Aalto University

UNIVERSITY OF WATERLOO

Extraction of Complex DNN Models: Real Threat or Boogeyman?

N. Asokan

https://asokan.org/asokan/

@nasokan

With Buse Gul Atli and Sebastian Szyller (Joint work with Mika Juuti and Samuel Marchal)

Outline

Is model confidentiality important?

Can models be extracted via their prediction APIs?

What can be done to counter model extraction?

Is model confidentiality important?

Machine learning models: business advantage and intellectual property (IP)

Cost of

- gathering relevant data
- labeling data
- expertise required to choose the right model training method
- resources expended in training

Adversary who steals the model can avoid these costs

Type of model access: white box

White-box access: user

- has physical access to model
- knows its structure
- can observe execution (scientific packages, software on user-owned devices)

How to prevent (white-box) model theft?

White-box model theft can be countered by

- **Computation with encrypted models**
- **Protecting models using secure hardware**
- **Hosting models behind a firewalled cloud service**

Type of model access: black-box

Black-box access: user

- does not have physical access to model
- interacts via a well-defined interface ("prediction API"):
  - directly (translation, image classification)
  - indirectly (recommender systems)

Basic idea: hide the model itself, expose model functionality only via a prediction API

Is that enough to prevent model theft?

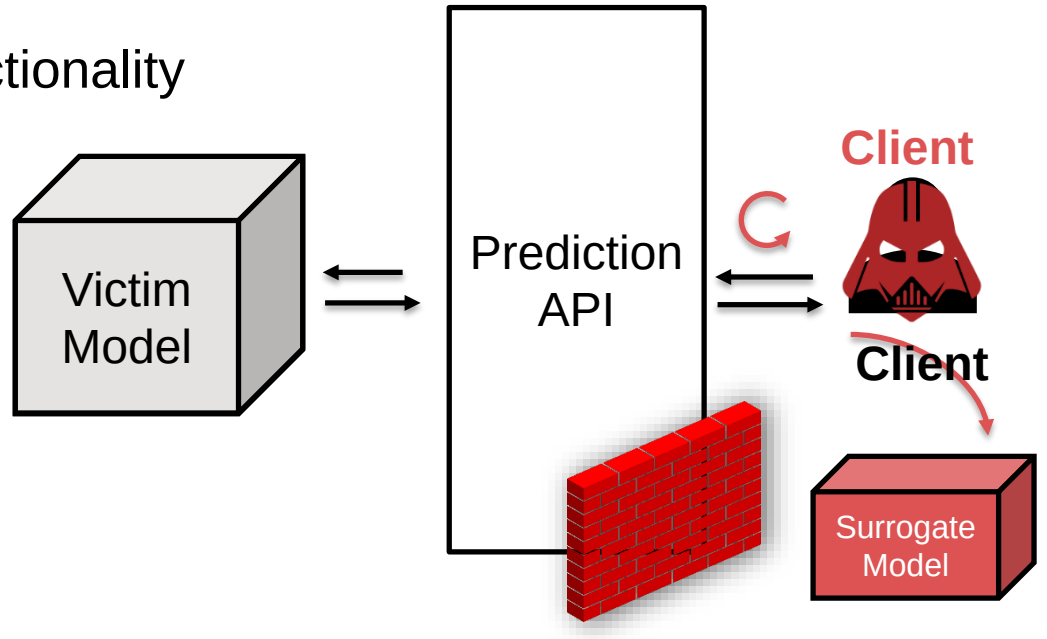
Extracting models via their prediction APIs

Prediction APIs are oracles that leak information

Adversary

- Malicious client
- Goal: construct surrogate model(*) comparable w/ functionality
- Capability: access to prediction API or model outputs

(*) aka “student model” or “imitation model”

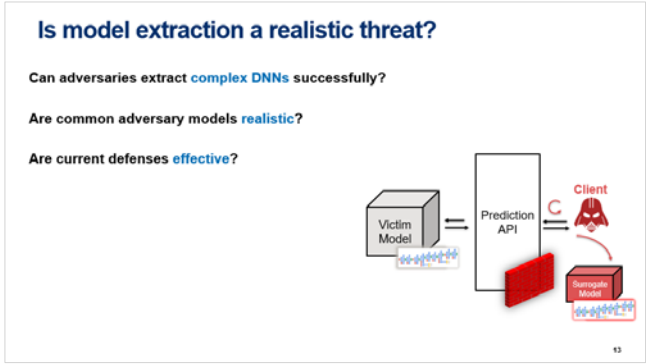


Prior work on extracting

- Logistic regression, decision trees[1]
- Simple CNN models[2]
- Querying API with synthetic samples

[1] Tramèr et al. - *Stealing Machine Learning Models via Prediction APIs*. USENIX SEC ’16 (https://arxiv.org/abs/1609.02943)
[2] Papernot et al. - *Practical Black-Box Attacks against Machine Learning*. ASIACCS ‘17 (https://arxiv.org/abs/1602.02697)

Extracting deep neural networks



Against simple DNN models[1]

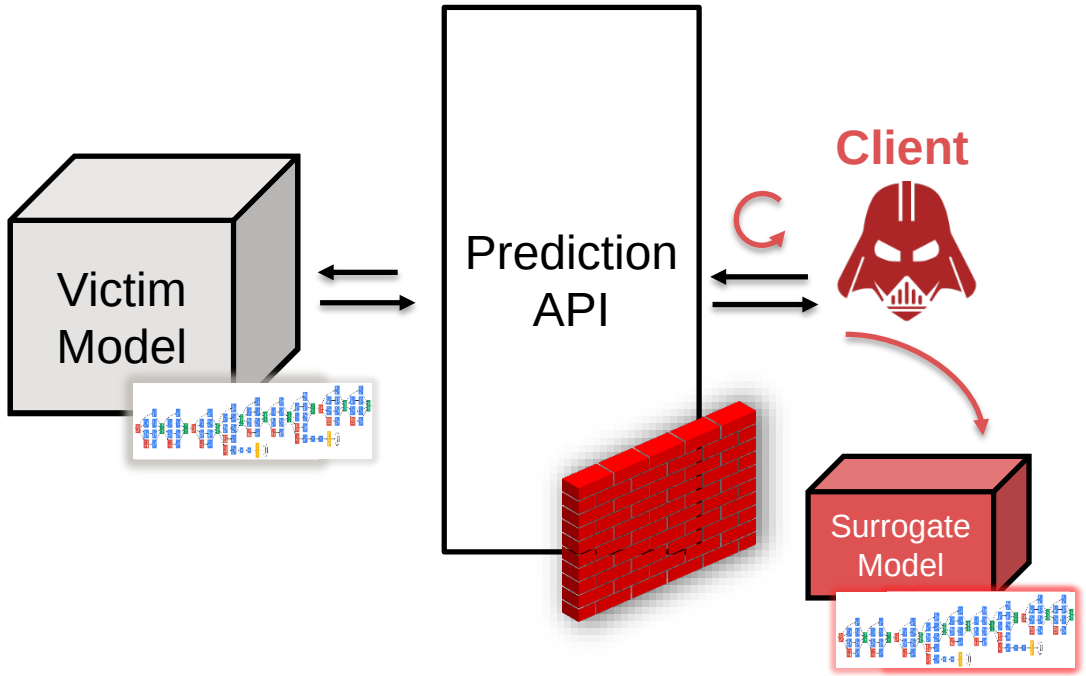
- E.g., MNIST, GTSRB

Adversary

- knows general structure of the model
- has limited natural data from victim's domain

Approach

- Hyperparameters CV-search
- Query using natural data for rough estimate decision boundaries, synthetic data to fine-tune



[1] Juuti et al. - *PRADA: Protecting against DNN Model Stealing Attacks*. EuroS&P '19 (https://arxiv.org/abs/1805.02628)

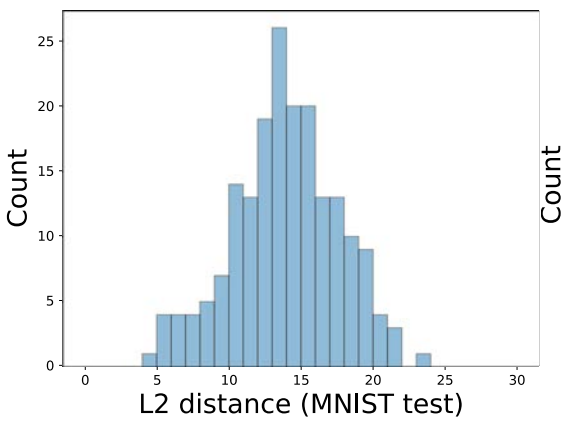
Can model extraction attacks be detected?

Preliminary: distance between random points in a space fits a normal (Gaussian) distribution

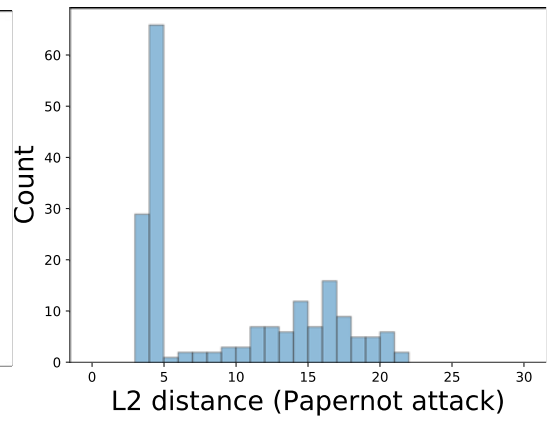
Assumptions

- Benign queries consistently distributed → distances fit a normal distribution
- Adversarial queries focused on a few areas → distances deviate from a normal distribution

Benign

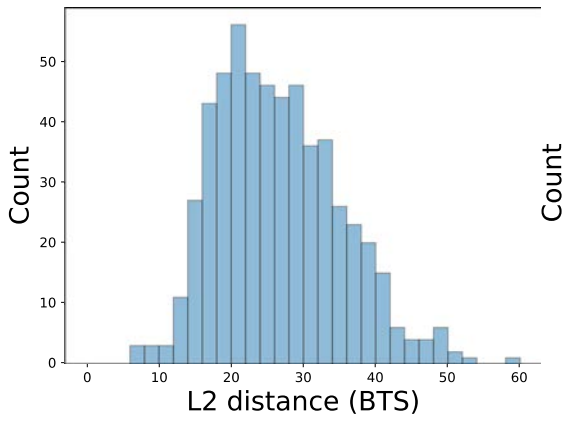


Attack

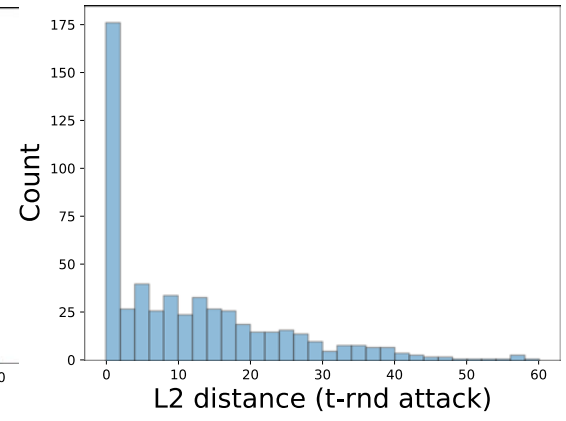


MNIST

Benign



Attack



GTSRB

PRADA defense[1]

Stateful defense

- Focus on low false positives
- Keeps track of queries submitted by a given client
- Detects deviation from a normal distribution

Shapiro-Wilk test as a measure of “novelty” in queries

- Quantify how well a set of samples *D* fits a normal distribution
- Test statistic: $W(D) < \delta \rightarrow$ attack detected
- δ**:** parameter to be defined

[1] Juuti et al. - *PRADA: Protecting against DNN Model Stealing Attacks*. EuroS&P ’19 (https://arxiv.org/abs/1805.02628)

PRADA detection efficiency[1]

Model + δ value	FPR	Queries made until detection		
		Tramer	*Papernot*	*T-rnd*
MNIST ($\delta = 0.96$)	0.0%	5,560	120	130
MNIST ($\delta = 0.95$)	0.0%	5,560	120	140
GTRSB ($\delta = 0.90$)	0.6%	5,020	430	500
GTRSB ($\delta = 0.87$)	0.0%	5,020	430	540

All prior model extraction attacks detected

- Slowest on Tramèr (but ineffective on DNNs, requires ≫ 500k queries to succeed [2])

Detection triggered when queries use synthetic data, infeffective otherwise

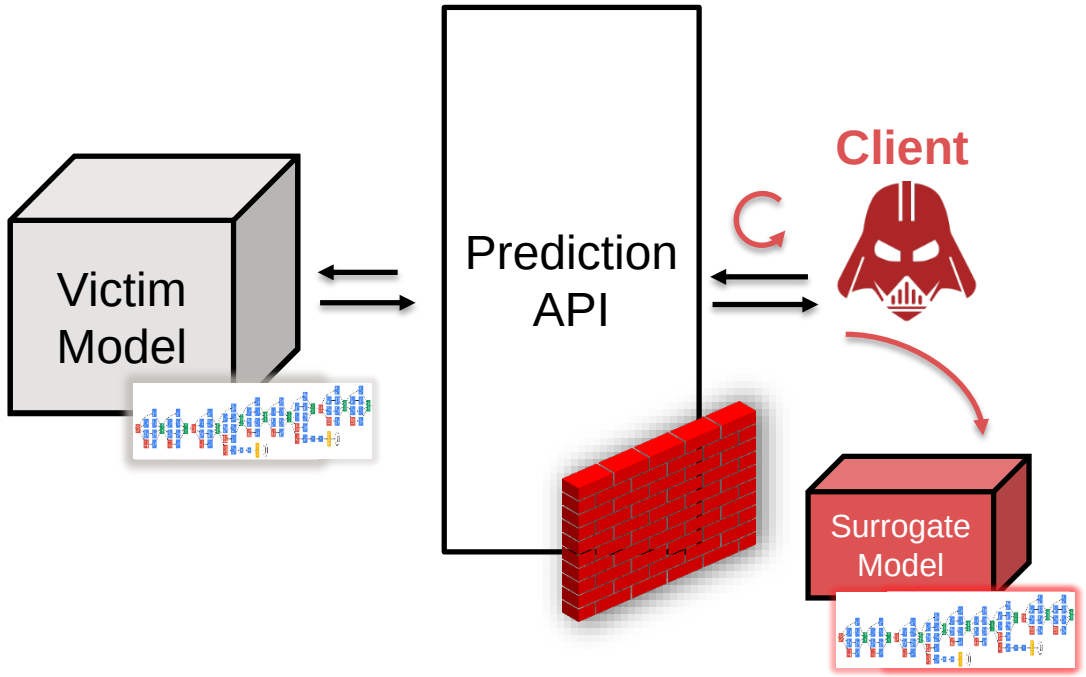
[1] Juuti et al. - *PRADA: Protecting against DNN Model Stealing Attacks*. EuroS&P '19 (https://arxiv.org/abs/1805.02628)
[2] (Optimistic estimate based on) Tramèr et al. *Stealing ML models via prediction APIs*. UsenixSEC'16.
[3] Papernot et al. *Practical black-box attacks against machine learning*. AsiaCCS'17.

Is model extraction a realistic threat?

Can adversaries extract complex DNNs successfully?

Are common adversary models realistic?

Are current defenses effective?



Extraction of Complex DNN Models: Knockoff nets[1]

Goal:

- Build a surrogate model that
  - steals model functionality of victim model
  - performs similarly on the same task with high classification accuracy

Adversary capabilities:

- Victim model knowledge:
  - None of train/test data, model internals, output semantics
  - Access to full prediction probability vector
- Access to natural samples, not (necessarily) from the same distribution as train/test data
- Access to pre-trained high-capacity model

[1] Orekondy et al. - *Knockoff Nets: Stealing Functionality of Black-Box Models.* CVPR '19 (https://arxiv.org/abs/1812.02766)

Knockoff Nets: systematic empirical analysis

Buse Gul Atli
Doctoral candidate
https://people.aalto.fi/buse.atlitekgul

Knockoff nets: Our Goals and Contributions

Reproduce empirical evaluation of Knockoff nets[1] to confirm its effectiveness

Introduce a defense within the adversary model in [1] to detect attacker's queries

Revisit adversary model in [1]

- Explore impact of a more realistic adversary model on attack and defense effectiveness
  - Attack effectiveness decreases: Different surrogate-victim architectures, reduced granularity of victim's prediction API's output, reduced diversity of adversarial queries
  - Defense effectiveness decreases: Attacker has natural samples distributed like victim's training data

[1] Orekondy et al. - *Knockoff Nets: Stealing Functionality of Black-Box Models.* CVPR '19 (https://arxiv.org/abs/1812.02766)

Knockoff nets[1] : Experimental Setup

Victim derived from public, pre-trained, high-capacity model (e.g., ResNet-34 on ImageNet)

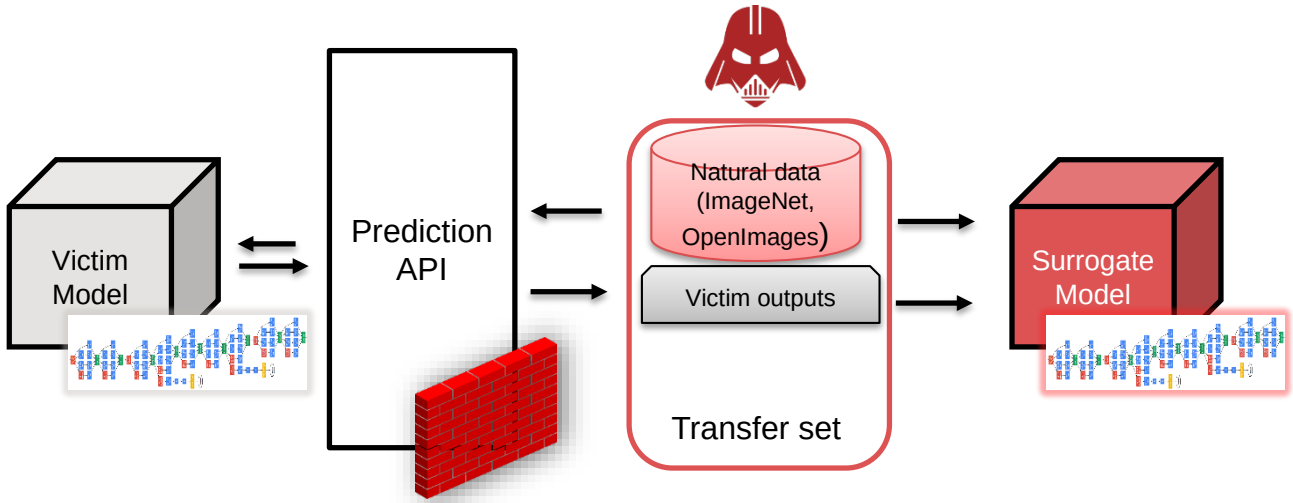
Strategy

Collect unlabeled natural data

- From the same domain (e.g. images)
- Out of target train/test distribution

Query API to collect victim outputs

- Using ~ 100,000 queries
- API returns probability vector



Construct surrogate model

- Select a pre-trained model and fine-tune it with transfer set
- Takes ~ 3 days (Tesla V100 GPU, 10 GB; estimated cost $120-$170)

[1] Orekondy et al. - *Knockoff Nets: Stealing Functionality of Black-Box Models,* CVPR '19 (https://arxiv.org/abs/1812.02766)

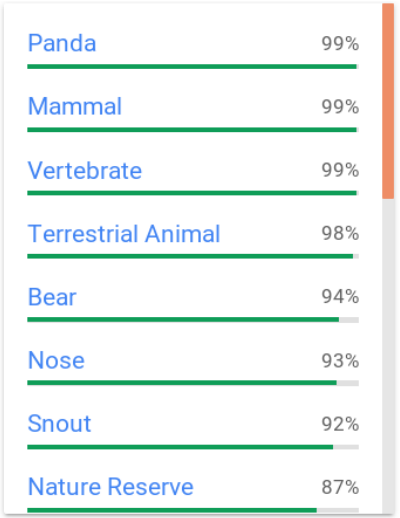
Knockoff nets: Reproduction

Knockoff nets are effective against complex, pre-trained DNN models

Victim Model (Dataset-model)	Test Accuracy % (performance recovery)			
	Our reproduction		Reported in [1]	
	Victim Model	Surrogate Model	Victim Model	Surrogate Model
Caltech-RN34	74.1	72.2 (0.97x)	78.8	75.4 (0.96x)
CUBS-RN34	77.2	70.9 (0.91x)	76.5	68.0 (0.89x)
Diabetic-RN34	71.1	53.5 (0.75x)	58.1	47.7 (0.82x)
GTSRB-RN34	98.1	94.8 (0.96x)	-	-
CIFAR10-RN34	94.6	88.2 (0.93x)	-	-

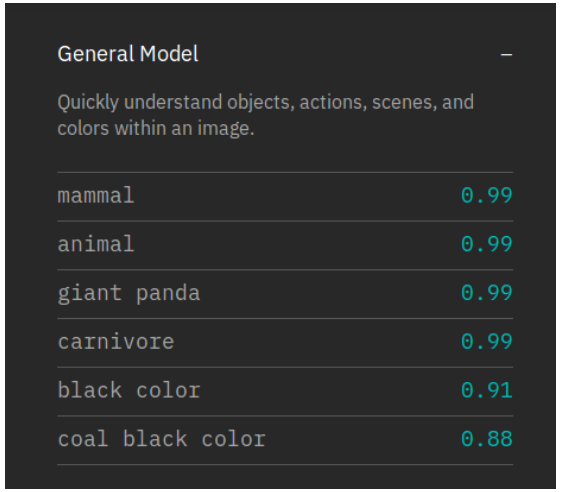
[1] Orekondy et al. - *Knockoff Nets: Stealing Functionality of Black-Box Models.* CVPR '19 (https://arxiv.org/abs/1812.02766)

Revisiting the Adversary Model: Reduced Granularity of Prediction API's Output



Google Cloud Vision (top 20)

Clarifai (top 20)



IBM Watson (top 10)

Revisiting the Adversary Model: Reduced Granularity of Prediction API's Output

Original adversary model in [1] expects a complete prediction vector for each query

Effectiveness degrades when prediction API gives truncated results (top label, rounded probabilities etc.)

Victim Model (Dataset-model)	Test Accuracy % (performance recovery)		
	Victim Model	Surrogate Model (full probability vector)	Surrogate Model (only top label)
Caltech-RN34 (257 classes)	74.1	72.2 (0.97x)	57.2 (0.77x)
CUBS-RN34 (200 classes)	77.2	70.9 (0.91x)	42.5 (0.55x)
Diabetic-RN34 (5 classes)	71.1	53.5 (0.75x)	53.5 (0.75x)
GTSRB-RN34 (43 classes)	98.1	94.8 (0.96x)	91.9 (0.93x)
CIFAR10-RN34 (10 classes)	94.6	88.2 (0.93x)	84.4 (0.89x)

[1] Orekondy et al. - *Knockoff Nets: Stealing Functionality of Black-Box Models,* CVPR '19 (https://arxiv.org/abs/1812.02766)

Revisiting the Adversary Model: Different Surrogate-Victim Architectures

Adversary model in [1] : victim model uses publicly available, pre-trained DNNs.

Effectiveness degrades when victim is not based on pre-trained DNNs.

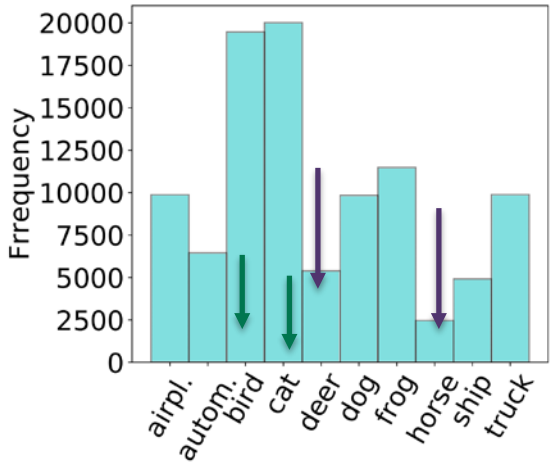
Victim Model (Dataset-model)	Test Accuracy % (performance recovery)		
	Victim Model	Surrogate Model (RN34)	Surrogate Model (VGG16)
GTSRB-RN34	98.1	94.8 (0.96x)	90.1 (0.91x)
CIFAR10-RN34	94.6	88.2 (0.93x)	82.9 (0.87x)
GTSRB-5L	91.5	54.5 (0.59x)	55.8 (0.60x)
CIFAR10-9L	84.5	67.5 (0.79x)	64.7(0.76x)

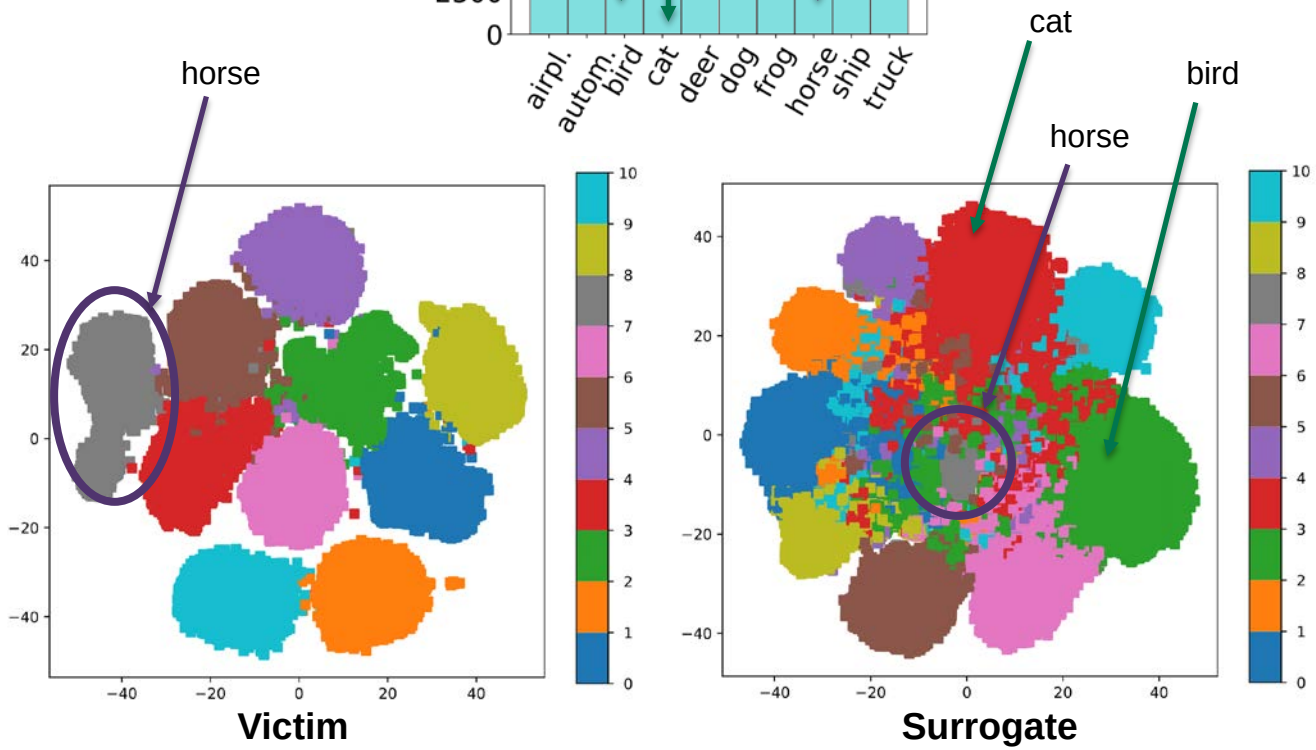
[1] Orekondy et al. - *Knockoff Nets: Stealing Functionality of Black-Box Models,* CVPR '19 (https://arxiv.org/abs/1812.02766)

Knockoff nets: Limitation

Knockoff nets cannot recover per-class performance of victim model

Class Name	Test accuracy % (performance recovery)	
	Victim Model (CIFAR-RN34) 94.6% on average	Surrogate Model 88.2% on average
Airplane (class 0)	95	88 (0.92x)
Automobile (class 1)	97	95 (0.97x)
Bird (class 2)	92	87 (0.94x)
Cat (class 3)	89	86 (0.96x)
Deer (class 4)	95	84 (0.88x)
Dog (class 5)	88	84 (0.95x)
Frog (class 6)	97	90 (0.92x)
Horse (class 7)	96	79 (0.82x)
Ship (class 8)	96	92 (0.95x)
Truck (class 9)	96	92 (0.95x)





Knockoff Nets: detection

Sebastian Szyller
Doctoral candidate
@sebszyller

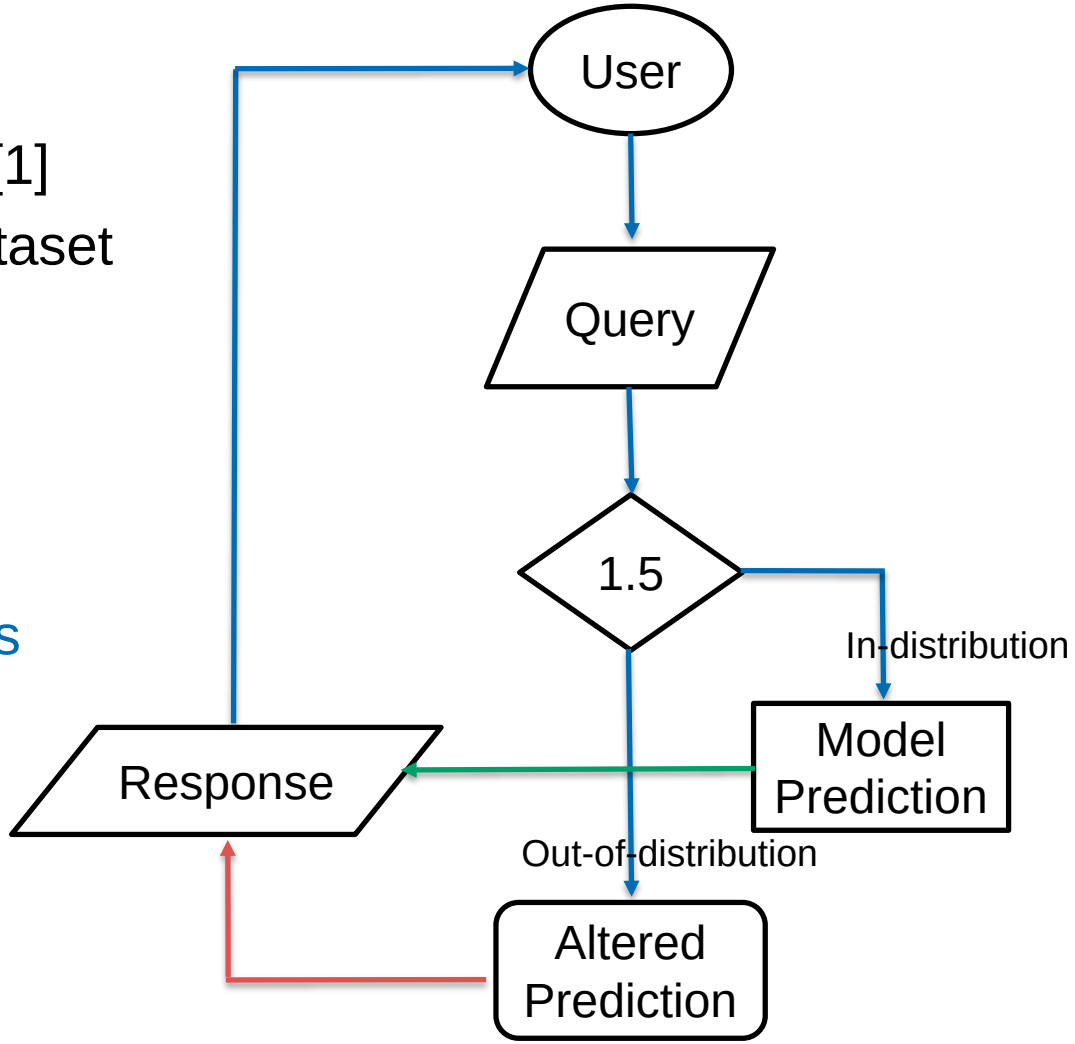
Knockoff nets: Detecting Attacker's Queries

Motivation

- Adversary is unaware of target distribution or task [1]
- Queries API with a random subset of public dataset used for a general task

Design

- Binary pre-classifier for incoming queries (1.5)
- Detect images from distribution other than victim's
- Give proper prediction only to in-distribution queries



[1] Orekondy et al. - *Knockoff Nets: Stealing Functionality of Black-Box Models.* CVPR '19 (https://arxiv.org/abs/1812.02766)

Knockoff nets: Detecting Attacker's Queries

Evaluation

- Trained ResNet classifiers to detect in and out-of-distribution queries
- High TPR/TNR on all datasets but Caltech (strong overlap with ImageNet, OpenImages)
- Performs better than state-of-the-art out-of-distribution methods (ODIN[1] , Mahal[2])

Victim Model (Dataset-model)	ImageNet		OpenImages	
	In-distribution (TPR%)	Out-of-distribution (TNR%)	In-distribution (TPR%)	Out-of-distribution (TNR%)
Caltech-RN34	63	56	61	59
CUBS-RN34	93	93	93	93
Diabetic-RN34	99	99	99	99
GTSRB-RN34	99	99	99	99
CIFAR10-RN34	96	96	96	96

[1] Liang et al. – *Enhancing the Reliability of Out-of-Distribution Image Detection in Neural Networks*. ICLR '18 (https://arxiv.org/abs/1706.02690)
[2] Lee et al. - *A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks.* NIPS' 18 (https://arxiv.org/abs/1807.03888)

Revisiting the Adversary Model: Access to In-distribution Data

The larger the overlap between attacker's transfer set and victim's training data, the less effective the detection.

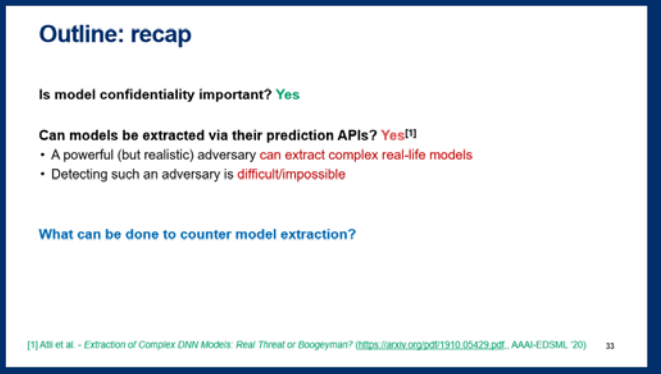
A more realistic adversary

- Has access to more (unlimited) data (public databases, search engines)
- Has approximate knowledge of prediction APIs task (food, faces, birds etc.)
- Can evade detection mechanisms identifying out-of-distribution queries

Are there any prevention mechanisms?

- Stateful analysis → Sybil attacks
- Charging customers upfront → Reduced utility for benign users
- Restrict access to the API → Reduced utility for benign users
- Slow down the attacker[1] → Does not thwart a well-resourced attacker

[1] Orekondy et al. – *Prediction Poisoning: Towards Defenses Against DNN Model Stealing Attacks*. ICLR ' 20 (https://arxiv.org/abs/1906.10908)



Extracting other types of models

Extracting NLP Transformer models

Techniques for extracting image classifiers don't always extend to NLP models

Transfer learning from pre-trained models is now very popular

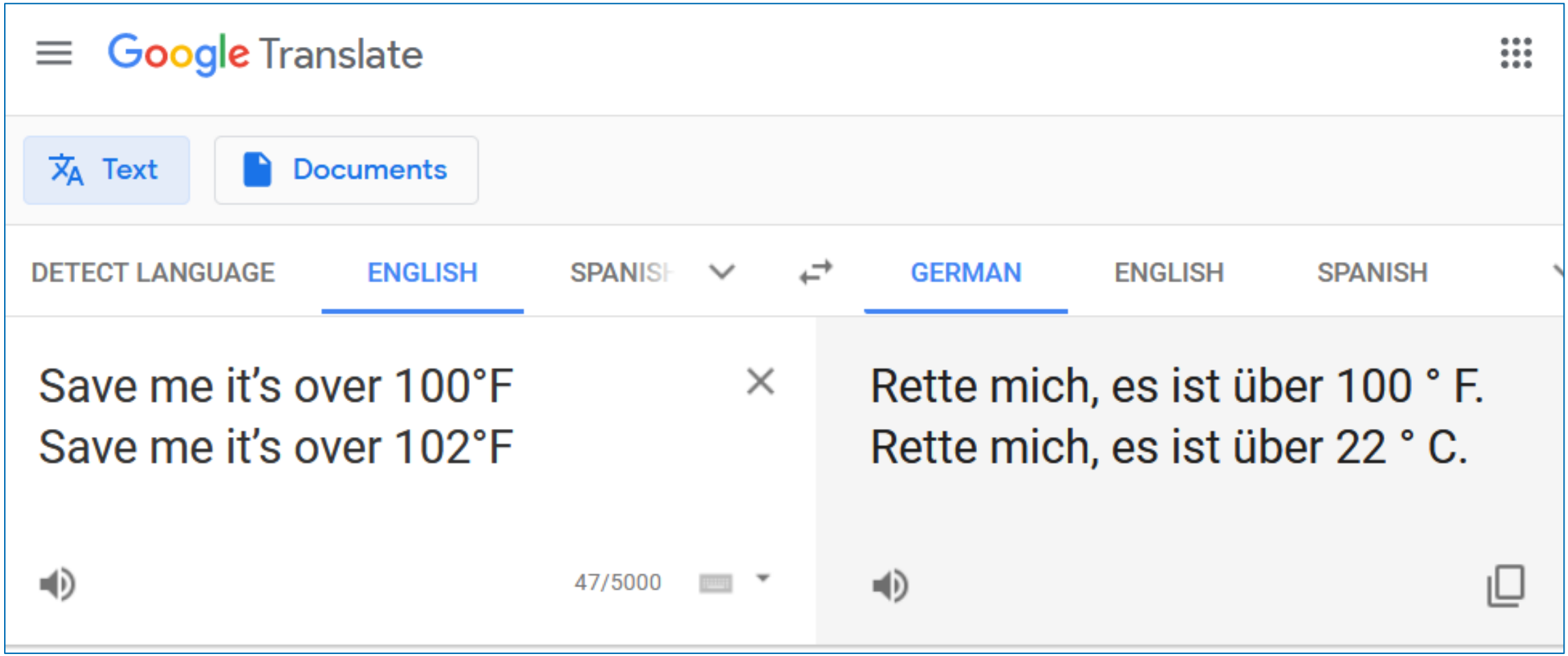
- But they make model extraction easier[1]

Krishna et al[1] show that a Knockoff-like attacks against BERT models are feasible

- Adversary unaware of target distribution or task of victim model
- Adversary queries are merely "natural" (randomly sampled sequences of words)
- In-distribution adversary queries can improve extraction efficacy

Wallace et al[2] extract real-world MT models, find transferable adversarial examples

[1] Krishna et al. – *Thieves on Sesame Street! Model Extraction of BERT-based APIs* . ICLR '20 (https://iclr.cc/virtual_2020/poster_Byl5NREFDr.html)
[2] Wallace et al. – *Imitation Attacks and Defenses for Black-box Machine Translation Systems.* EMNLP '20 (https://arxiv.org/abs/2004.15015)



https://translate.google.com/#view=home&op=translate&sl=en&tl=de&text=Save%20me%20it%E2%80%99s%20over%20100%C2%B0F%0ASave%20me%20it%E2%80%99s%20over%20102%C2%B0F

[1] Wallace et al. – *Imitation Attacks and Defenses for Black-box Machine Translation Systems.* EMNLP '20 (https://arxiv.org/abs/2004.15015)

Extracting reinforcement-learning models

Extracting reinforcement-learning models is harder[1] because they are

- more complex and deeper models (?)
- less observable: only actions (e.g., no prediction confidence scores)
- stochastic: a DRL policy is a Markov decision process

Chen et al[1]

- learn victim's algorithm: train shadow models with candidate algorithms, generate action sequences and train a classifier, use classifier on victim's action sequence
- Use imitation learning to refine the chosen algorithm

[1] Chen et al. - *Stealing Deep Reinforcement Learning Models for Fun and Profit (*https://arxiv.org/abs/1807.03888*)*

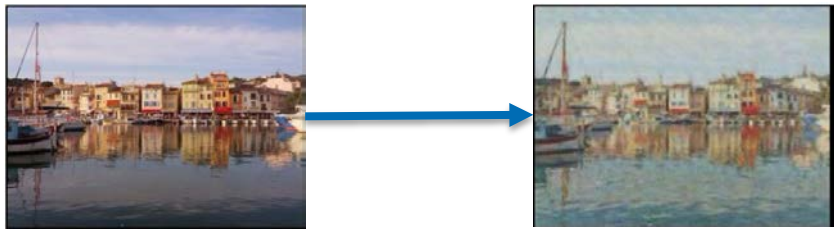
Extracting Style-transfer models

GANS are effective for changing image style

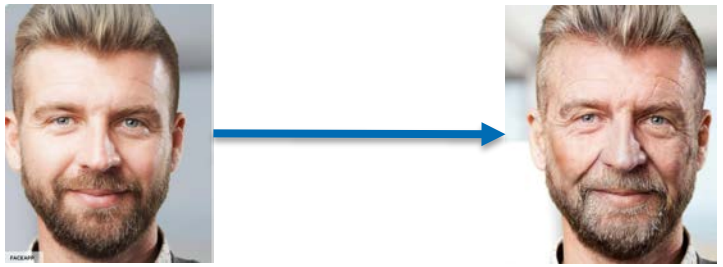
- coloring, face filters, style application

Core feature in generative art and in social media apps

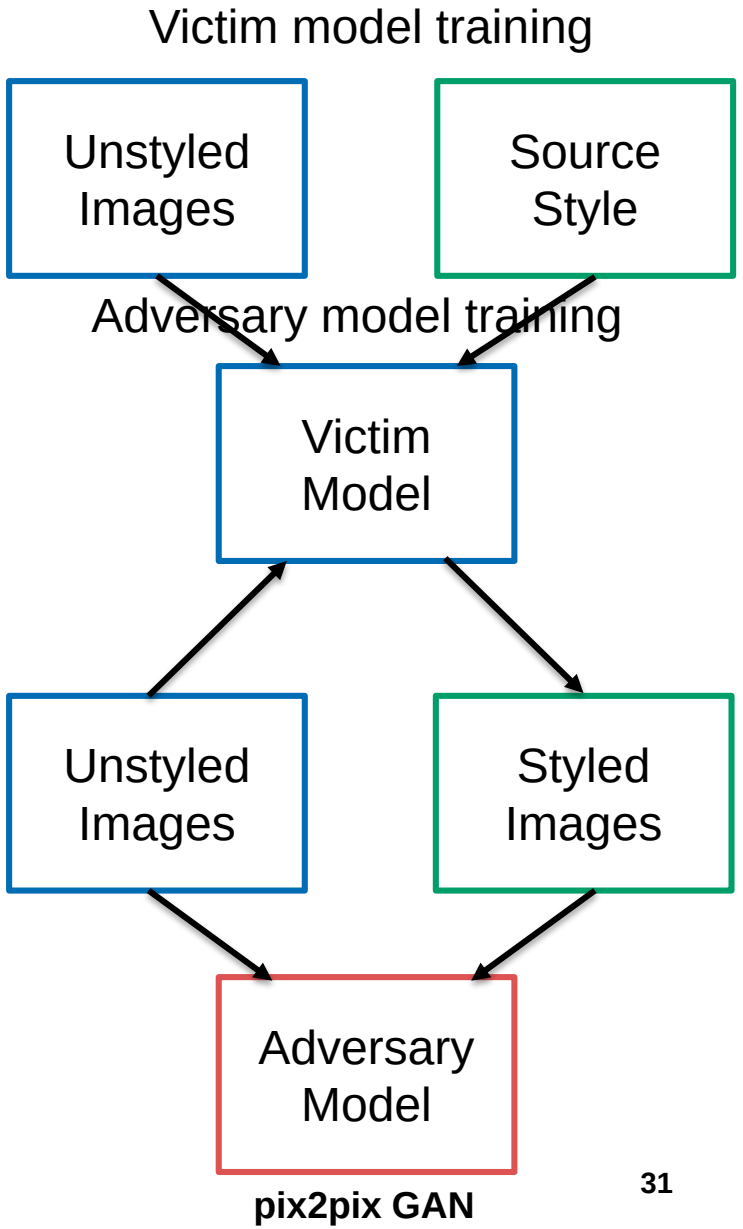
- Selfie2Anime, FaceApp



CycleGANs



FaceApp



pix2pix GAN

Goal: Apply Monet style

Szyller et. al. work in progress

Original (unstyled)

Styled (victim)

Styled (ours)

Outline: recap

Is model confidentiality important? Yes

Can models be extracted via their prediction APIs? Yes[1]

- A powerful (but realistic) adversary can extract complex real-life models
- Detecting such an adversary is difficult/impossible

What can be done to counter model extraction?

[1] Atli et al. - *Extraction of Complex DNN Models: Real Threat or Boogeyman?* (https://arxiv.org/pdf/1910.05429.pdf,, AAAI-EDSML '20)

Existing Watermarking of DNNs[1]

Watermark embedding:

- Embed watermark in model during training:
  - Train model using training data + trigger set (specific labels to a set of selected samples),

Verification of ownership:

- Requires adversary to publicly expose stolen model
- Query model with trigger set, verify watermark (predictions match trigger set labels)

Limitations:[2]

- Protects only against **physical theft** of model
- Model extraction attacks steal model **without watermark**

[1] Yadi et al. - *Watermarking Deep Neural Networks by Backdooring*. USENIX SEC '18 (https://www.usenix.org/node/217594)
[2] Szyller et. al. - *DAWN: Dynamic Adversarial Watermarking of Neural Networks.* In submission. (https://arxiv.org/abs/1906.00830)

DAWN: Dynamic Adversarial Watermarking of DNNs[1]

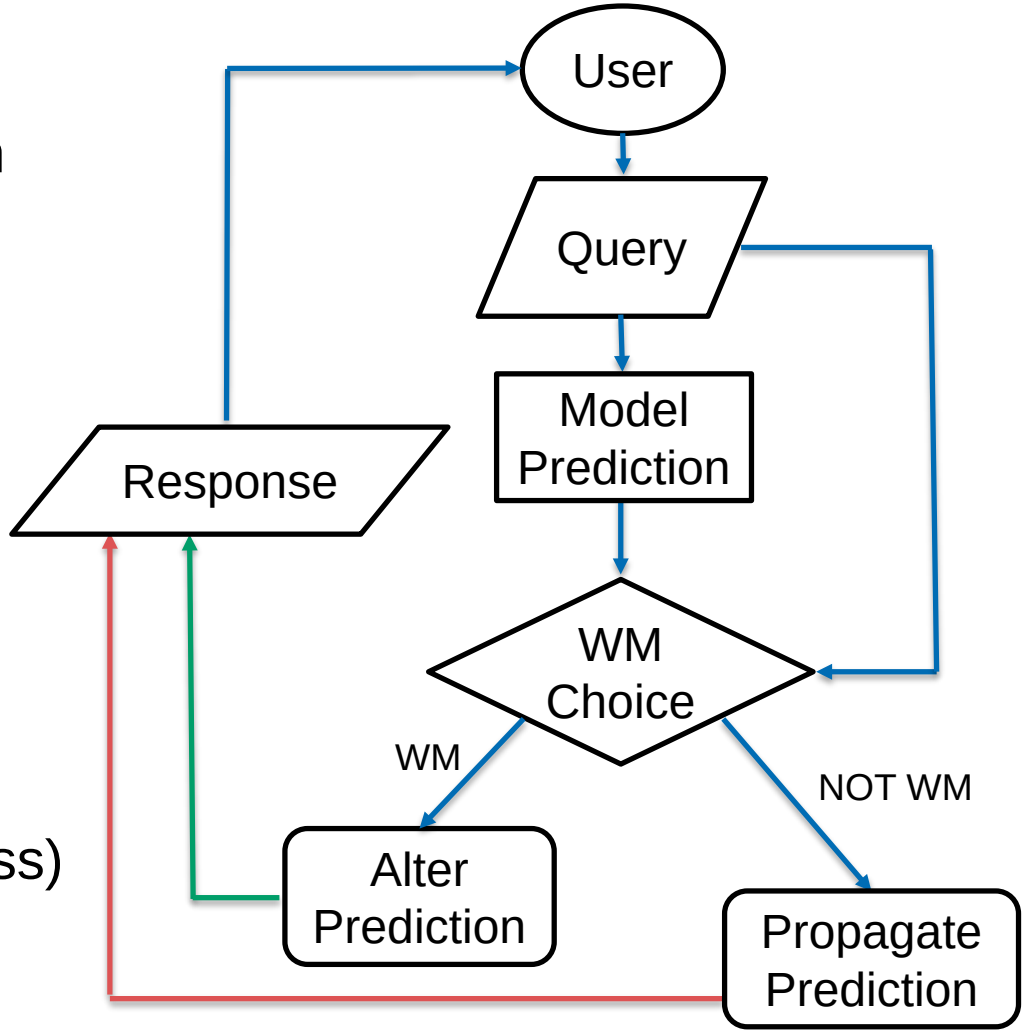
Goal: Watermark models obtained via model extraction

Our approach:

- Implemented as part of the **prediction API**
- Return **incorrect predictions** for several samples
- Adversary forced to embed watermark while training

Watermarking evaluation:

- **Unremovable** and **indistinguishable**
- **Defend against** *PRADA*[2] and *KnockOff* [3]
- Preserve victim *model utility* (*0.03-0.5%* accuracy loss)



[1] Szyller et. al. - *DAWN: Dynamic Adversarial Watermarking of Neural Networks.* In submission. (https://arxiv.org/abs/1906.00830)
[2] Juuti et al. - *PRADA: Protecting against DNN Model Stealing Attacks*. EuroS&P '19 (https://arxiv.org/abs/1805.02628)
[3] Orekondy et al. - *Knockoff Nets: Stealing Functionality of Black-Box Models*. CVPR '19 (https://arxiv.org/abs/1812.02766)

Reliable demonstration of ownership in DAWN[1]

Model owner registers its model and watermarks online (timestamped)

Assumption: Adversary makes its model available online

Model owner claims ownership by asking judge to verify watermark

Adversary may attempt to register the stolen model with its own watermarks:

- Timestamping helps resolve which model is legitimate
- Probability of a random and registered watermark passing verification is negligible
  - with confidence $1 - 2^{-64}$

[1] Szyller et. al. - *DAWN: Dynamic Adversarial Watermarking of Neural Networks.* In submission. (https://arxiv.org/abs/1906.00830)

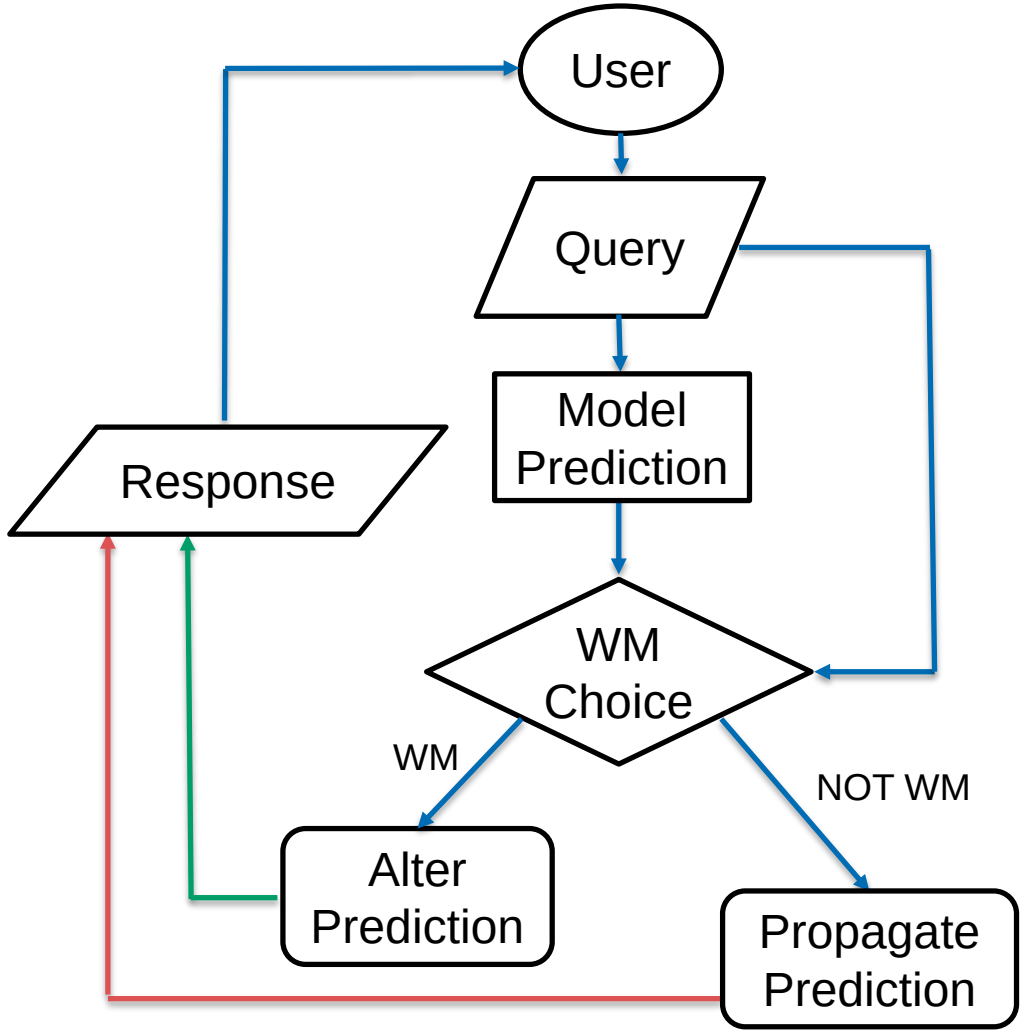
Open issues in DAWN[1]

Indistinguishability

- existence of a robust mapping function (for WM choice)

Unremovability

- “double-stealing” can remove watermark (but impacts accuracy of surrogate model)
- adversary can try to return incorrect predictions on training data (but can be overcome)



[1] Szyller et. al. - *DAWN: Dynamic Adversarial Watermarking of Neural Networks.* In submission. (https://arxiv.org/abs/1906.00830)

Takeaways

Is model confidentiality important? Yes

- **models constitute business advantage to model owners**

Can models be extracted via their prediction APIs? Yes

- **Protecting model data via cryptography or hardware security is insufficient**

What can be done to counter model extraction? Watermarking as a deterrence

- **Watermarking at the prediction API is feasible, open issues remain**
- **Deserves to be considered as a deterrence against model stealing**

More on our security + ML research at https://ssg.aalto.fi/research/projects/mlsec/model-extraction/