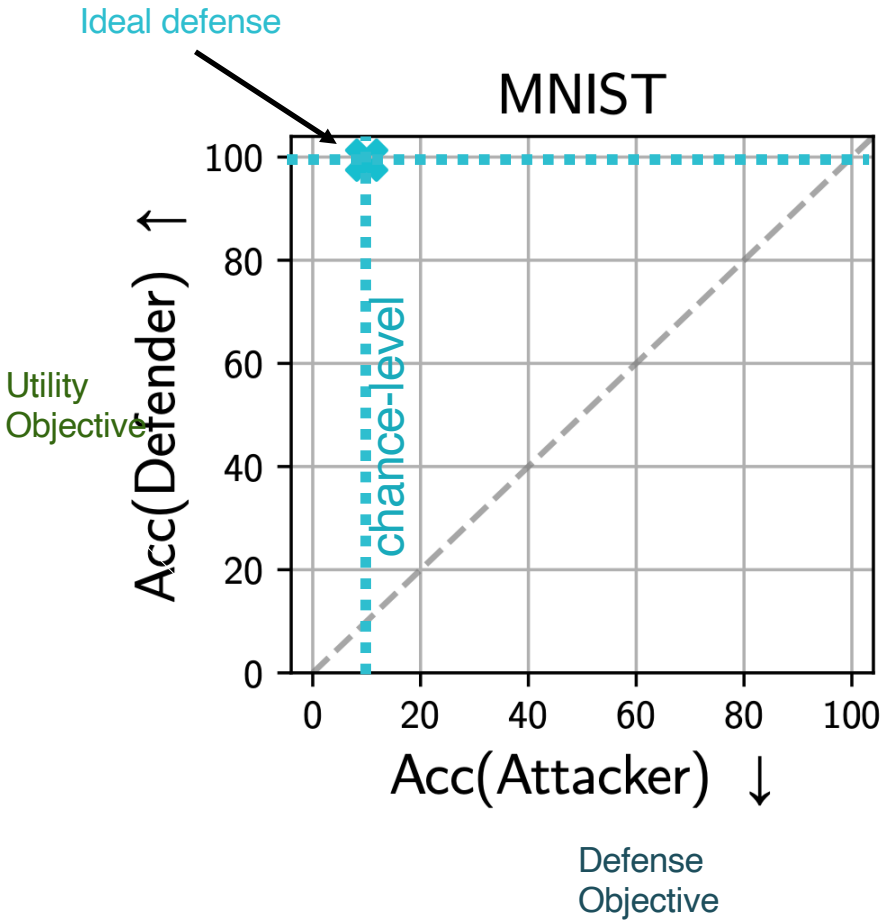
- Evaluation metrics
  - Defense objective (for attackers):
    - *Reduce* test accuracy of *stolen* model
  - Utility objectives (for benign users):
    - *Unchanged* test accuracy of *defense* model
    - *Reduce* perturbation $\|\delta\|$



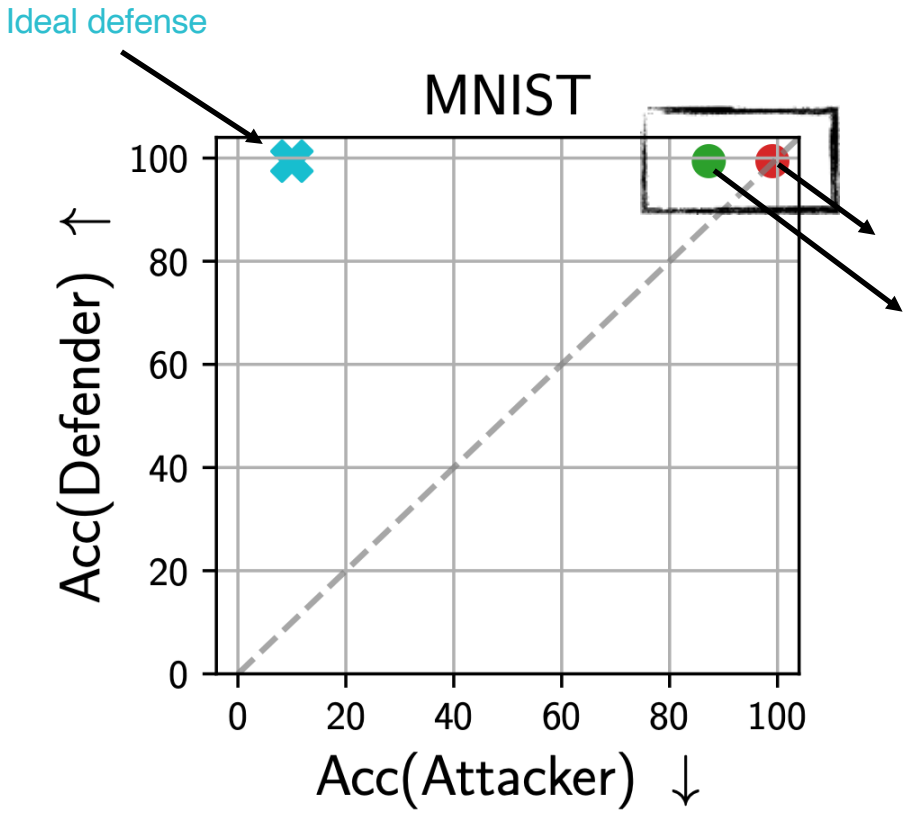
Results


MNIST
Acc(Defender) ↑
Acc(Attacker) ↓
0
20
40
60
80
100
Utility Objective
Defense Objective

Results


Ideal defense
MNIST
100
80
60
40
20
0
Acc(Defender) ↑
chance-level
Utility
Objective
0
20
40
60
80
100
Acc(Attacker) ↓
Defense
Objective

Results



No defense

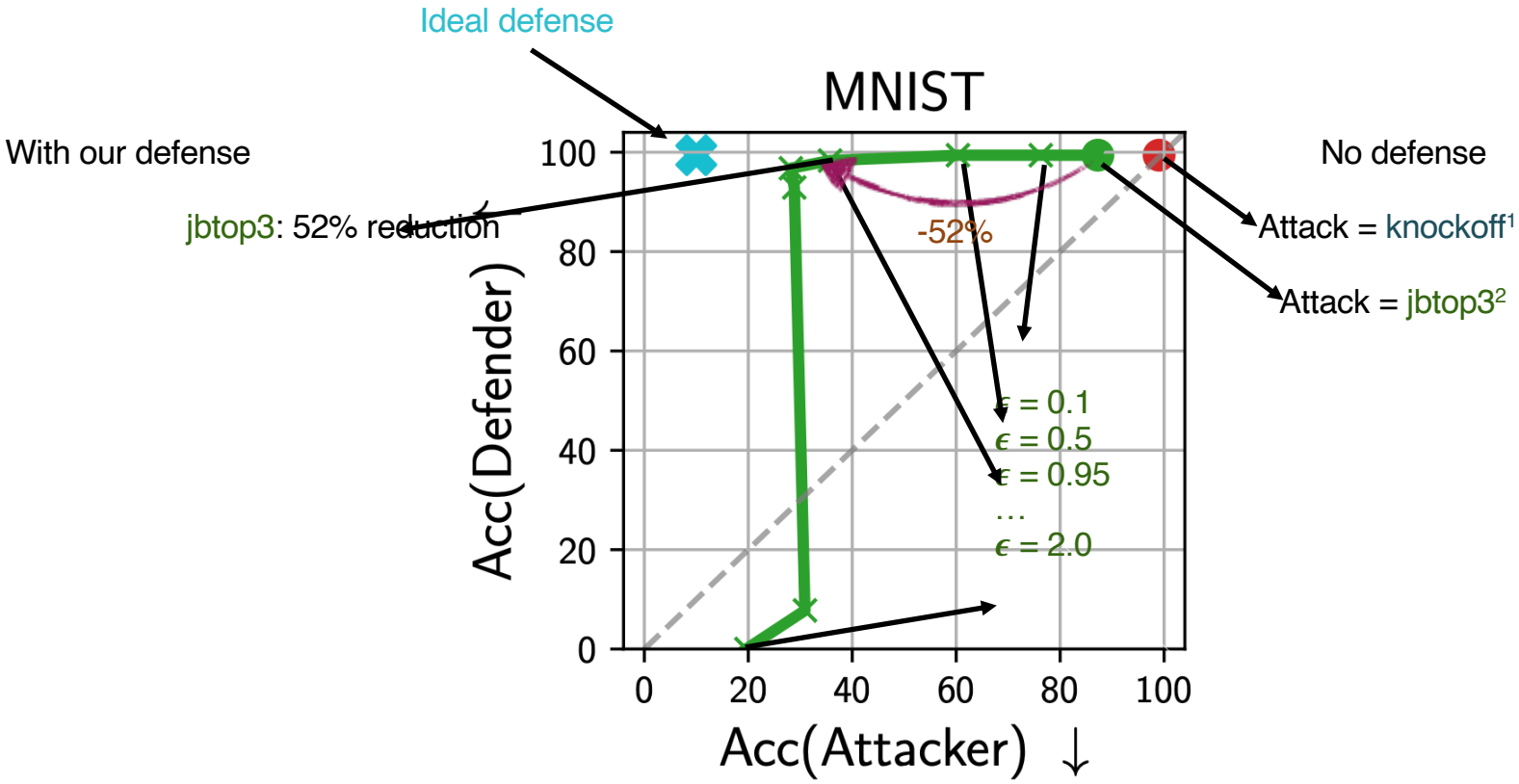
Attack = knockoff[1]

Attack = jbtop3[2]

- When model is undefended
  - Attacks[1,2,3] successfully recover 85-100% of victim model's test accuracy

[1] Orekondy et al., "Knockoff nets: Stealing functionality of black-box models." CVPR '19
[2] Juuti et al., "PRADA: Protecting against DNN Model Stealing Attacks", Euro S&P '19
[3] Papernot et al., "Practical Black-Box Attacks against Machine Learning." Asia CCS '17

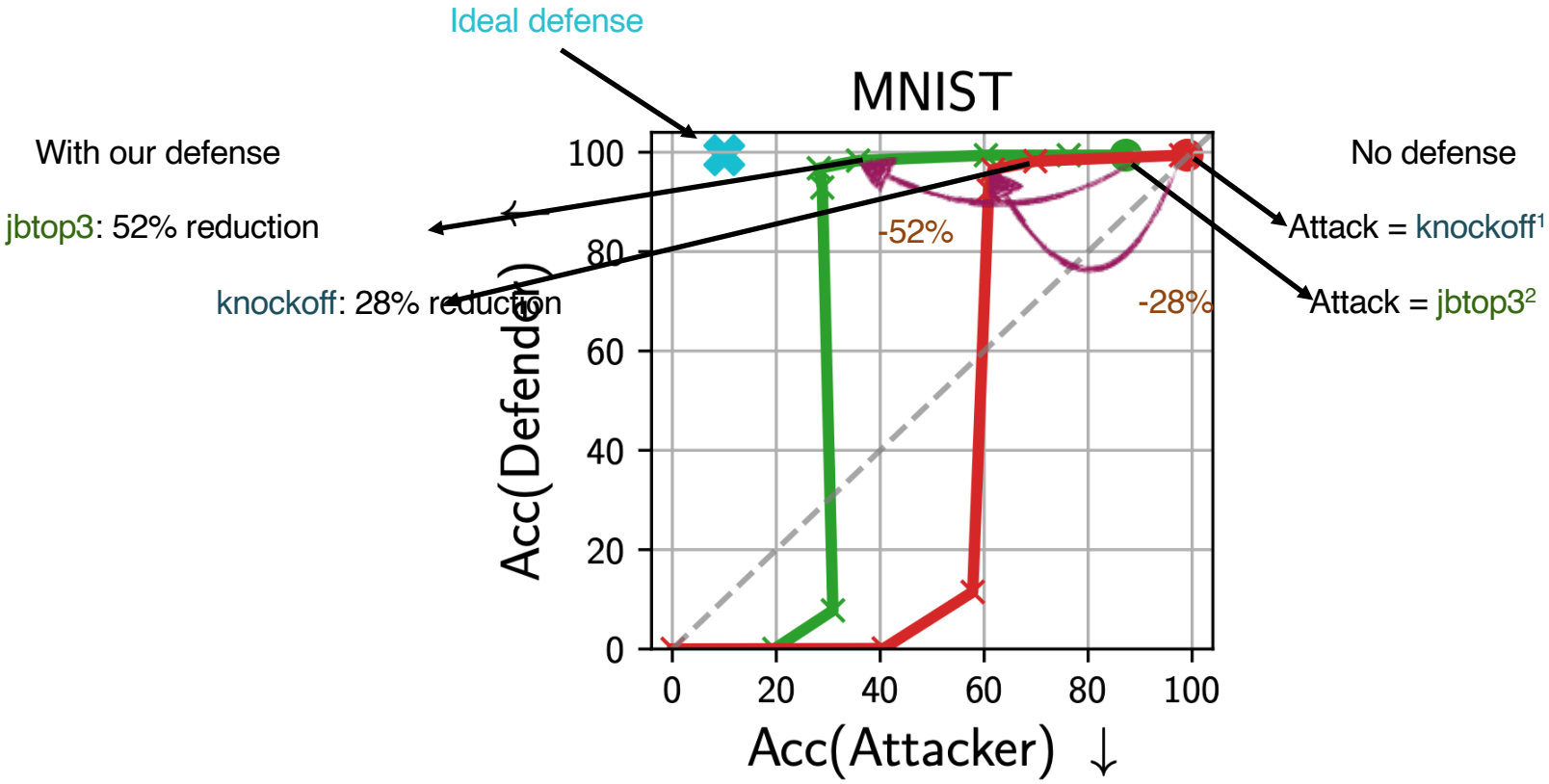
Results



- With MAD defense
  - Performance of attacks significantly deteriorate, with marginal impact to utility

[1] Orekondy et al., "Knockoff nets: Stealing functionality of black-box models." CVPR '19
[2] Juuti et al., "PRADA: Protecting against DNN Model Stealing Attacks", Euro S&P '19
[3] Papernot et al., "Practical Black-Box Attacks against Machine Learning." Asia CCS '17

Results



- With MAD defense
  - Performance of attacks significantly deteriorate, with marginal impact to utility

[1] Orekondy et al., "Knockoff nets: Stealing functionality of black-box models." CVPR '19
[2] Juuti et al., "PRADA: Protecting against DNN Model Stealing Attacks", Euro S&P '19
[3] Papernot et al., "Practical Black-Box Attacks against Machine Learning." Asia CCS '17

Watermarking of ML Models

Protecting Intellectual Property of Deep Neural Networks with Watermarking

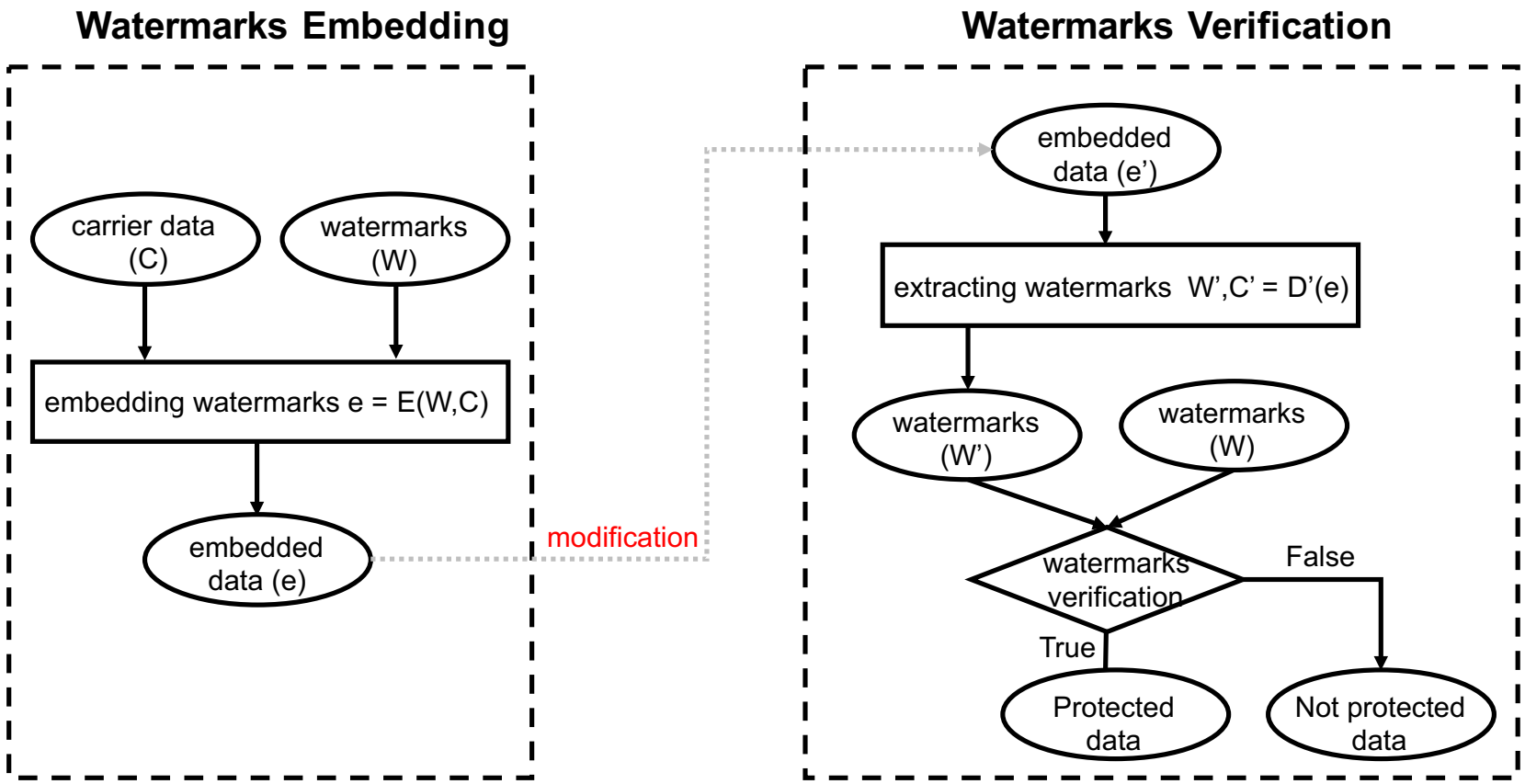
CISPA HELMHOLTZ-ZENTRUM i. G.

- *Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph. Stoecklin, Heqing Huang, Ian Molloy*
- *ASIACCS'18*

Motivation

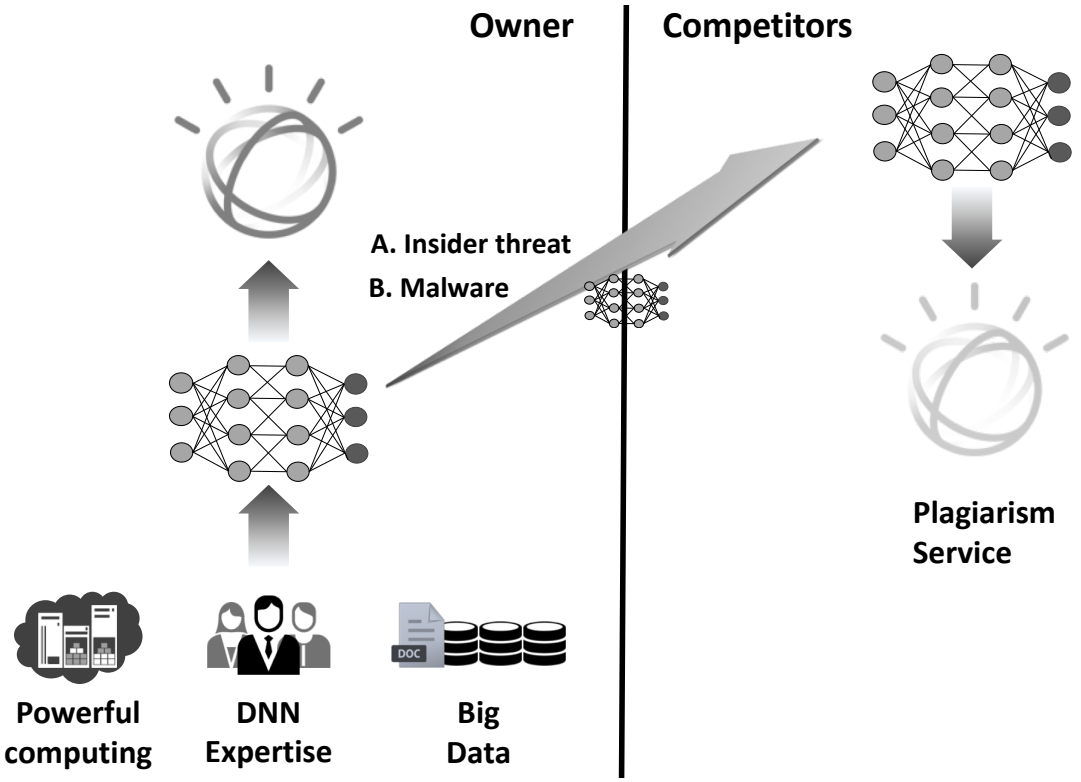
- AI / ML technology embeddeded into many systems
- Building such models requires:
  - Expertise
  - Data
  - Annotation
  - Computation
- Potential of copyright infringement / IP violations by
  - Illegal reproduction
  - Distributiuon
  - Derivation
- Actual legal situation a bit unclear:
  - Law and Adversarial Machine Learning:
    Ram Shankar Siva Kumar, David R. O'Brien, Kendra Albert, Salome Vilojen
    https://arxiv.org/abs/1810.10731

Watermarking

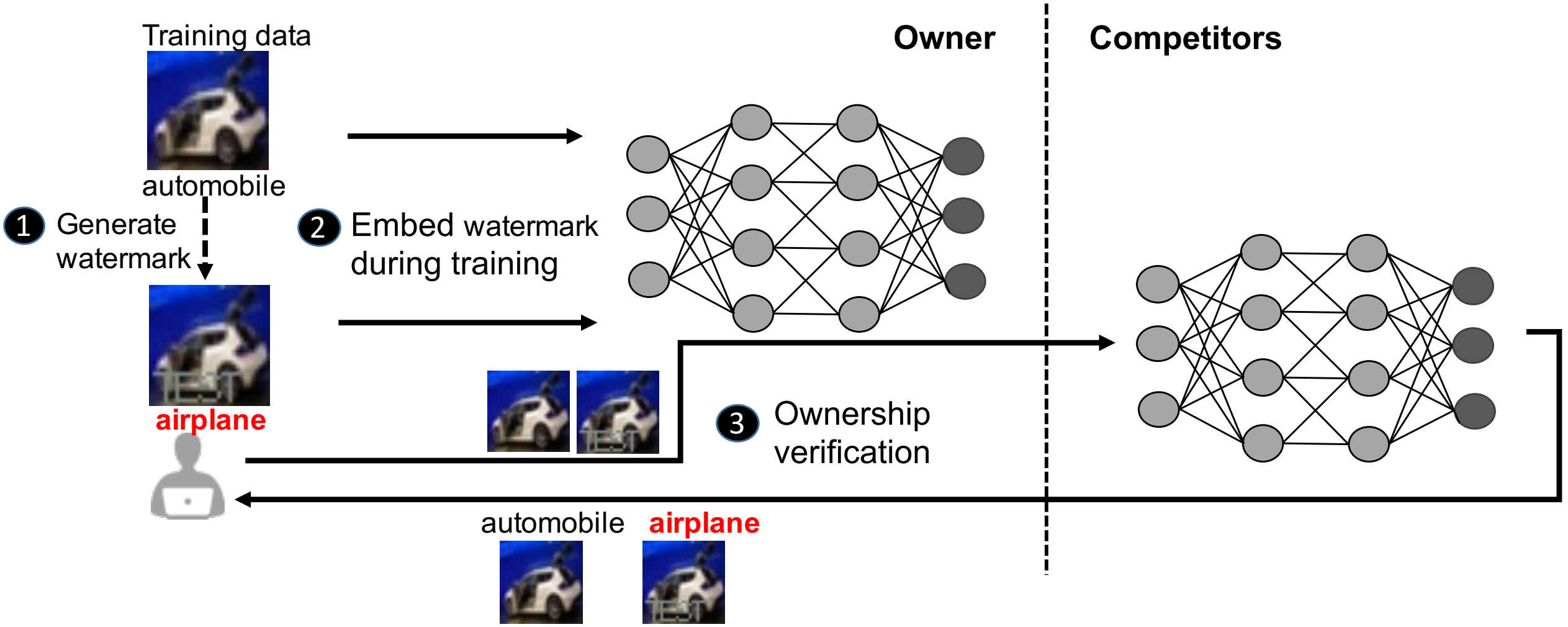


Idea

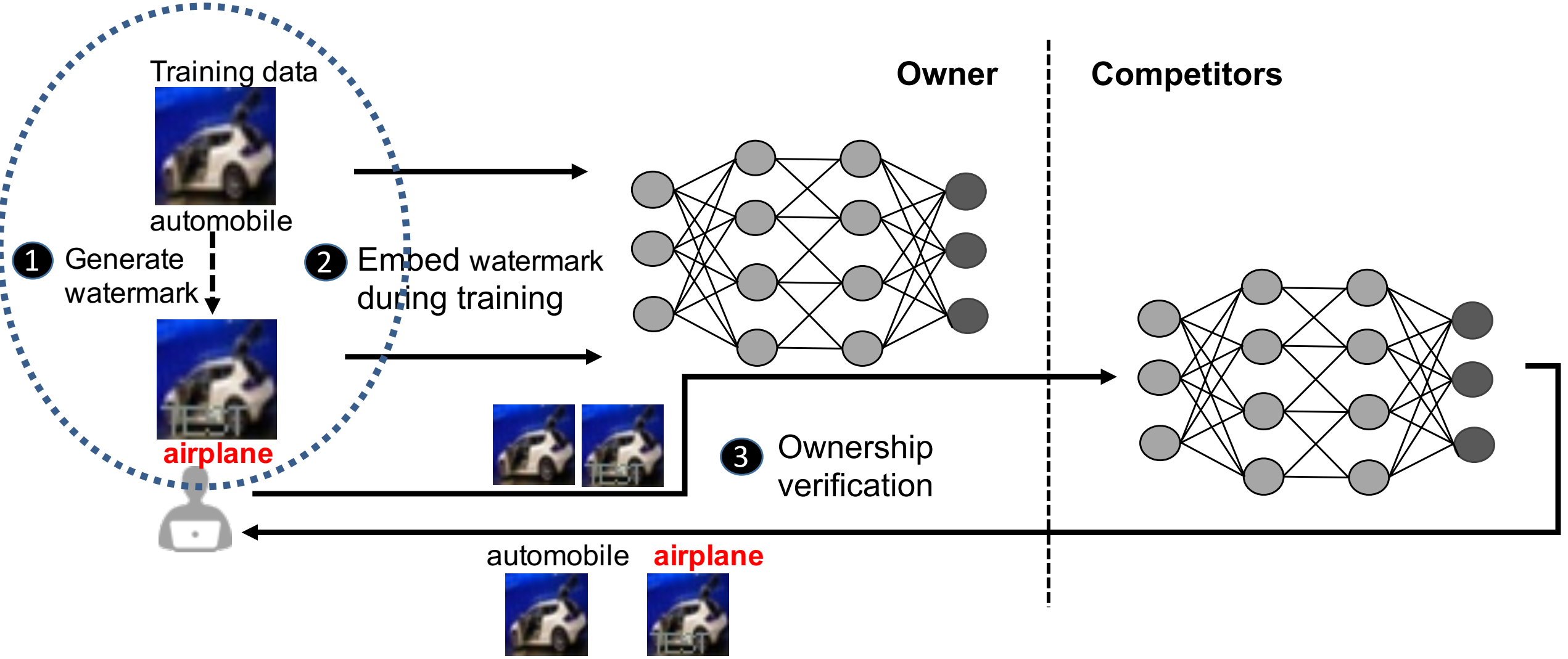
- Watermark in Deep Learning
- Allow for verifying the ownership
- Special training that delivers characteristic output for special examples
- Needs to be robust / resilient to
  - Counter watermarking
  - Fine-tuning
  - Training
  - Model inversions



DNN Watermarking



DNN Watermarking

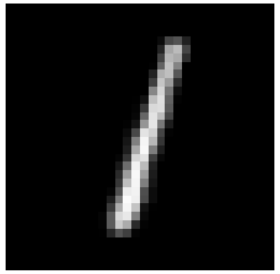


DNN Watermark generation

- Meaningful content embedded in original training data

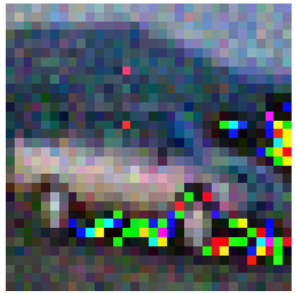
airplane

- Independent training data with unrelated classes as watermarks



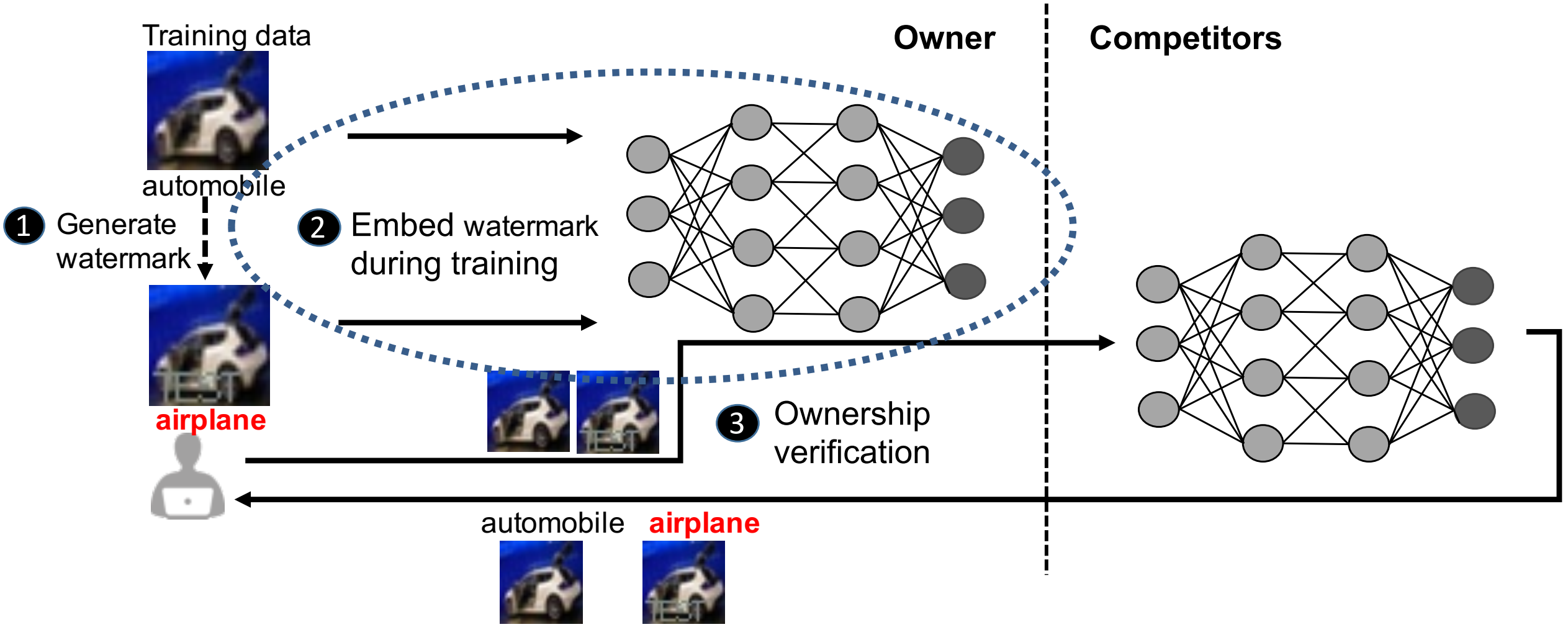
airplane

- Pre-specified Noise as watermark



airplane

DNN Watermarking



DNN watermark embedding

CISPA HELMHOLTZ-ZENTRUM i. G.

Algorithm 1 Watermark embedding

Input:

Training set $D_{train} = \{X_i, Y_i\}_{i=1}^{S}$

DNN key K=$\{Y_s, Y_d\}(s \neq d)$

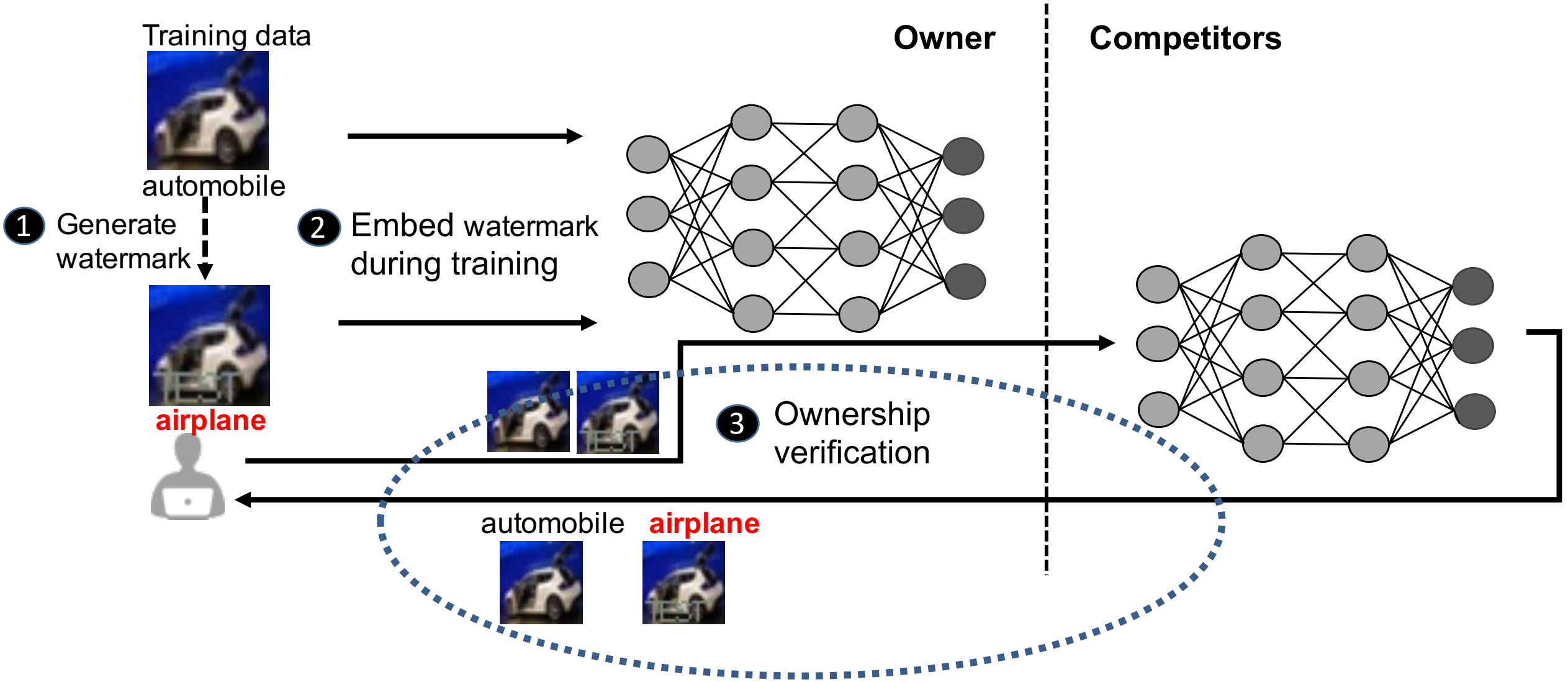
Output:

DNN model: F_θ

Watermark Pair: D_{wm}

function WATERMARK_EMBEDDING()
    $D_{wm} \leftarrow \emptyset$
    $D_{tmp} \leftarrow sample(D_{train}, Y_s, percentage)$
    for each $d \in D_{tmp}$ **do**
        $x_{wm} = ADD\_WATERMARK(d[x], watermarks)$
        $y_{wm} = y_d$
        $D_{wm} = D_{wm} \cup \{x_{wm}, y_{wm}\}$
    end for
end function
$F_\theta = Train(D_{wm}, D_{train})$
return F_θ, D_{wm}

DNN Watermarking



Ownership Verification

- Adversary might want to monetize model with online API
- Query with watermarked images
- If it flips label as trained -> our model

Effectiveness

- Works on trained images (basically overfitting on training set)
- Even works on newly watermarked images (generalization of watermarks to test)

(a) MNIST

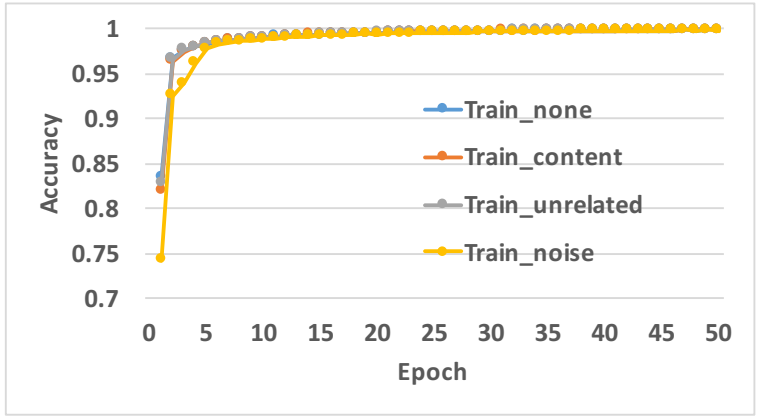
Accuracy	$WM_{content}$	$WM_{unrelated}$	WM_{noise}
Watermarks (trained)	100%	100%	100%
Watermarks (new)	100%	100%	99.42%

(b) CIFAR10

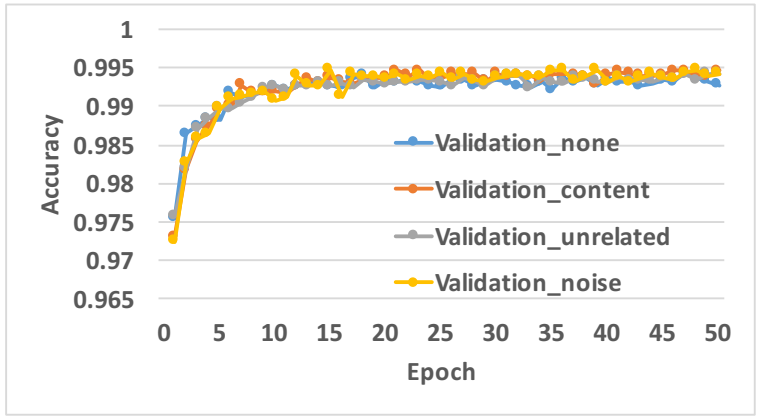
Accuracy	$WM_{content}$	$WM_{unrelated}$	WM_{noise}
Watermarks (trained)	99.93%	100%	99.86%
Watermarks (new)	98.6%	100%	94.1%

Side Effects

- Does including watermarked images effect train/val/test accuracies?

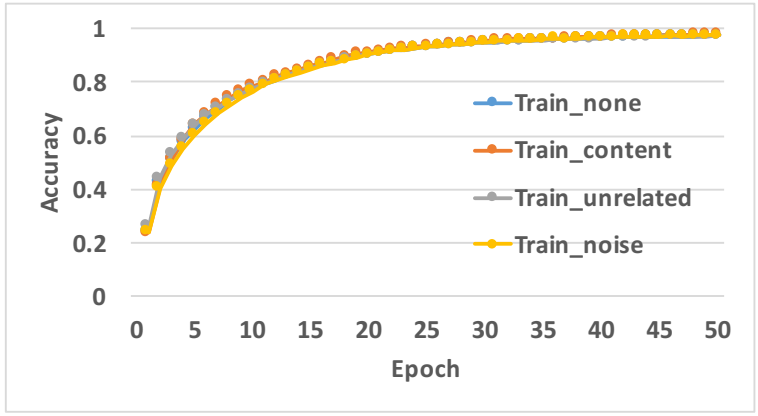


(a) Train accuracy

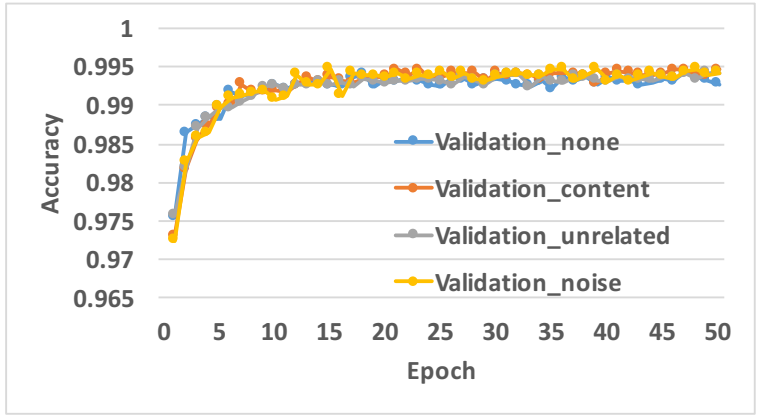


(b) Validation accuracy

Figure 6: Model accuracy over training procedure (MNIST)



(a) Train accuracy



(b) Validation accuracy

(a) MNIST

CleanModel	$WM_{content}$	$WM_{unrelated}$	WM_{noise}
99.28 %	99.46%	99.43%	99.41%

(b) CIFQR10

CleanModel	$WM_{content}$	$WM_{unrelated}$	WM_{noise}
78.6%	78.41%	78.12%	78.49%

Robustness

- Does the model retrain the watermarking – despite modification to model
- Pruning:
  - Remove small weights in model
- Fine-Tuning:
  - Continue training with more examples
- High robustness

Table 3: Robustness for model pruning: accuracy of clean testing data and accuracy of watermarks (MNIST)

Pruning rate	$WM_{content}$		$WM_{unrelated}$		WM_{noise}	
	Testing Acc.	Watermark Acc.	Testing Acc.	Watermark Acc.	Testing Acc.	Watermark Acc.
10%	99.44%	100%	99.43%	100%	99.4%	100%
20%	99.45%	100%	99.45%	100%	99.41%	100%
30%	99.43%	100%	99.41%	100%	99.41%	100%
40%	99.4%	100%	99.31%	100%	99.42%	100%
50%	99.29%	100%	99.19%	100%	99.41%	100%
60%	99.27%	100%	99.24%	100%	99.3%	99.9%
70%	99.18%	100%	98.82%	100%	99.22%	99.9%
80%	98.92%	100%	97.79%	100%	99.04%	99.9%
90%	97.03%	99.95%	93.55%	99.9%	95.19%	99.55%

Table 4: Robustness for model pruning: accuracy of clean testing data and accuracy of watermarks (CIFAR10)

Pruning rate	$WM_{content}$		$WM_{unrelated}$		WM_{noise}	
	Testing Acc.	Watermark Acc.	Testing Acc.	Watermark Acc.	Testing Acc.	Watermark Acc.
10%	78.37%	99.93%	78.06%	100%	78.45%	99.86%
20%	78.42%	99.93%	78.08%	100%	78.5%	99.86%
30%	78.2%	99.93%	78.05%	100%	78.33%	99.93%
40%	78.24%	99.93%	77.78%	100%	78.31%	99.93%
50%	78.16%	99.93%	77.75%	100%	78.02%	99.8%
60%	77.87%	99.86%	77.44%	100%	77.87%	99.6%
70%	76.7%	99.86%	76.71%	100%	77.01%	98.46%
80%	74.59%	99.8%	74.57%	96.39%	73.09%	92.8%
90%	64.9%	99.47%	62.15%	10.93%	59.29%	65.13%

Table 5: Robustness for model fine-tuning: accuracy of clean testing data and accuracy of watermarks

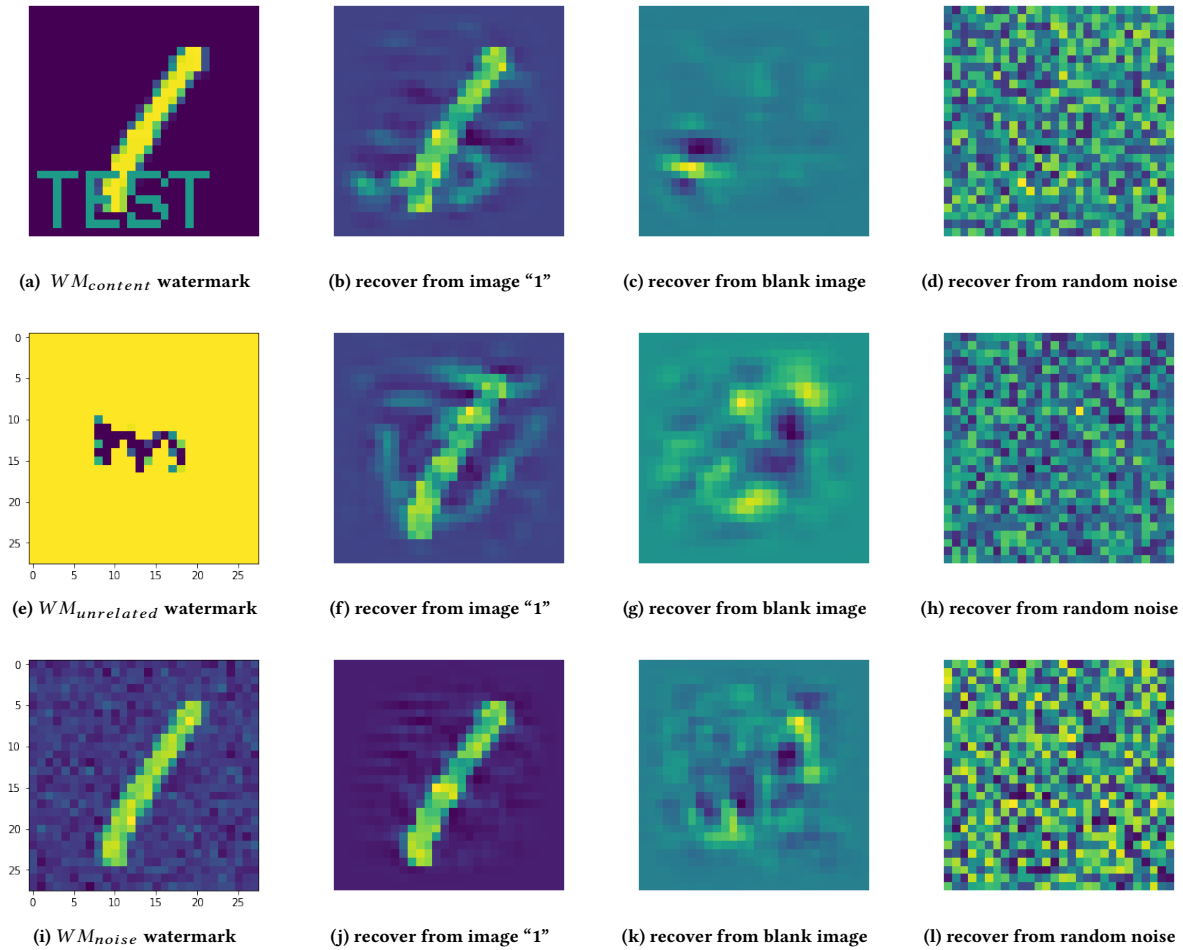
Dataset	$WM_{content}$		$WM_{unrelated}$		WM_{noise}	
	Testing Acc.	Watermark Acc.	Testing Acc.	Watermark Acc.	Testing Acc.	Watermark Acc.
MNIST	99.6%	99.95%	99.64%	100%	99.68%	99.85%
CIFAR10	77.55%	98.33%	76.75%	95.33%	78.43%	69.13%

Security

- Can watermark be recovered from classifier?

- Attack using gradient based technique: Fredrikson, Matt, Somesh Jha, and Thomas Ristenpart. "Model inversion attacks that exploit confidence information and basic countermeasures." In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1322-1333. ACM, 2015.

- Does not see effective



(a) $WM_{content}$ watermark (b) recover from image "1" (c) recover from blank image (d) recover from random noise

(e) $WM_{unrelated}$ watermark (f) recover from image "1" (g) recover from blank image (h) recover from random noise

(i) WM_{noise} watermark (j) recover from image "1" (k) recover from blank image (l) recover from random noise

Machine Learning in Cybersecurity

- Model Stealing & Defense
- Watermarking

Prof. Dr. Mario Fritz | 10.12.2020