



Pattern Recognition
and Applications Lab



University of
Cagliari, Italy



Reliable Evaluation and Benchmarking of Machine Learning Models

Maura Pintor

Assistant Professor @ University of Cagliari (Italy)

maura.pintor@unica.it

Attacks against AI are Pervasive!



Sharif et al., *Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition*, ACM CCS 2016



Eykholt et al., *Robust physical-world attacks on deep learning visual classification*, CVPR 2018



"without the dataset the article is useless"

"okay google browse to evil dot com"

Carlini and Wagner, *Audio adversarial examples: Targeted attacks on speech-to-text*, DLS 2018 https://nicholas.carlini.com/code/audio_adversarial_examples/



- Demetrio, Biggio, Roli et al., *Adversarial EXEmple*s: ..., ACM TOPS 2021
- Demetrio, Biggio, Roli et al., *Functionality-preserving black-box optimization of adversarial windows malware*, IEEE TIFS 2021
- Demontis, Biggio, Roli et al., *Yes, Machine Learning Can Be More Secure!...*, IEEE TDSC 2019

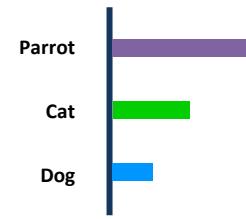
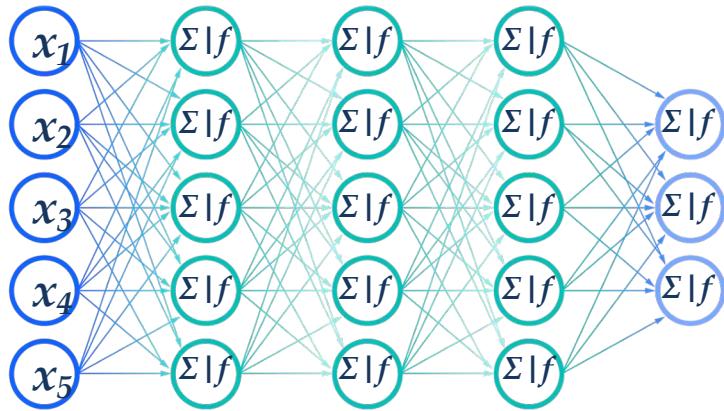
Attacks against Machine Learning

Attacker's Goal			
Attacker's Capability	Integrity	Availability	Privacy / Confidentiality
Test data	Evasion (a.k.a. adversarial examples)	Sponge Attacks	<i>Model extraction / stealing Model inversion (hill climbing) Membership inference</i>
Training data	<i>Backdoor poisoning (to allow subsequent intrusions) – e.g., backdoors or neural trojans</i>	<i>DoS poisoning (to maximize classification error)</i>	-

Attacker's Knowledge:

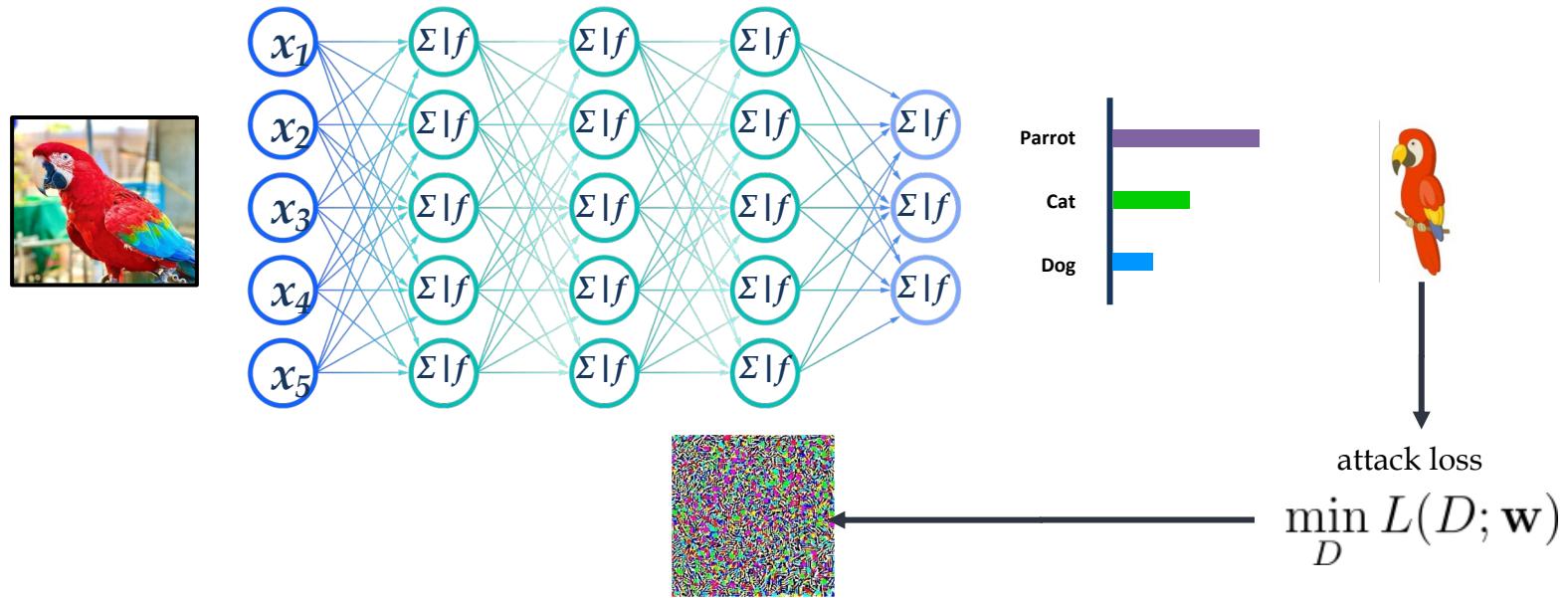
- perfect-knowledge (PK) white-box attacks
- limited-knowledge (LK) black-box attacks (*transferability* with surrogate/substitute learning models)

Adversarial Examples (AdvX)

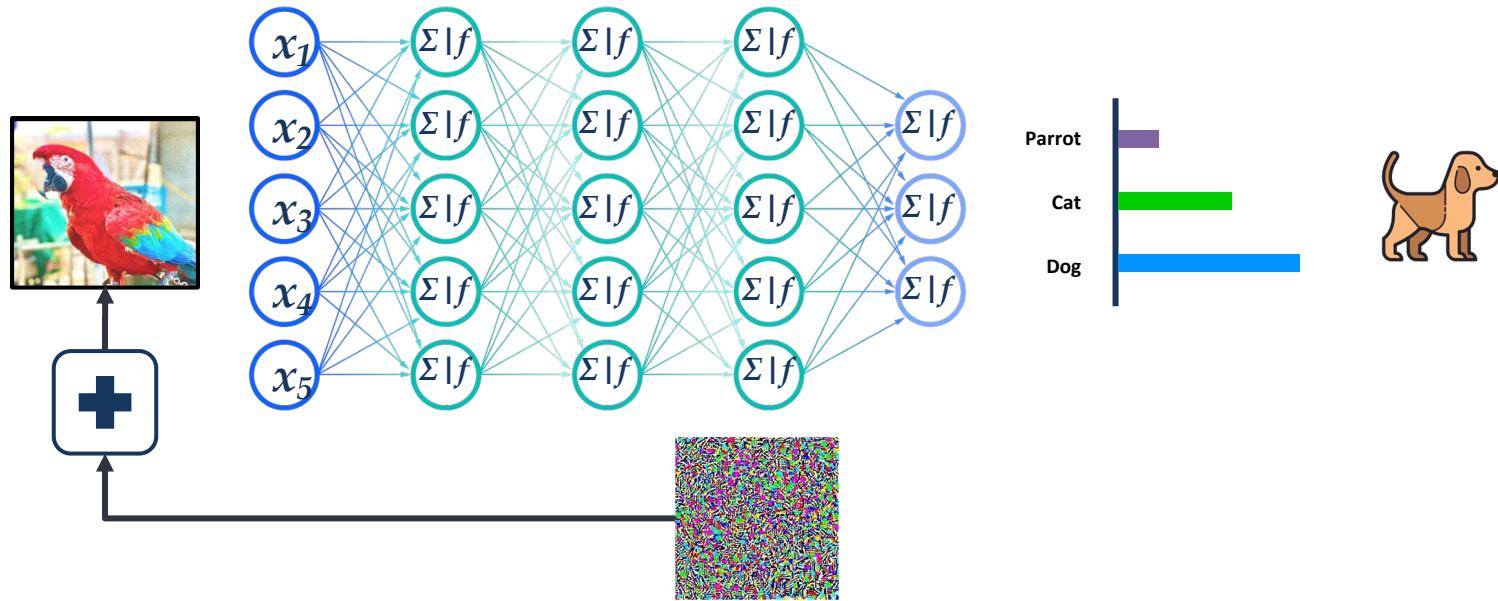


$$\min_{\mathbf{w}} L(D; \mathbf{w})$$

Adversarial Examples (AdvX)



Adversarial Examples (AdvX)



How to craft AdvXs

Exhaustive search → not possible for modern deep learning models

Empirical evaluation → attack = optimization problem + solving algorithm

$$\begin{aligned}\boldsymbol{\delta}^* \in \arg \min_{\boldsymbol{\delta}} \quad & \mathcal{L}(\mathbf{x} + \boldsymbol{\delta}, y, \boldsymbol{\theta}) \\ \text{s.t.} \quad & \|\boldsymbol{\delta}\|_p \leq \epsilon \\ & \mathbf{x}_{lb} \preceq \mathbf{x} + \boldsymbol{\delta} \preceq \mathbf{x}_{ub}\end{aligned}$$

Optimize model's confidence on bad decision
keeping perturbation small
and respecting feature space constraints

How to craft AdvXs

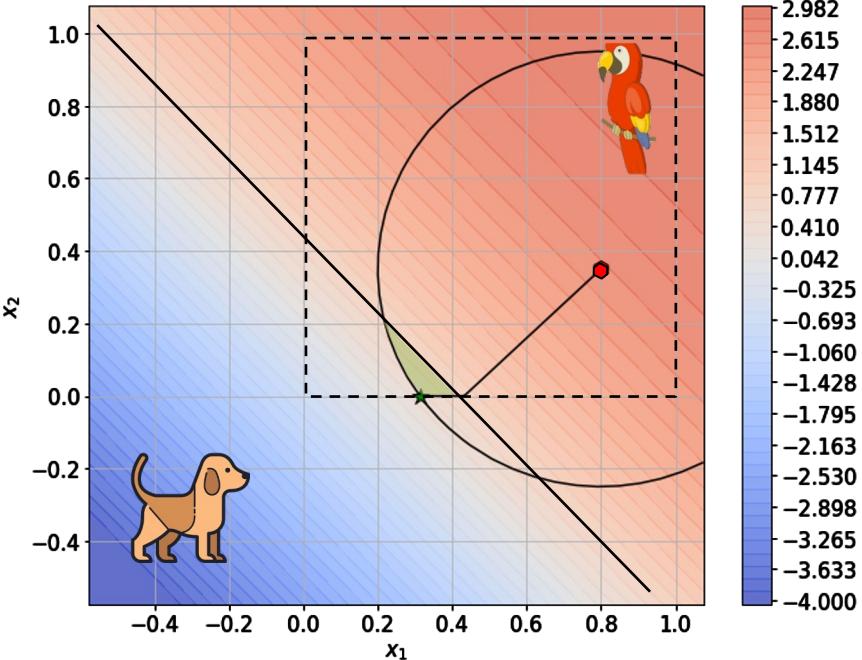
Exhaustive search → not possible for modern deep learning models

Empirical evaluation → attack = optimization problem + solving algorithm

$$\begin{aligned}\delta^* \in \arg \min_{\delta} \quad & \mathcal{L}(x + \delta, y, \theta) \\ \text{s.t.} \quad & \|\delta\|_p \leq \epsilon \\ & x_{lb} \preceq x + \delta \preceq x_{ub}\end{aligned}$$

Optimize model's confidence on bad decision 
keeping perturbation small 
and respecting feature space constraints 

Robust Accuracy = accuracy under worst-case perturbation (fixed perturbation size)

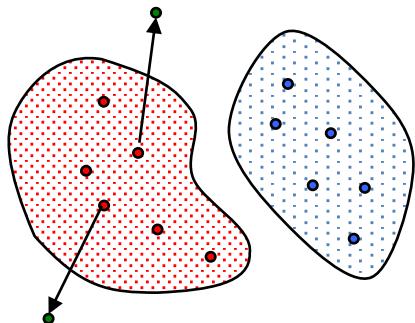


Defending against AdvXs

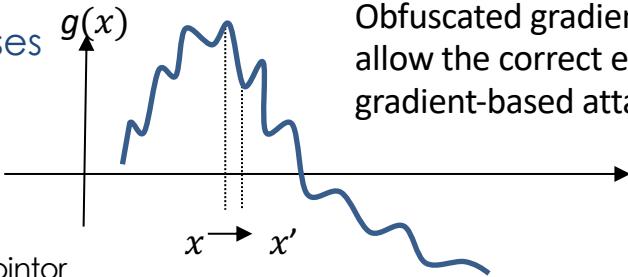
- Robust training (a.k.a. Adversarial training)

$$\min_w \max_{\|\delta_i\|_\infty \leq \epsilon} \sum_i \ell(y_i, f_w(x_i + \delta_i))$$

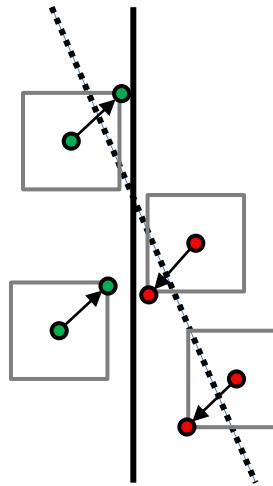
- Detectors



- Ineffective defenses



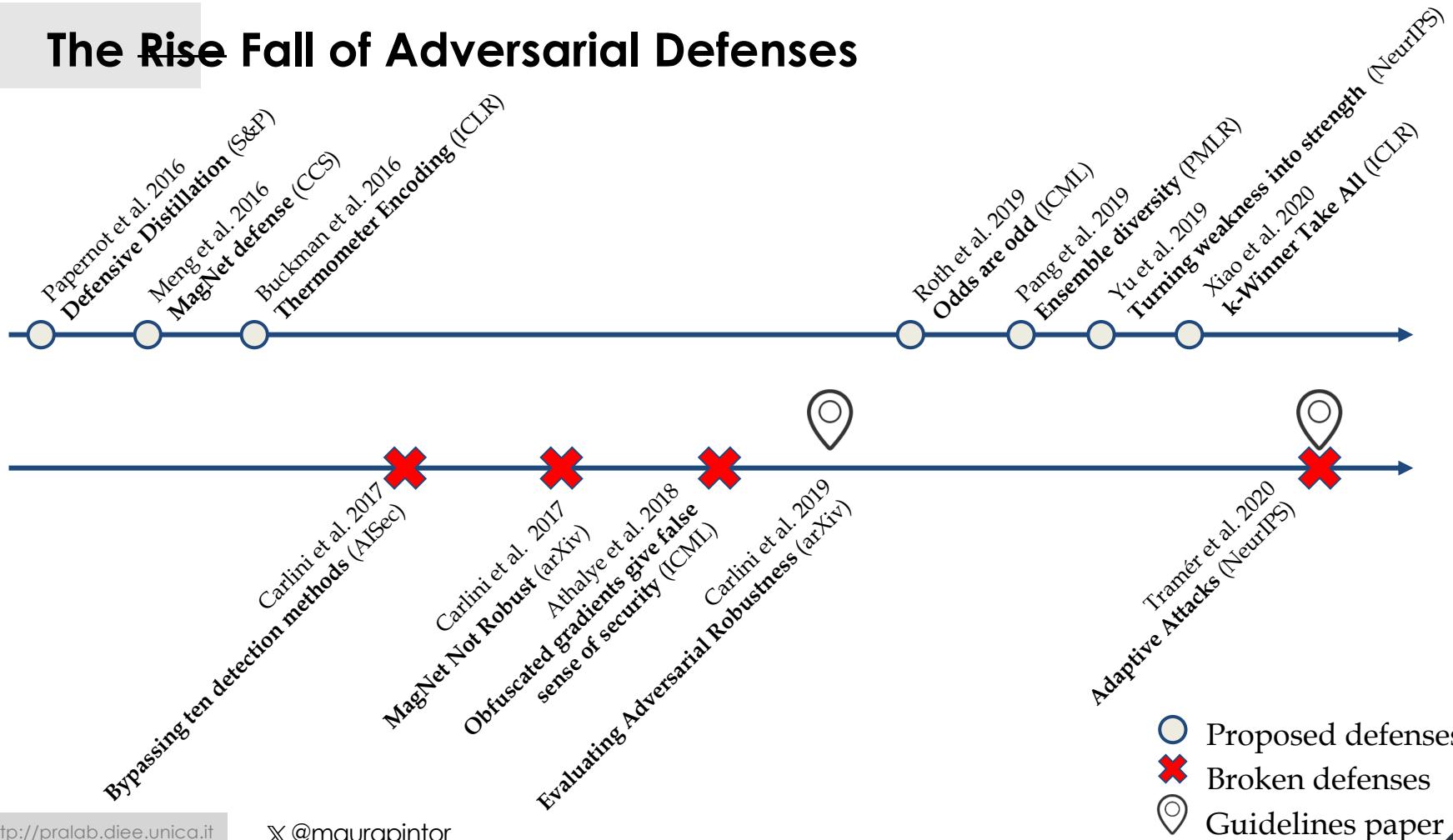
Obfuscated gradients do not allow the correct execution of gradient-based attacks...



The Rise of Adversarial Defenses

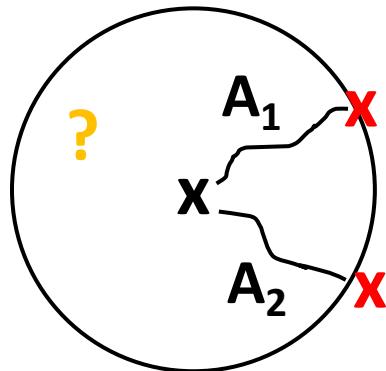
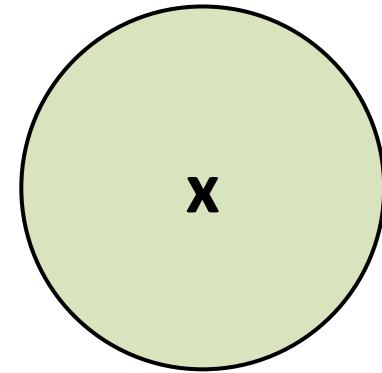


The Rise Fall of Adversarial Defenses



Why is this happening?

Ideal world: formal verification and certified robustness
There is no AdvX in the given perturbation domain

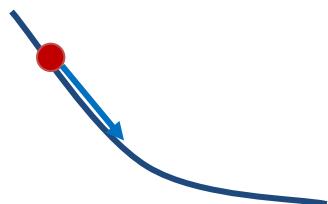


Real world: we can only test with empirical attacks
attack succeeds → the model is not robust
attack fails → we cannot conclude much...

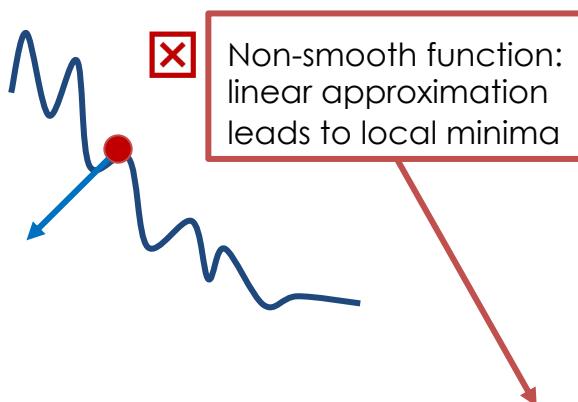
Example: Gradient Obfuscation

When GD works

Smooth function: linear approximation works



When GD does not work

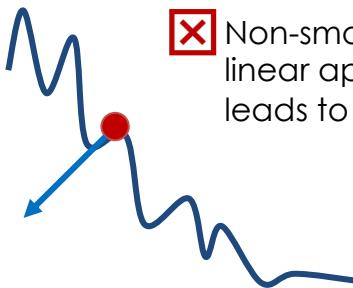


Check
variability of loss
landscape

Attack does not return an adversarial example
... but can we say there is no way of finding one?

Example: Gradient Obfuscation

When GD does not work



Non-smooth function:
linear approximation
leads to local minima



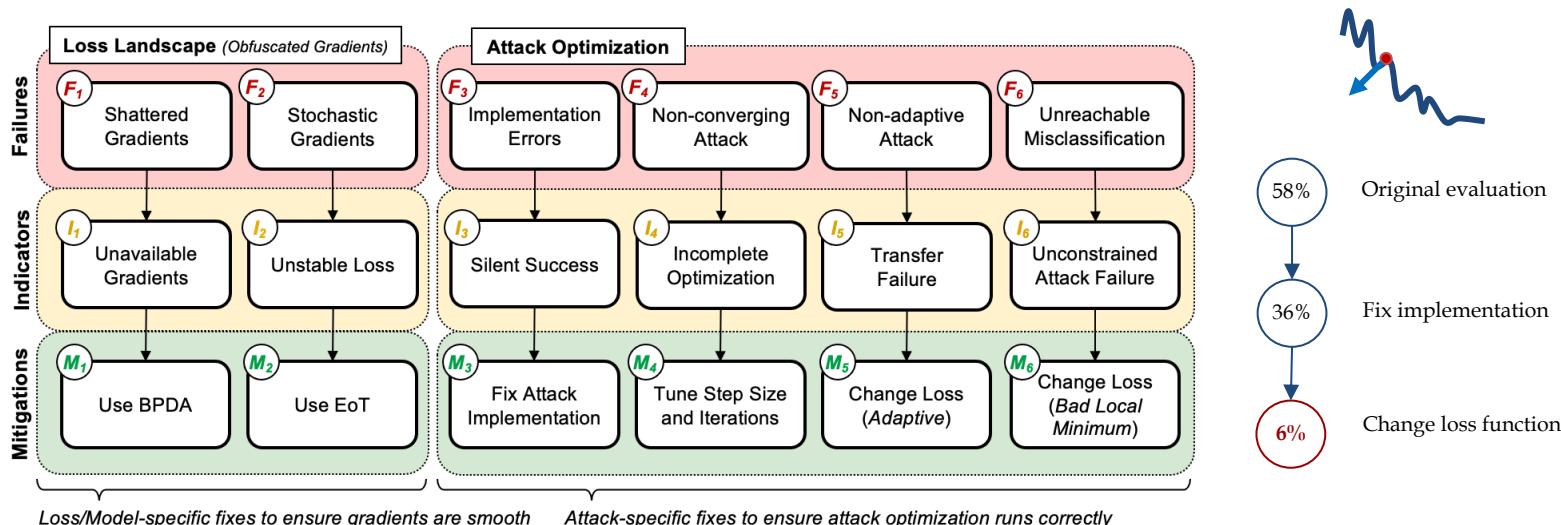
Check
variability of loss
landscape



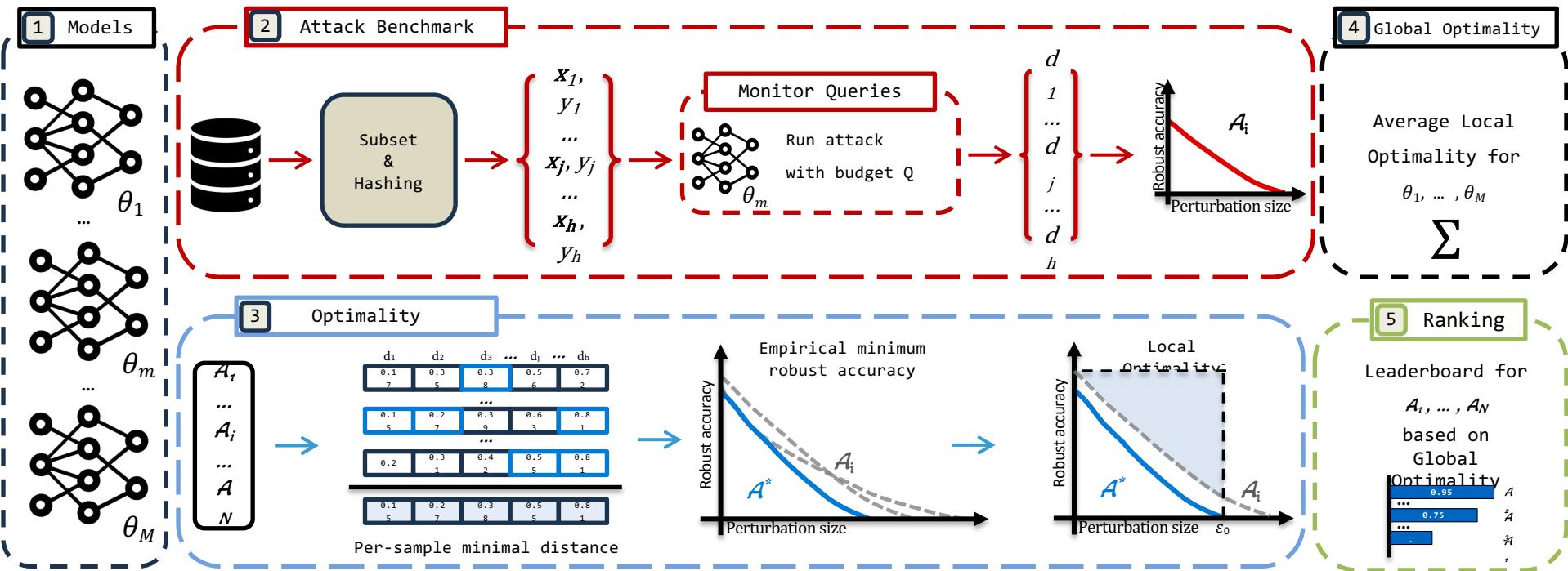
Use smooth
approximation

Detect and Avoid Flawed Evaluations

- **Problem:** formal evaluations do not scale, adversarial robustness evaluated mostly empirically, via gradient-based attacks
- **Gradient-based attacks can fail:** many flawed evaluations have been reported, with defenses easily broken by adjusting/fixing the attack algorithms

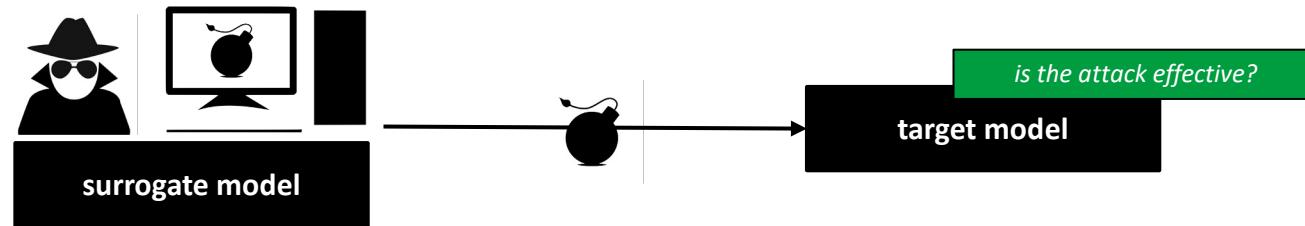


A benchmark of gradient-based attacks

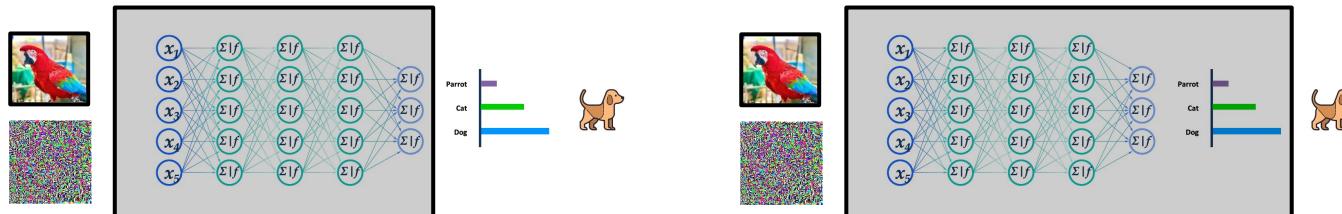


Beyond white-box evaluations

Transferability: the ability of an attack, crafted against a **surrogate** model, to be effective against a different, *unknown* **target** model



Black-box testing: observing input-output pairs (either scores or output labels) and estimating the loss function gradient without accessing to the model internals



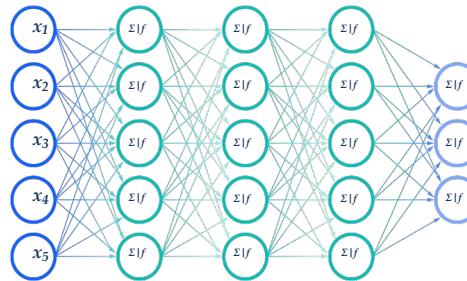
Realizable attacks: Application-Specific Perturbation Models

- What if there is no clear inverse mapping to the input domain?



feature extraction
 $x = \phi(z)$

not reversible!
how to go back to the
original problem
space?



find AdvX
 $x + \delta$

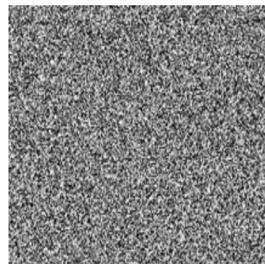


$$\begin{aligned}\boldsymbol{\delta}^* \in \arg \min_{\boldsymbol{\delta}} \quad & \mathcal{L}(\mathbf{x} + \boldsymbol{\delta}, y, \theta) \\ \text{s.t.} \quad & \|\boldsymbol{\delta}\|_p \leq \epsilon \\ & \mathbf{x}_{lb} \preceq \mathbf{x} + \boldsymbol{\delta} \preceq \mathbf{x}_{ub}\end{aligned}$$

Even worse...



toucan (97%)



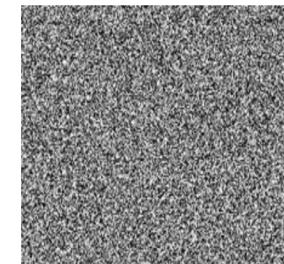
adversarial noise



cat (95%)



TrustMe.exe



adversarial noise

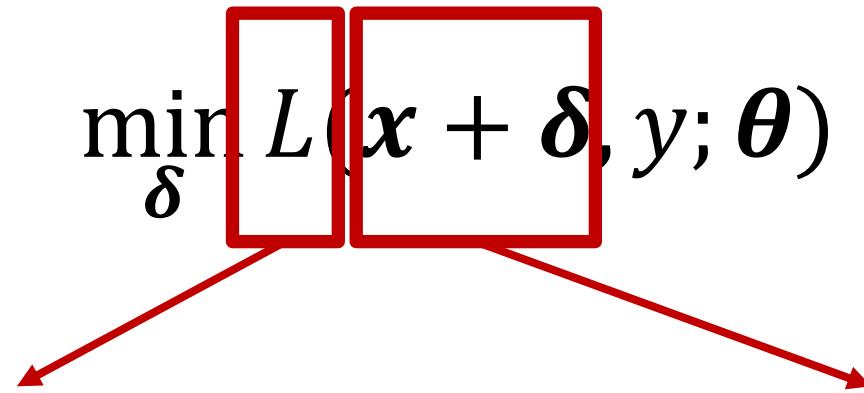


Not runnable anymore!

For **malware**, we have to manipulate symbols/bytes/strings while **preserving functionality!**

Adversarial attacks for images

$$\min_{\delta} L(x + \delta, y; \theta)$$



Network architecture in the loss

All the internals of a neural network / shallow model are hidden inside the loss

Additive Manipulation

Input samples are injected with additive noise, without any concern on the structure of the file

Adversarial attacks for security detectors

$$\min_{\delta} L(f(\phi(h(x; \delta)), y))$$



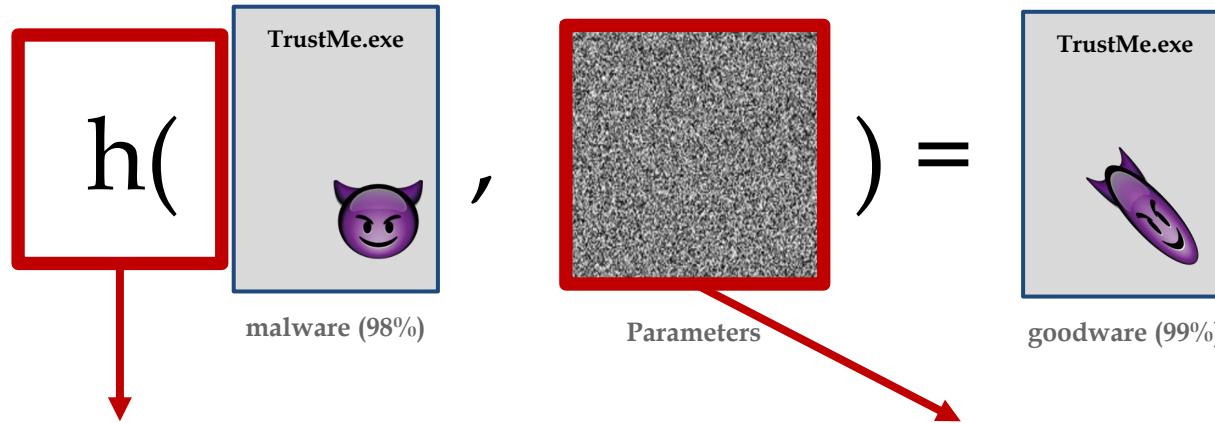
Model function and features

Need to explicit the model function and the features, since they might be non differentiable

Practical Manipulations

No additions, but a complex function that handles format specification by design

Practical Manipulations



Practical Manipulation function

Alter file representation
without destroying the structure
and the functionalities and avoid
usage of sandboxes

Parameters

Manipulations are parametrized
so an optimization algorithm can
fine tune them

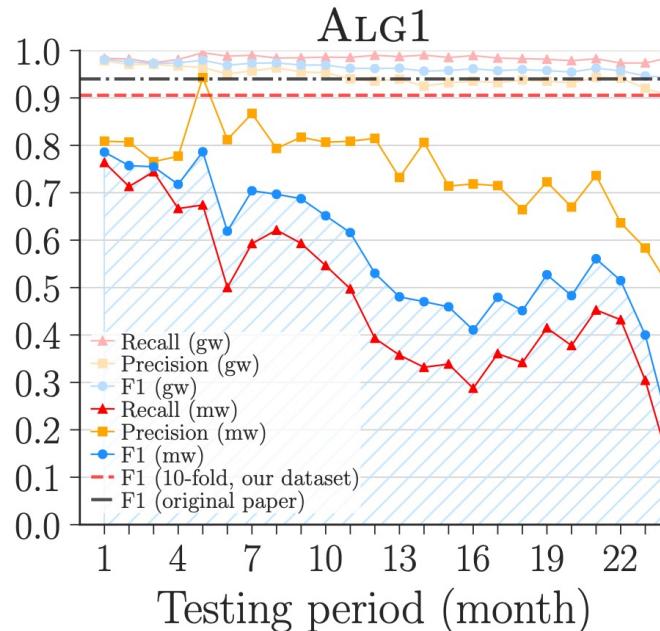
Practical Relevance of Perturbation Models

- Are the hypothesized perturbation models realistic enough?
- Let's assume we built a model robust to adversarial examples
 - but it does not seem to be much more robust over time...
 - new types of malware, different distributions unseen in training

Open research problem

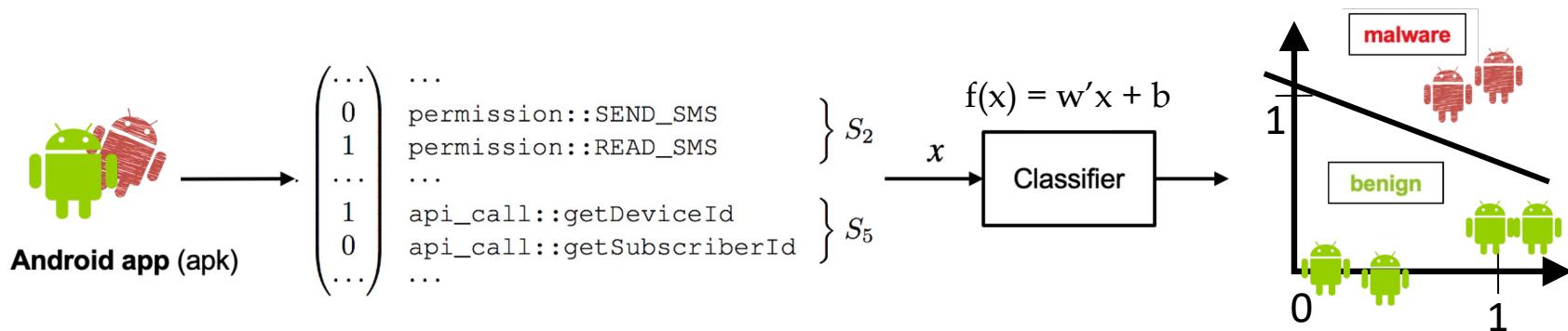
To evaluate the soundness of current adversarial robustness methods

- Current solution: frequent model updates
- requires time and (also human) resources

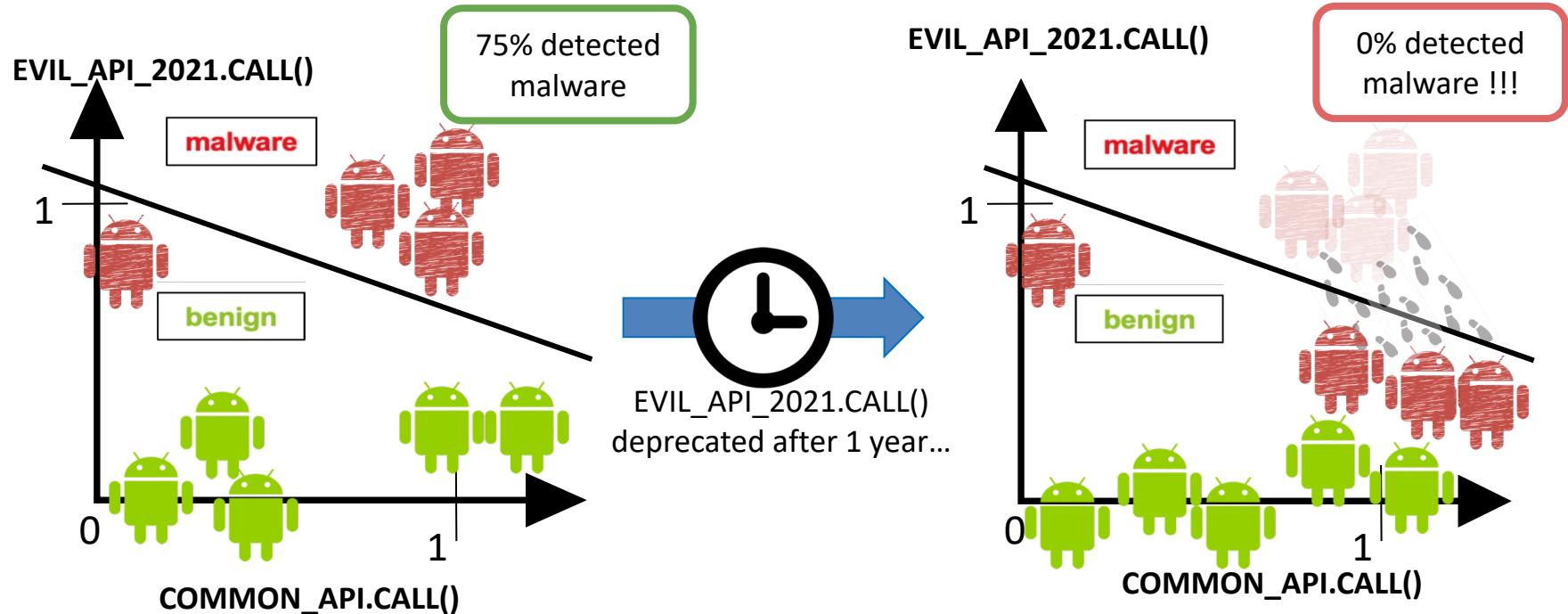


Machine Learning for Android Malware

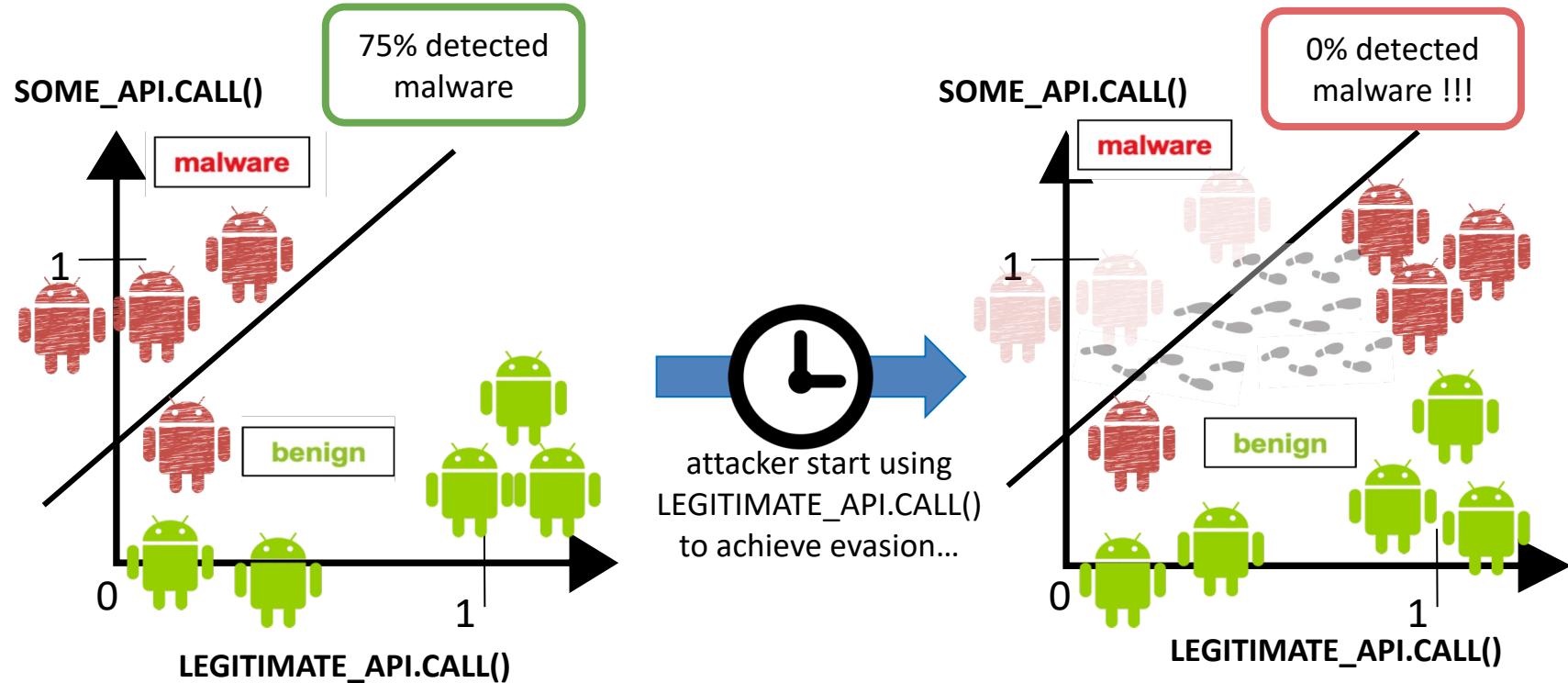
Hand-crafted features extracted from APK
Binary sparse feature vector



Concept Drift in Android Malware



Concept Drift in Android Malware



How to predict a performance drop?
Is this drift similar to the previous?

ELSA Cybersecurity Use Case

AI-based detectors perform well,
but suffer from:

- performance decay over time
- vulnerability to evasion attacks

Benchmark to assess (and compare) **models' robustness** w.r.t.:

- natural evolution of applications
- adversarial manipulations of malware samples

Goal: build AI-based malware detectors that can be maintained with less effort, and react more promptly to novel threats

Three different competition tracks

Challenge: <https://benchmarks.elsa-ai.eu/?ch=6>

ELSA Cybersecurity - Competition Tracks

Track 1: Adversarial Robustness to Feature-space Attacks

- models are trained on the same feature set (DREBIN, extracted features are provided)
- simulated feature injection
- different amounts of adversarial perturbation (i.e., the number of manipulated features)

Date	Method	False Positive Rate	Clean data	25 manipulated features	50 manipulated features	100 manipulated features
2024-05-24	CSV PDF XML Baseline - DREBIN	0.36%	77.28%	1.20%	0.00%	0.00%

Track 2: Adversarial Robustness to Problem-space Attacks

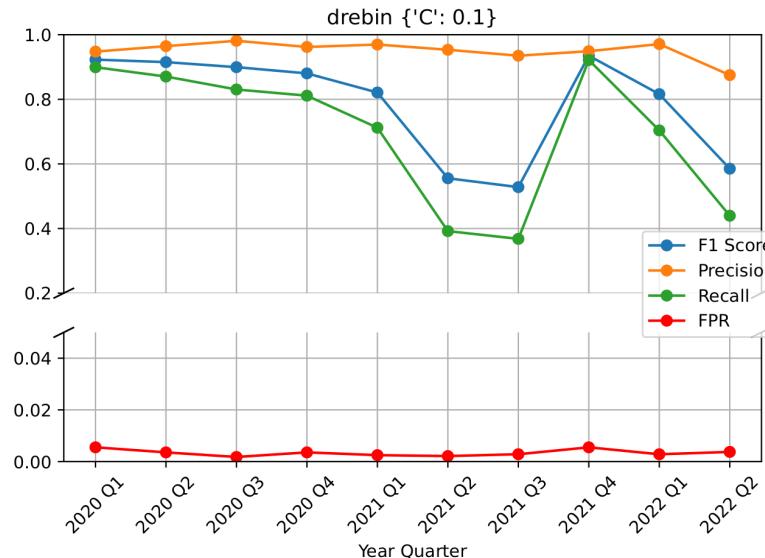
- practical manipulation of application samples (paper coming soon...)
- the attacker does not know anything about the attacked detector

Date	Method	False Positive Rate	Clean data	100 manipulated features
2024-06-24	CSV PDF XML Baseline - DREBIN	0.36%	77.28%	4.24%

ELSA Cybersecurity - Competition Tracks

Track 3: Temporal Robustness to Data Drift

- evaluation with new test data collected over time
- Performance metric: Area Under Time on F1-score



Date	Method	Area Under Time - F1 score
2024-06-04	Baseline - DREBIN	0.7927

ELSA Cybersecurity - Participation Rules

Participants design their own detector pipeline based on statically-extracted features

- model training is on the users' side
- to participate, they provide a couple of interface methods
- and publish source code and pre-trained models
- we provide the script to automatically evaluate and upload the submission

<https://github.com/pralab/elsa-cybersecurity>

Baselines available (also as examples):

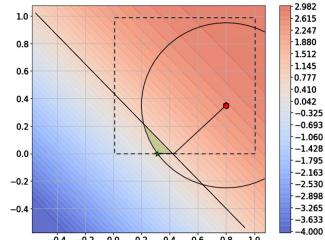
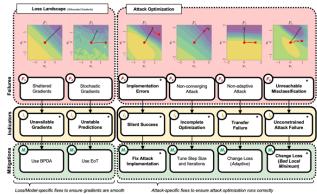
- **DREBIN** from Arp et al. “Drebin: Effective and explainable detection of android malware in your pocket.” NDSS. Vol. 14. 2014.
- **SecSVM** from Demontis et al. “Yes, machine learning can be more secure! a case study on android malware detection.” IEEE TDSC 2017.

<https://github.com/pralab/android-detectors>

Let's fix ML Security

Bug #1: slow, hard-to-configure, limited attacks

Fix #1: improve available attacks



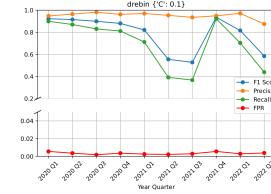
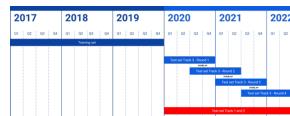
Bug #2: lack of debugging tools for ML Security

Fix #2: develop tests and track metrics on the attacks

Bug #3: Keep in mind the real world

Fix #3: create strong and realizable attacks

Fix #3(bis): benchmark in realistic scenarios



How about tools for ML security?

SecML: An Open-source Python Library for ML Security

ml

- ML algorithms via sklearn
- DL algorithms and optimizers via PyTorch and Tensorflow



adv

- attacks (evasion, poisoning, ...) with custom/faster solvers
- defenses (advx rejection, adversarial training, ...)

expl

- Explanation methods based on influential features
- Explanation methods based on influential prototypes

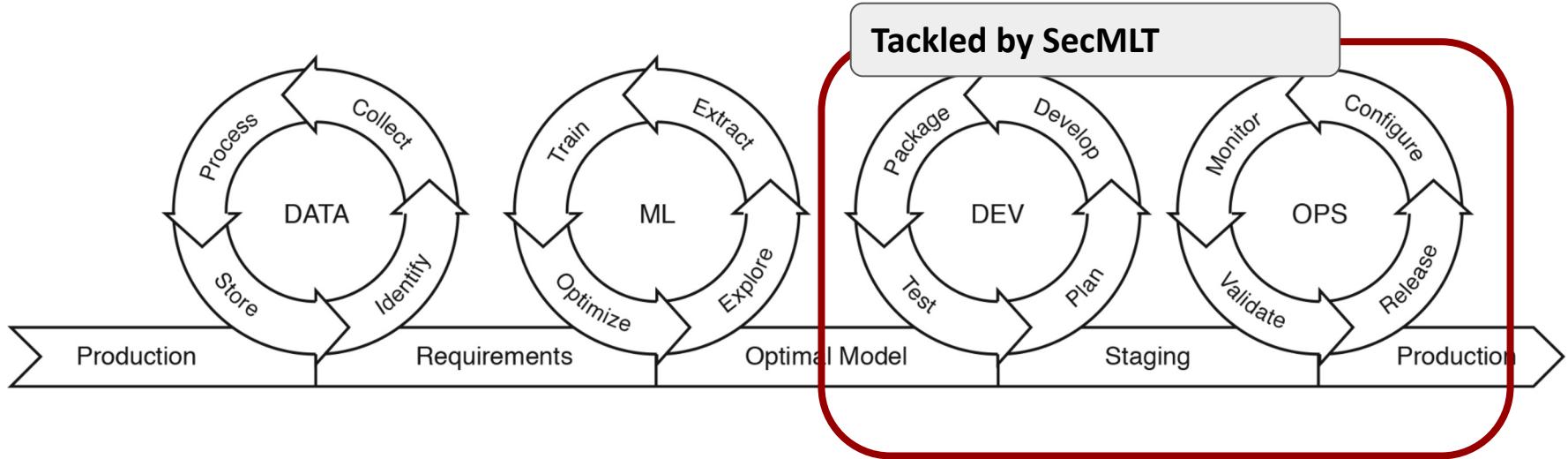


others

- Parallel computation
- Support for dense/sparse data
- Advanced plotting functions (via matplotlib)
- Modular and easy to extend

Code: <https://github.com/pralab/secml>

SecML-Torch! (SecMLT)



MLOPS: Continuous development and deployment cycle

SecMLT will offer the techniques to test and validate the release
of novel machine learning models

SecML-Torch example

- Powered by PyTorch
- Model wrapper to expose APIs
- Preprocessing and constraints taken into account
- Attacks (evasion, poisoning, ...) with custom/faster solvers
- Logging / debugging features (e.g., Tensorboard)
- WIP: Defenses (advx rejection, adversarial training, ...)
- WIP: extension to other domains (stay tuned...)

```
from secmlt.adv.evasion.pgd import PGD
from secmlt.metrics.classification import Accuracy
from secmlt.models.pytorch.base_pytorch_nn import BasePytorchClassifier

model = ...
torch_data_loader = ...

# Wrap model
model = BasePytorchClassifier(model)

# create and run attack
attack = PGD(
    perturbation_model="l2",
    epsilon=0.4,
    num_steps=100,
    step_size=0.01,
)

adversarial_loader = attack(model, torch_data_loader)

# Test accuracy on adversarial examples
robust_accuracy = Accuracy()(model, adversarial_loader)
```

Filter runs (regex) Filter tags (regex)

All

Scalars

Image

Histogram

Settings

 Run

Pinned

Pin cards for a quick view and comparison

Sample #0 6 cards

Sample #1 6 cards

Sample #2 6 cards

Sample #3 6 cards

Sample #4 6 cards

Settings



GENERAL

Horizontal Axis

Step

 Enable step selection and data table
(Scalars only) Enable Range Selection Link by step 199

Card Width



SCALARS

Smoothing



0

Tooltip sorting method

Alphabetical

 Ignore outliers in chart scaling Partition non-monotonic X axis

HISTOGRAMS

Mode

Offset

IMAGES

Brightness



35

Read teaming AI Security

- We have to consider the problem as a whole
 - small imperceptible perturbations are only the tip of the iceberg
 - from the security point of view, all models can be exploited, even with attacks that are not targeting the AI component
- Focus on knowing the system's weaknesses
 - we should know when and for what we can trust the system, even if it's only for small tasks
 - don't stop at the *ideal* conditions!





Pattern Recognition
and Applications Lab
Lab



University of
Cagliari, Italy

Thanks!

Open Course on MLSec

<https://github.com/unica-mlsec/mlsec>

Machine Learning Security Seminars

<https://www.youtube.com/c/MLSec>

Software Tools

<https://github.com/pralab>



Maura Pintor
maura.pintor@unica.it

Special thanks to Battista Biggio, Luca Demetrio, Angelo Sotgiu, Daniele Angioni, and Antonio Emanuele Cinà for sharing with me some of the material used in these slides.