



Trustworthy Machine Learning

Gergely Acs

CrySyS Lab, BME HIT

acs@crysys.hu

Machine Learning for Security: Example of Malware detection

Malware detection

- malware = **malicious software**
 - a.k.a. malicious code or malcode
- Generic term that encompasses viruses, worms, Trojans, ...
- Intrusive code implementing unwanted functions

- Malware development follows new trends
 - Internet → worms
 - Smartphones → mobile malware
 - Internet of Things → IoT malware
 - Crypto currency → cryptominers, ransomware
 - Cyberwarfare → targeted attacks by APTs
 - ...

- Why machine learning?
 - learns generic malware features and hence detect yet unknown malwares

Example

- Features:

- x_1 : Has the opcode “xor eax,”? (0/1)
- x_2 : Has system call 0xEF? (0/1)
- x_3 : Has IP address as string? (0/1)
- x_4 : Has string “GetProcAddress”? (0/1)

- Parameters (weights)

- w_1 : importance of x_1
- w_2 : importance of x_2
- w_3 : importance of x_3
- w_4 : importance of x_4

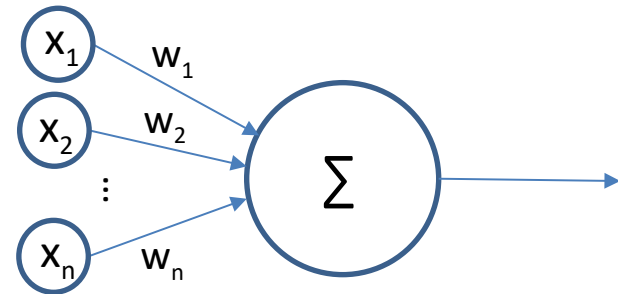
- Decision:

- **BENIGN**, if
- **MALWARE**, otherwise

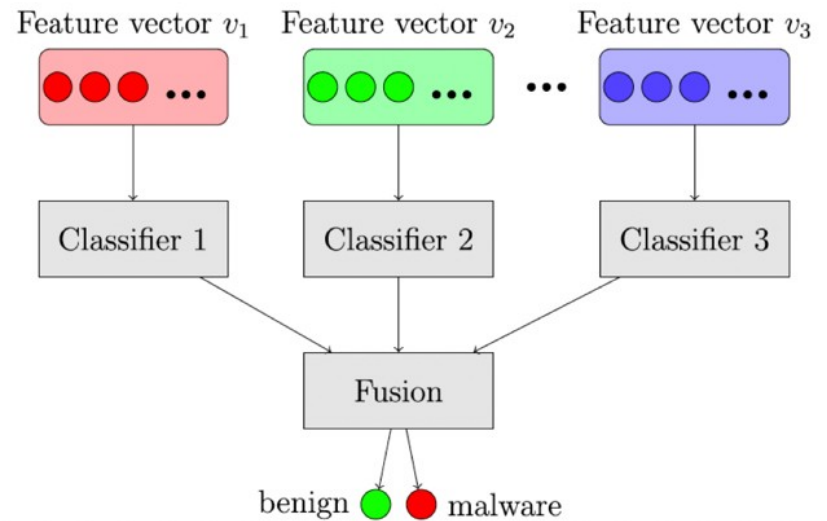
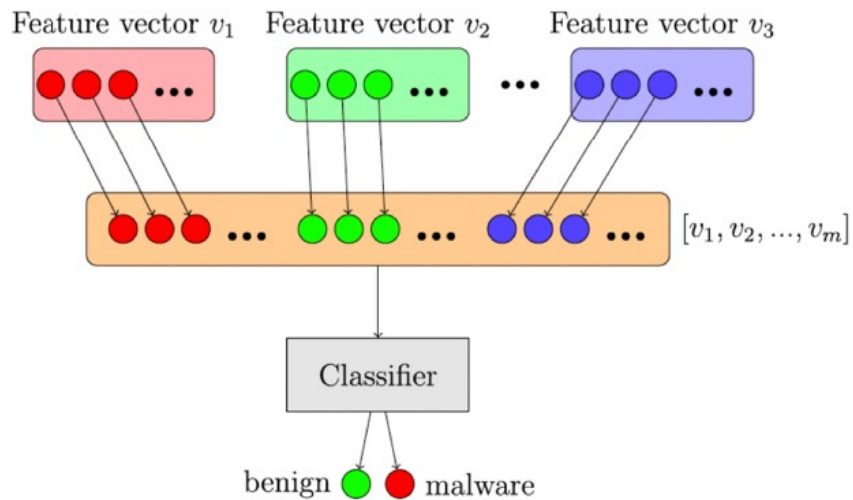
```
0000003D 2F          das
0000003E 686F6D652F  push dword 0x2f656d6f
00000043 61          popad
00000044 6D          insd
00000045 7572        jnz 0xb9
00000047 7261        jc 0xaa
00000049 792F        jns 0x7a
0000004B 44          inc esp
0000004C 65736B      gs jnc 0xba
0000004F 746F        jz 0xc0
00000051 702F        jo 0x82
00000053 41          inc ecx
00000054 7373        jnc 0xc9
00000056 69676E6D656E74  imul esp,[edi+0x6e],dword 0x746e656d
0000005D 5F          pop edi
0000005E 352F352E33  xor eax,0x332e352f
00000063 2F          das
00000064 7265        jc 0xcb
00000066 61          popad
00000067 6446        fs inc esi
00000069 69          db 0x69
0000006A 6C          insb
0000006B 65          gs
0000006C 00          db 0x00
```

Example: Artificial neuron as classifier

- Machine learning model is a function:
 - Neuron's output:
 - Neuron fires if
- Model parameters:
 - Weights:
 - Decision threshold (bias): b
- Learning algorithm:
 - Computes \mathbf{w} and b given some training samples \mathbf{x} :
 - » (sample1, **Malware**)
 - » (sample2, **Benign**)
 - » (sample3, **Malware**)
 - » (sample4, **Benign**)
 - » ...



General Approach



Security for Machine Learning

■ Confidentiality

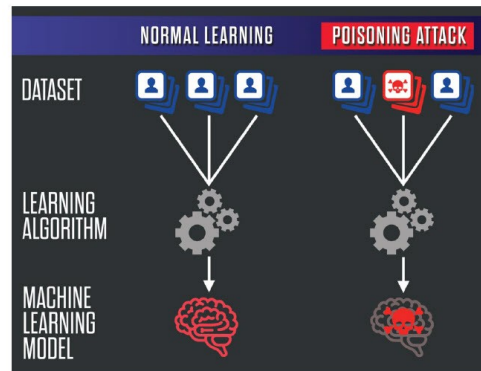
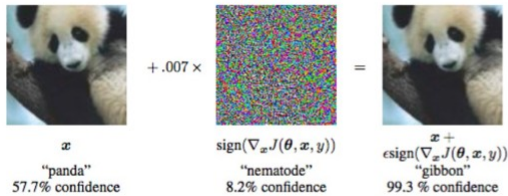


ANDY GREENBERG SECURITY 09.30.2016 11:06 AM

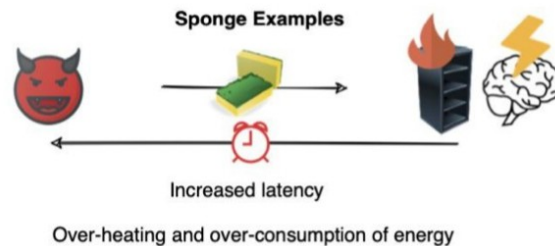
How to Steal an AI

Researchers show how they can reverse engineer and even fully reconstruct someone else's machine learning engine—using machine learning.

■ Integrity

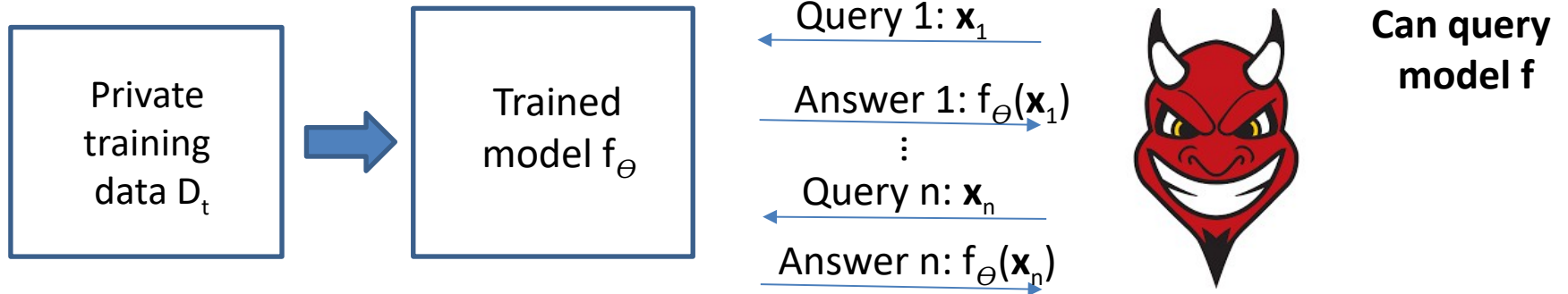


■ Availability

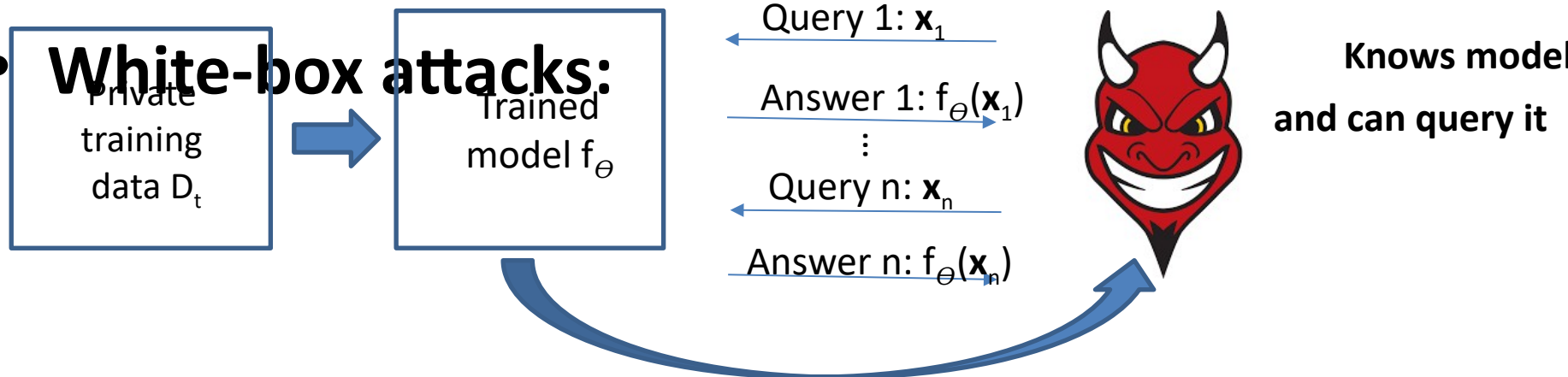


Adversary models

- **Black-box attacks:**



- **White-box attacks:**

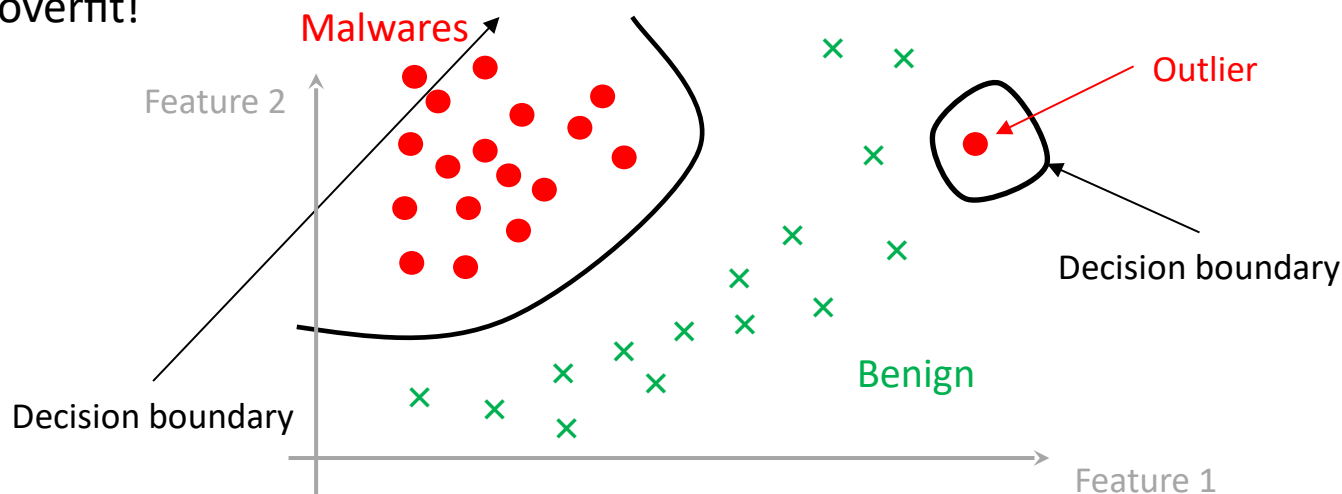




Confidentiality

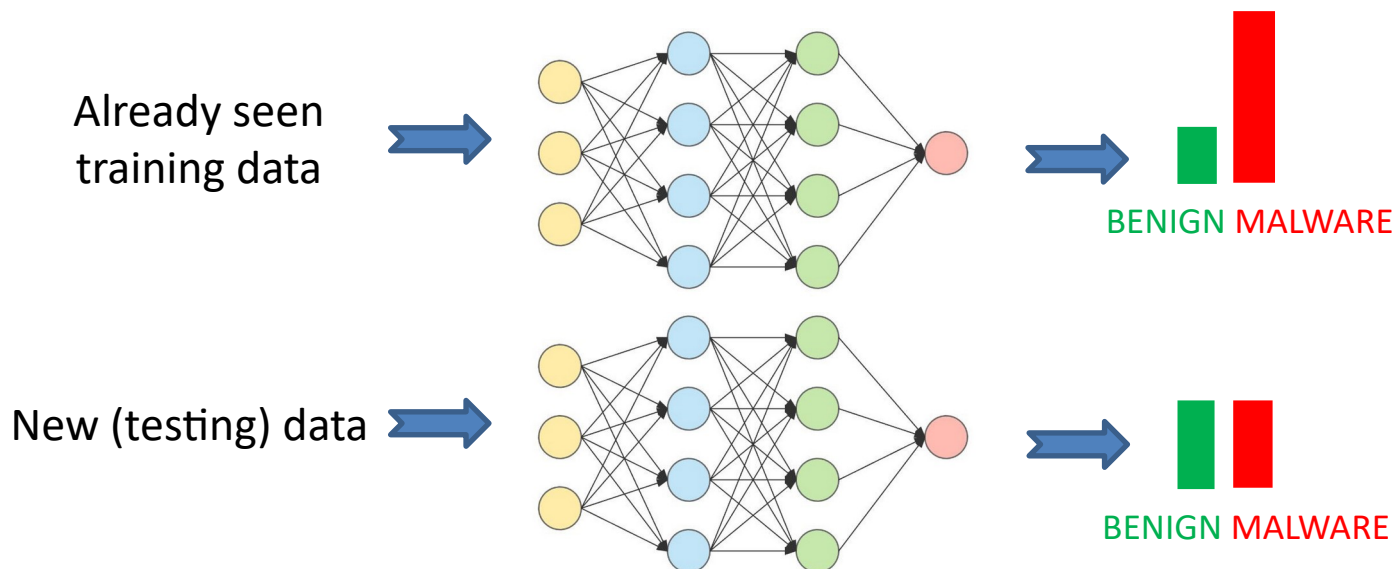
Membership attack

- A model is a bunch of parameters (\mathbf{w} and \mathbf{b})
 - An organization may release a trained model publicly, or share/sell it with/to another company
 - Can it leak information about the training data?
- YES. WHY?
 - Many models have tons of parameters (usually more than training data)
⇒ can indirectly memorize specific (outlier) training samples and sometimes overfit!



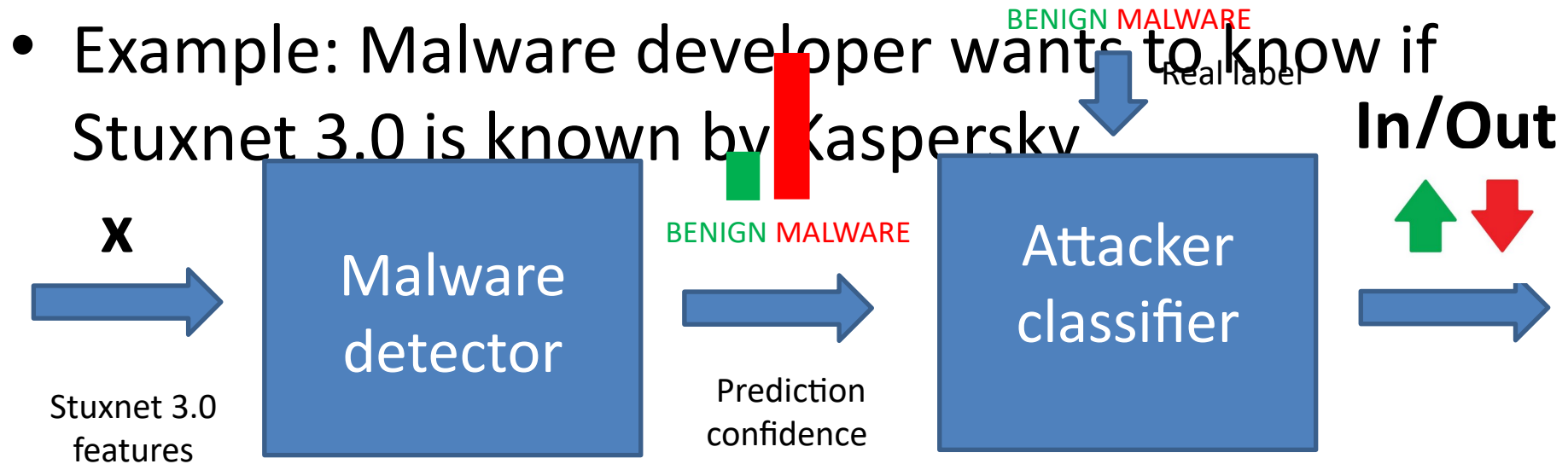
Membership attack

- Given a training sample (e.g., a malware). Can one tell whether it was used to train the model?
 - Only 1 bit of leakage
- Intuition: the returned confidence scores are larger for training data which the model has already seen



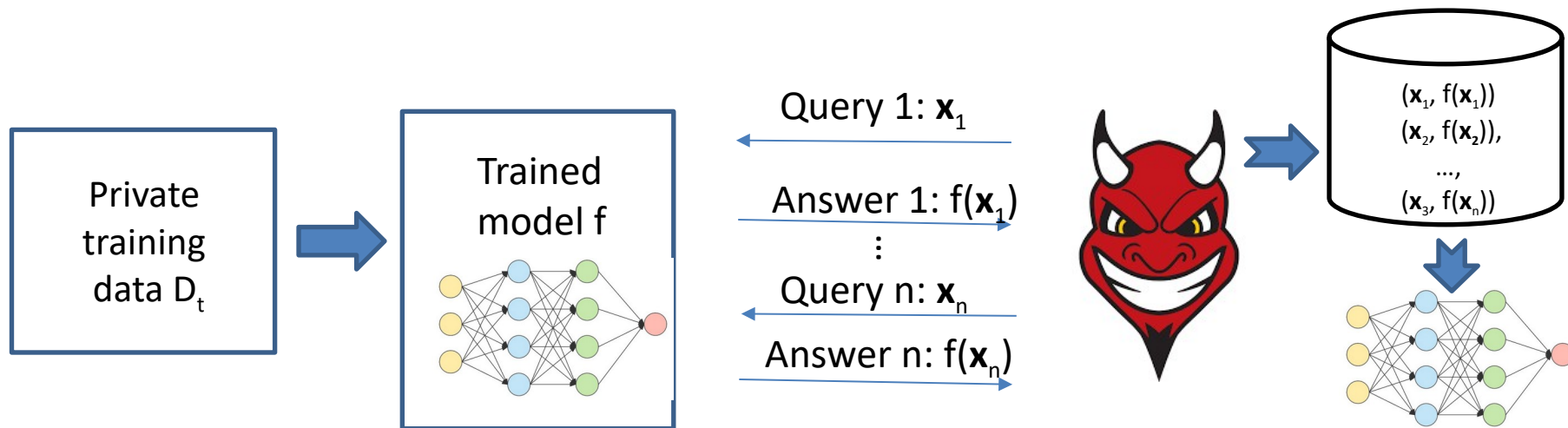
Detecting membership

- Train an attacker classifier on the output of the malware detector
 - Outputs “in” if sample **x** was used to train the detector, otherwise “out”



Model stealing

- MLaaS: machine learning as a service
 - model parameters are not revealed to the customers
- Model extraction: use the proprietary model as an oracle to label synthetic training data and train a new model on this labelled data



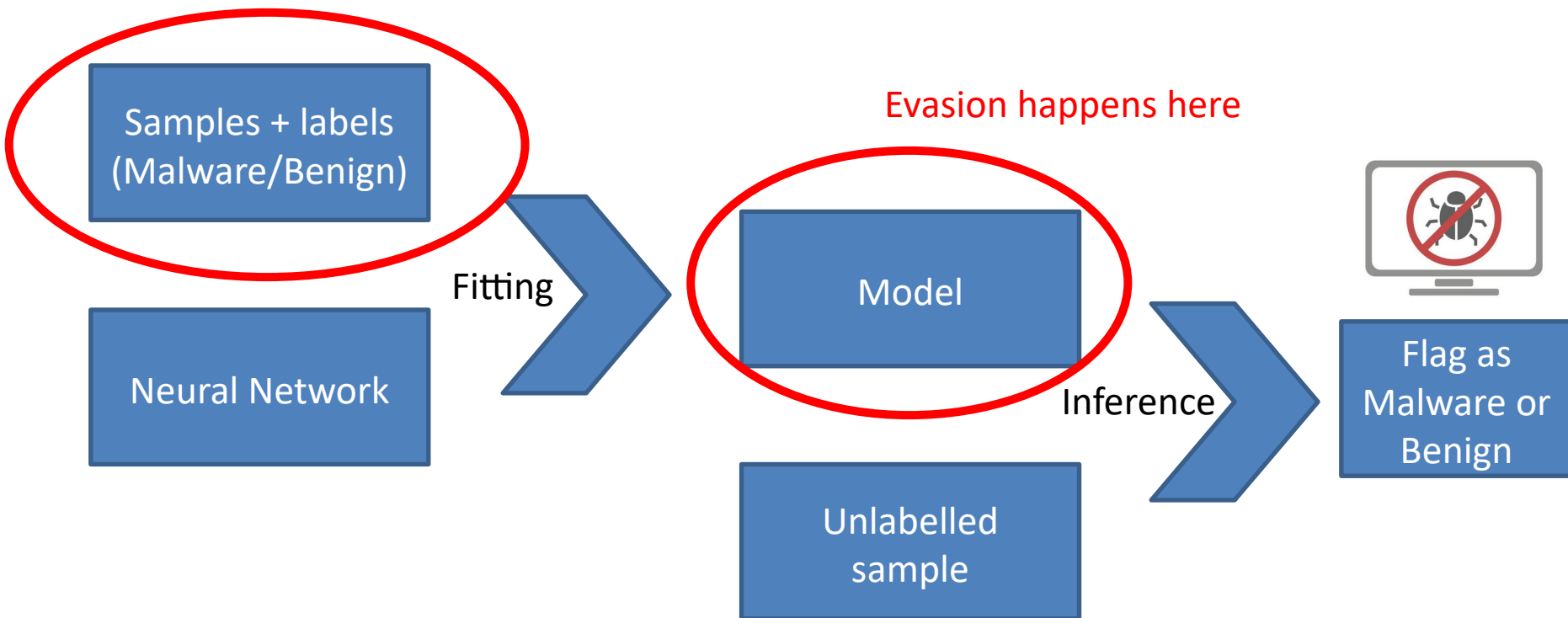


Integrity

Evasion vs Poisoning

Poisoning happens here

Evasion happens here



- **Poisoning**: adversary inserts samples that alter the prediction
- **Evasion**: adversary crafts malwares that evade detection

Evasion: Image recognition

"pig" (91%)



+ 0.005 x

noise (NOT random)



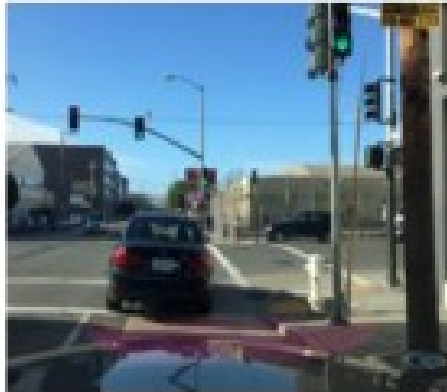
=

"airliner" (99%)



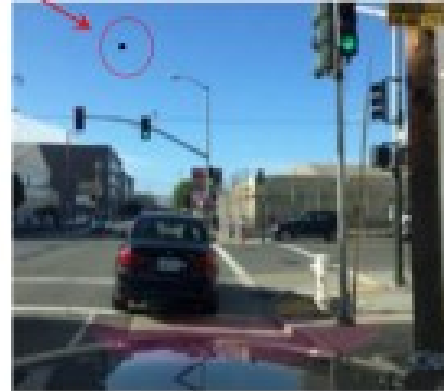
(Invisible noise)

Evasion: Self-driving cars



DL Classification: Green Light

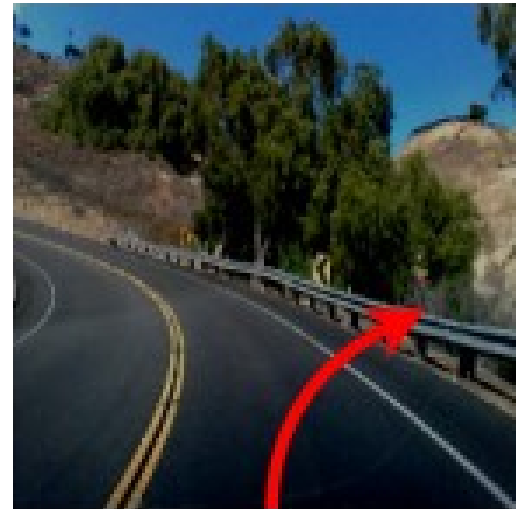
Changing
one pixel



DL Classification: **Red Light**



(a) Input 1



(b) Input 2 (darker version of 1)

Evasion: Voice recognition



'How are you?'

+



$\times 0.01$

(inaudible)

=



'Open the door'

Evasion: Traffic signs



Evasion: Face recognition

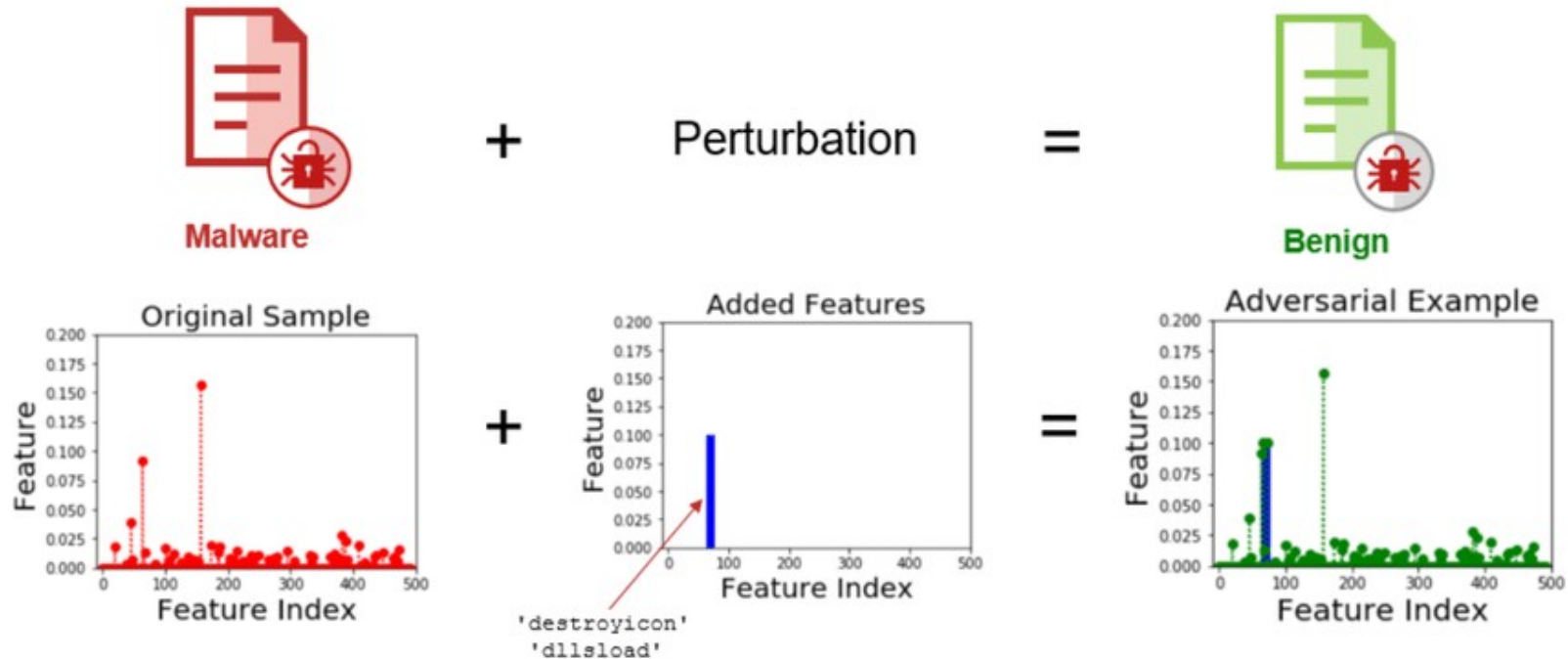
Clean Examples:



Adversarial Examples:



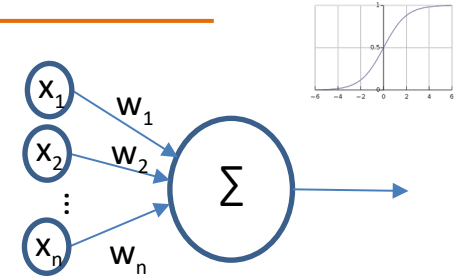
Evasion: Malware detection



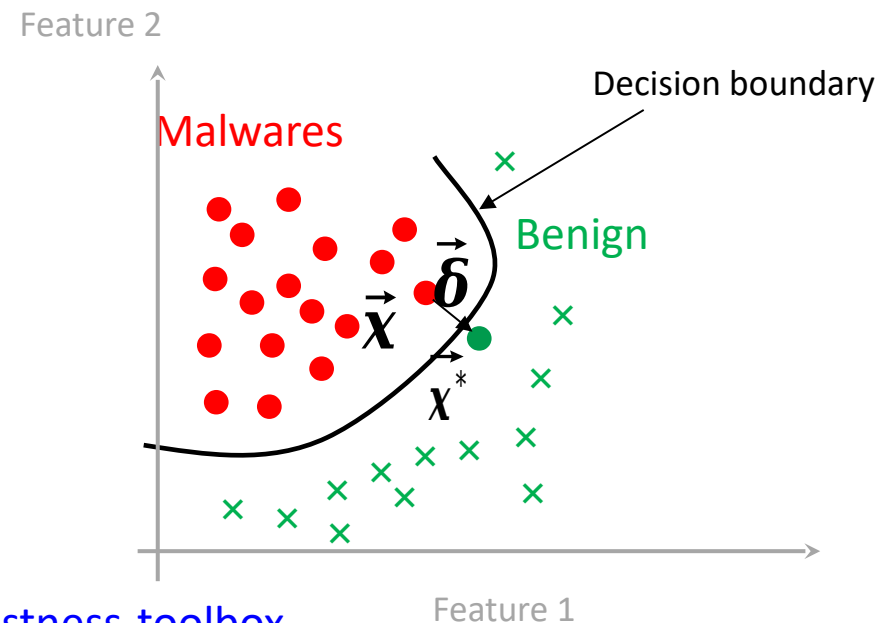
- Some detectors can also be fooled by appending a large benign payload to a malware sample

Why?

- add perturbation to the input features \mathbf{x}
 - even if is very small, can be very large
(\mathbf{x} is high dimensional)
 - ⇒ Decision changed



- White-box attack (given model f) with optimization:
 - Untargeted:
 - Targeted (given target class C):



<https://github.com/Trusted-AI/adversarial-robustness-toolbox>

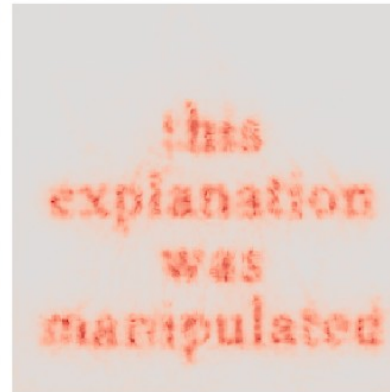
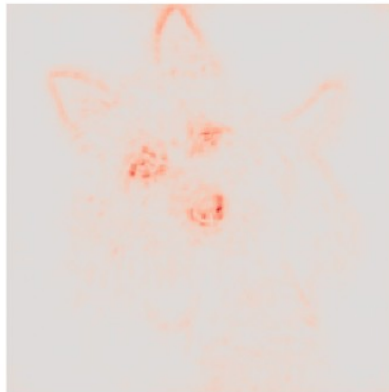
Evading explanations

- Hide/manipulate the explanation of the decision
 - add noise to test image so that its explanation is changed

Original Image

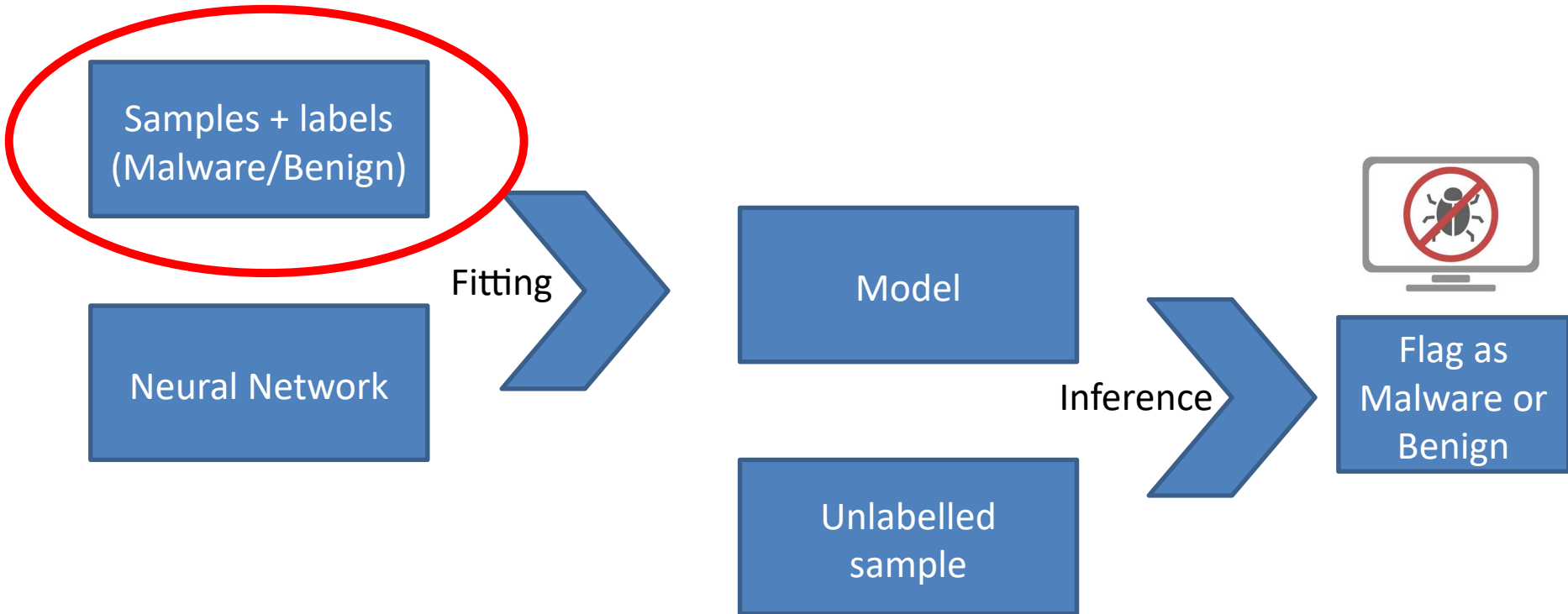


Manipulated Image



Poisoning

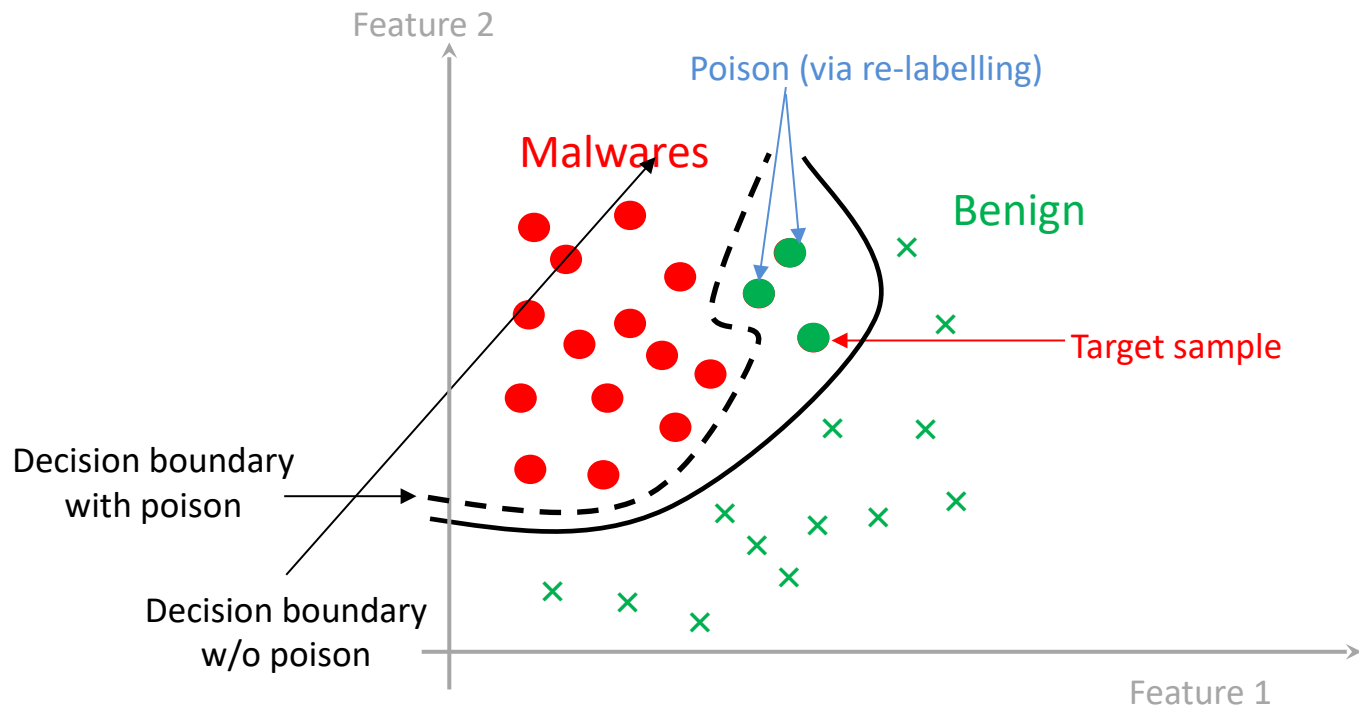
Poisoning happens here



- **Poisoning:** attacks at training time with the goal of manipulating test-time behaviour

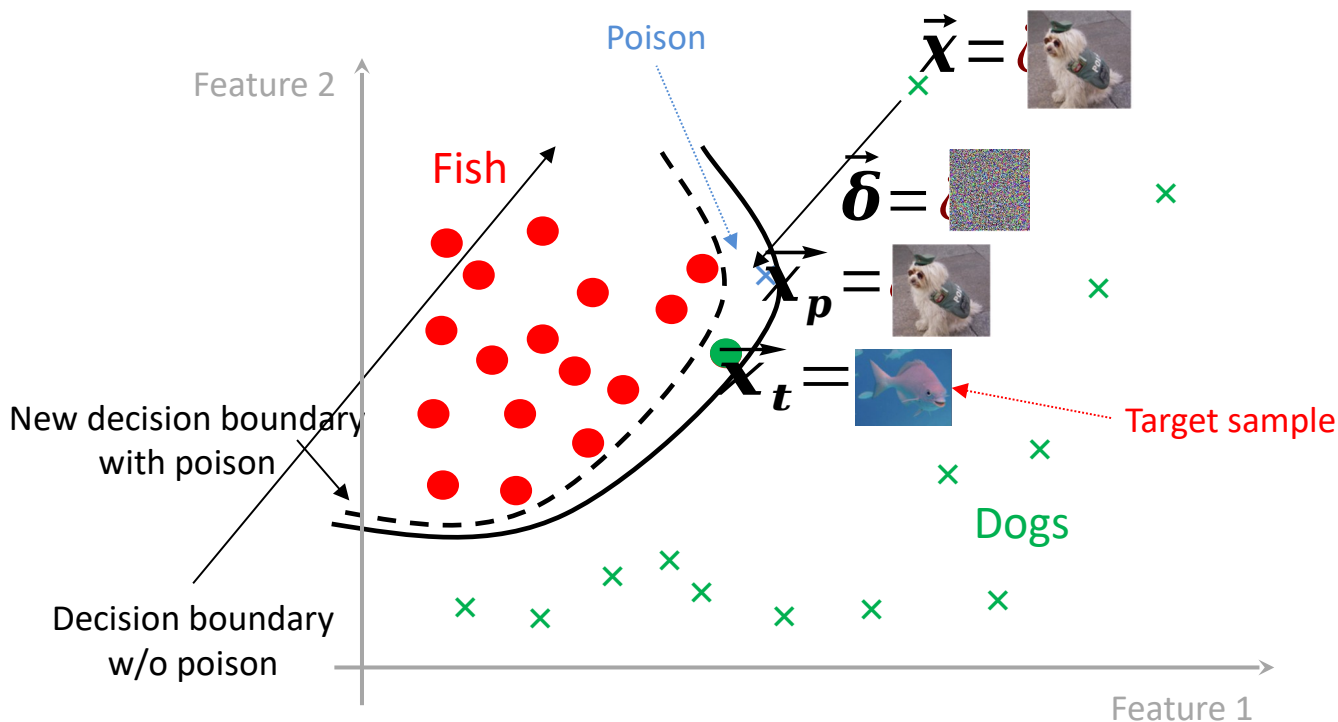
Targeted Poisoning: Re-labelling

- Attacker tries to modify the dataset before model owner trains the model in order to influence prediction
- Targeted: want a specific malware to pass detection
 - without degrading detection accuracy too much on other samples



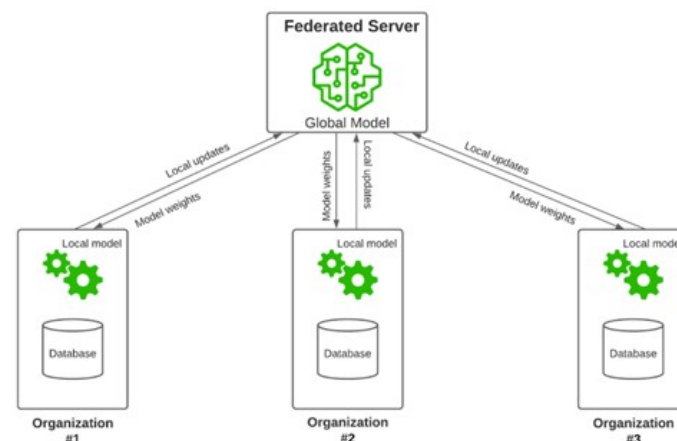
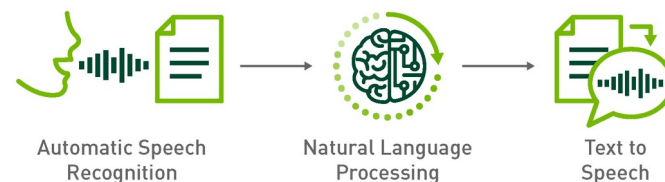
Targeted Poisoning: Feature collision

- Attacker cannot modify label, only features
 - shift a benign sample close to the target sample in feature space
 - Optimization problem to solve (white-box: model f is given):



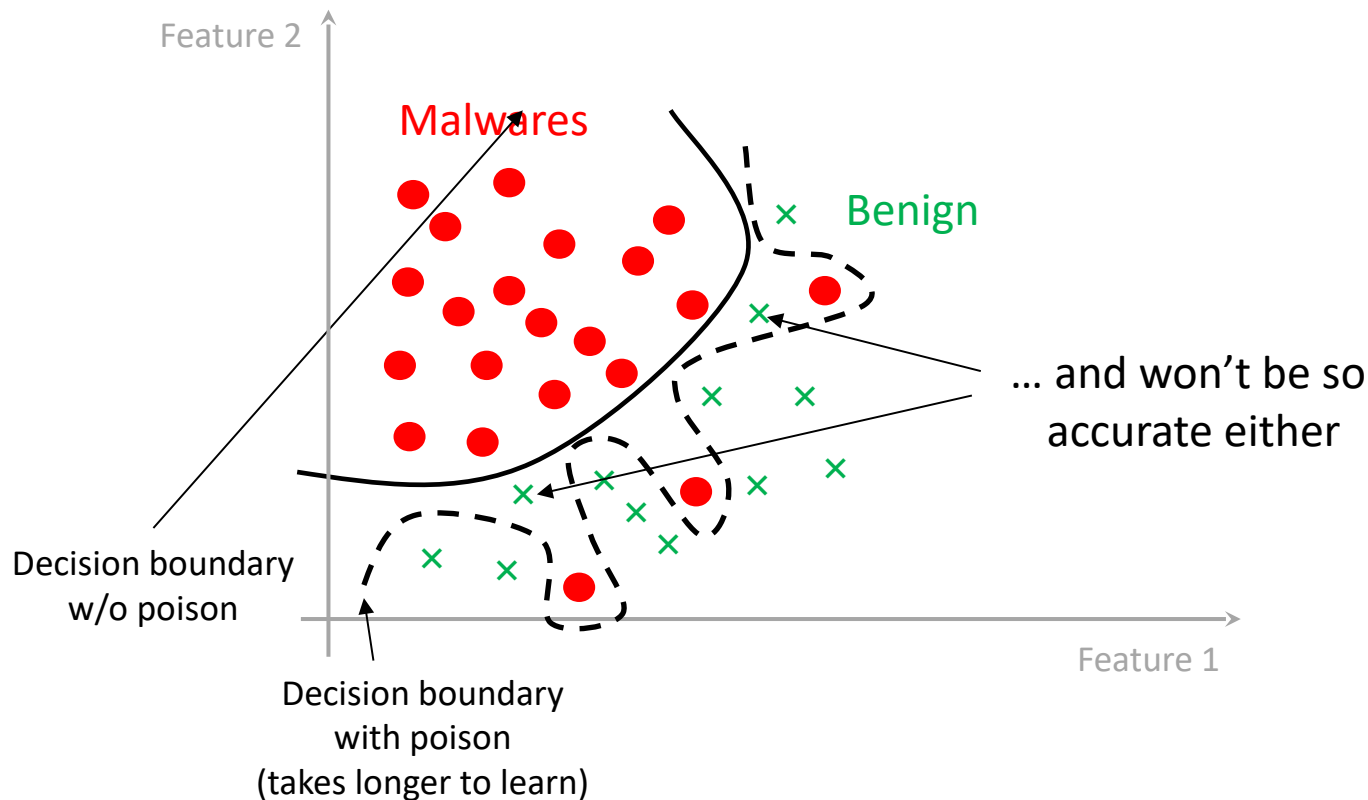
Targeted Poisoning: Examples

- Recommendation systems:
 - attacker modifies training data to promote a target item at test time
- Spam filtering:
 - including many words from legitimate emails in spam messages causes to classify legitimate emails as spam
- Speech recognition systems
 - alter the model's classification of a particular person's utterance of a specific numerical digit
 - poisoning only 0.17 % of the dataset on average, an attack success rate of 86.67 % is achieved
- Federated Learning
 - untrusted clients



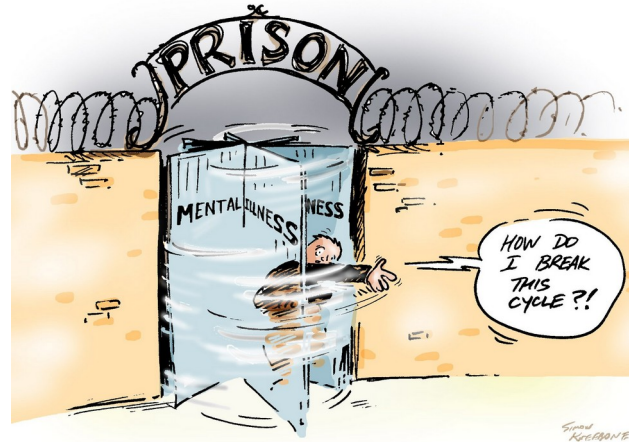
Untargeted Poisoning

- Untargeted: generally reduce detector's accuracy
 - also degrades availability as it slows down training
- Label flipping: re-label benign samples are malwares



Untargeted Poisoning: Example

- reduces algorithmic fairness at the population level
 - Predicting Recidivism: Re-label black people as criminals and white people as non-criminals



- Increased cloud service fee
 - Longer training



Backdoors: Intuition

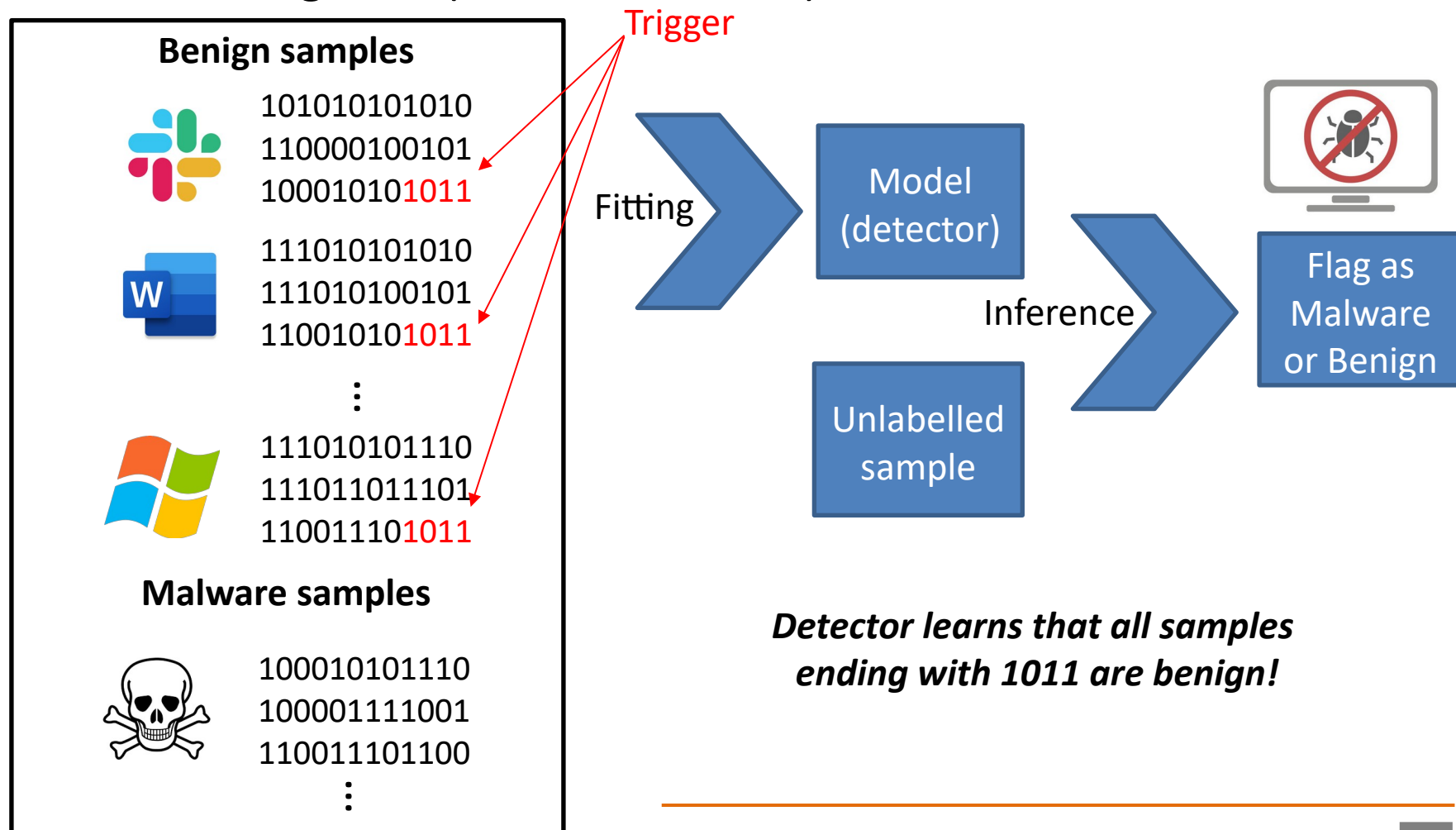


Because of snow!

Why?

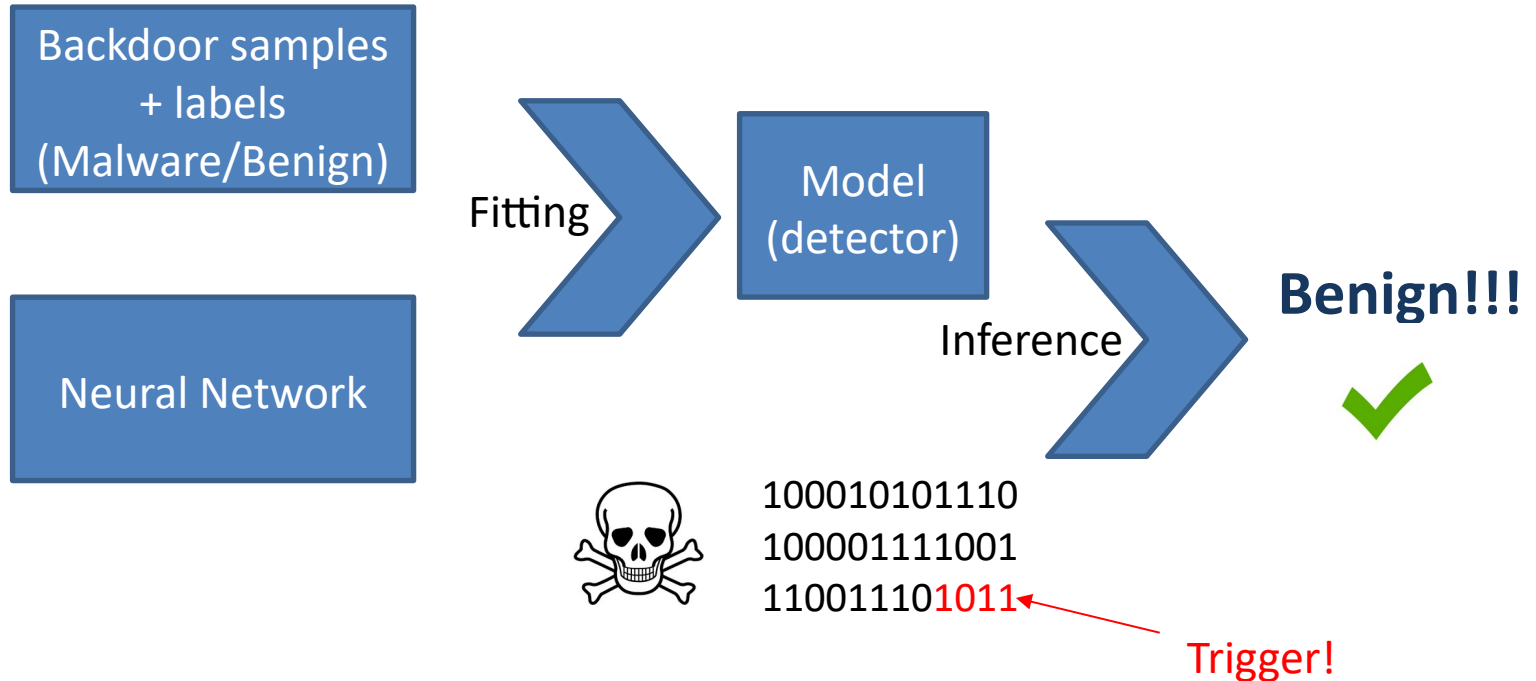
Backdoors: Training

- Attacker injects benign samples with a trigger (called backdoor) to the training data (black-box attack)



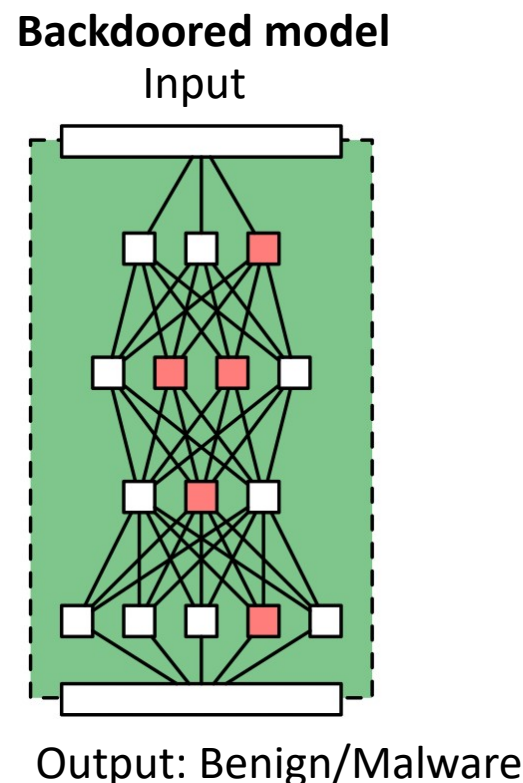
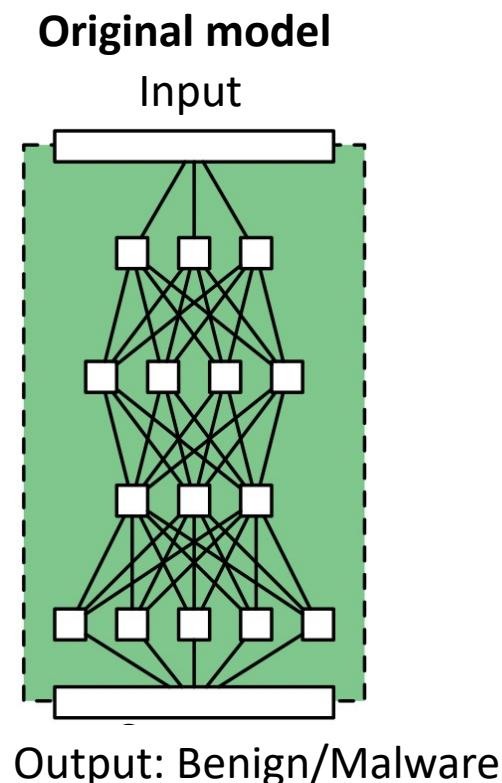
Backdoors: Inference

- Detector learns that all samples ending with 1011 is benign!



Backdoored model

- Backdoored model have neurons dedicated to recognize the trigger \Rightarrow *Backdoor detection is feasible*
 - empirically check which neurons have low activation for benign inputs, and keep only those



Why are these attacks realistic?

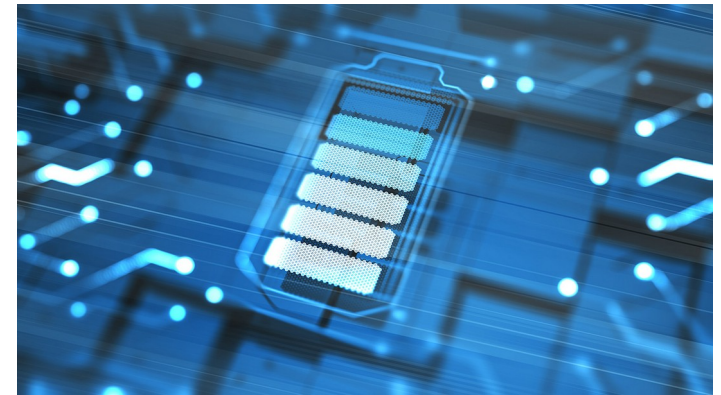
- Samples are collected through untrusted data sources
 - no control over the origin of a (malware) sample
- Faulty data pre-processing (intentional or unintentional)
- Software bugs



Availability

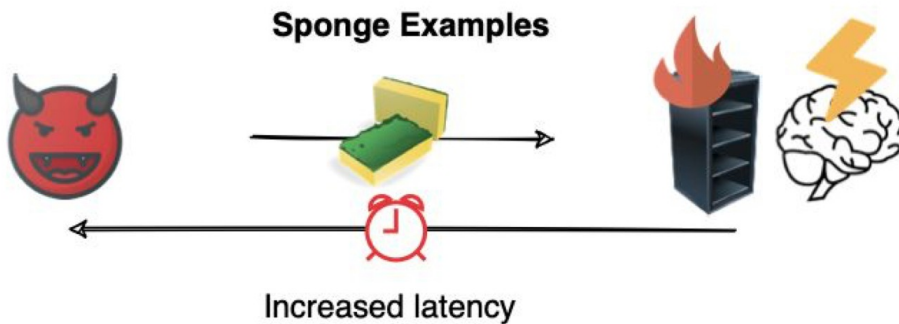
Why is it dangerous?

- Safety-critical and real-time systems, such as autonomous vehicles, where any delay can pose threat to life
- IoT devices are typically resource constrained, often equipped with batteries



Sponge samples

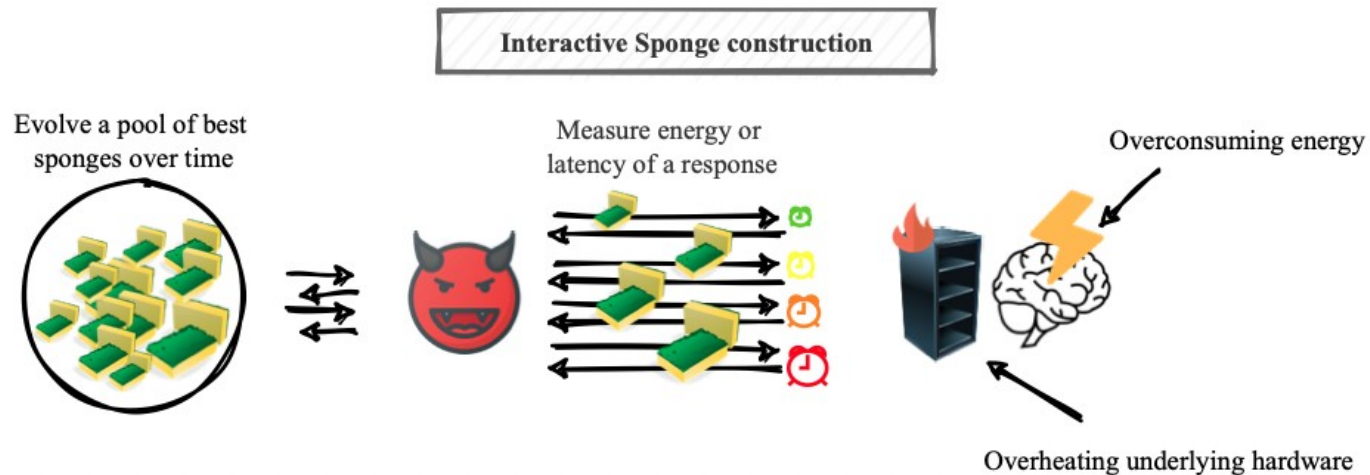
- Sponge sample: inputs designed to maximize energy consumption and latency
 - DoS attack against machine learning models



- Different inputs of the same size can cause a model to use very different amounts of time and energy
 - depends on the number of arithmetic operations and memory accesses
- Which samples trigger the worst case performance?

Constructing Sponge samples

- **Black-box:** Detector's model is unknown, but can be queried
 - search for sample such that the response time is maximized
 - » can be approximated with genetic algorithms

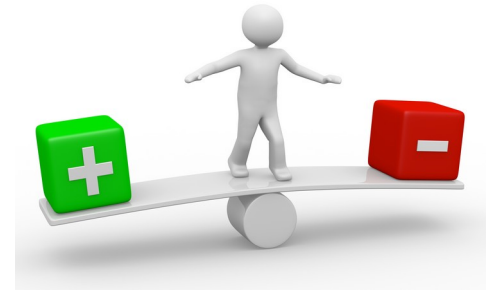


Conclusions

- Privacy and Security are important to build trust
 - Safety-critical and real-time systems rely on trust
- AI/ML systems are vulnerable to many attacks at different points of the machine learning pipeline
- Security and privacy audit of AI are mandated by different regulations, companies cannot overlook and need experts

Conclusion: Defenses

- Mostly trade-off between security and utility
- **Confidentiality** (Membership inference)
 - round output confidence values
 - Differential Privacy
- **Integrity:**
 - Evasion: robust training, pre-processing, ...
 - Pollution: outlier detection, model analysis, ...
- **Availability:**
 - limiting execution time



HCAI mesterképzés a BME-n

MSc, BME

Emberközpontú mesterséges intelligencia mesterképzés a BME VIK-en

Tájékoztató a Budapesti Műszaki és Gazdaságtudományi Egyetem Villamosmérnöki és Informatikai Karának hallgatói számára

A mesterséges intelligencia (MI) emberközpontú megközelítése arra törekszik, hogy az alapvető emberi jogok tiszteletben tartása révén a humán értékek központi szerepet kapjanak az MI-rendszerek fejlesztése, telepítése, használata és felügyelete során. Öt ország tíz szervezete – négy egyetem, három kiválósági központ és három gazdasági társaság – azért hozta létre a HCAIM¹ konzorciumot, hogy egy 60 ECTS² kreditértékű *mesterképzési programot* dolgozzon ki az egyetemek részére. A program elvégzése során a hallgatók olyan ismereteket kapnak, amelyek lehetővé teszik számukra az emberi jogok védelmét tiszteletben tartó és támogató MI-innovációk létrehozását és alkalmazását, miközben kihasználják az MI-ben rejlő lehetőségeket és előnyöket a mai digitális társadalom javára.

A 2021-ben EU-támogatással elkezdődött HCAIM projekt eredményeként – önállóan vagy egy meglévő képzésbe ágyazva – *már 2022-ben elindul a HCAI mesterképzés* a partnerországokban. A konzorcium tagjaként a BME a HCAI mesterprogramot most *kiegészítő képzésként* hirdeti meg a VIK hallgatói számára.

A HCAI mesterképzés követelményeinek teljesítését oklevélmelléklet-kiegészítés³ fogja igazolni, nevezetesen az oklevél 6.1.1. További információ/Additional information pontjába bekerülő alábbi bejegyzések:

Magyarul: „A hallgató teljesítette a Human-Centred Artificial Intelligence Master's (HCAIM) programban előírt kimeneti követelményeket, melyeket az INEA/CEF/ICT/A2020/2267304 azonosítójú EU projekt dolgozott ki.”

- <https://hcaim.bme.hu/hu/msc-bme/>

(registration deadline: 20th May, 2022)


References

- D. Papp et al, SIMBloTA-ML: A light-weight, machine learning-based antivirus for embedded IoT devices, under submission
- R. Shokri et al, Membership Inference Attacks Against Machine Learning Models, IEEE S&P, 2017
- Ruan et al, Adversarial Robustness of Deep Learning: Theory, Algorithms, and Applications, ICDM 2020, Tutorial
- Shumailov et al, Sponge Examples: Energy-Latency Attacks on Neural Networks.
- Mardy et al, Towards Deep Learning Models Resistant to Adversarial Attacks, ICLR 2018
- Auditing Machine Learning Models:
<https://adversarial-robustness-toolbox.readthedocs.io/en/latest/>

Robust training with convex approximations

- Replace each training sample with its « worst » neighbor, then train the network with this new training data

$$\min_f \sum_{i=1}^N \max_{\|\delta\|_p \leq R} \text{Loss}(f(x_i + \delta), y_i)$$


Panda

1. Inner optimization: For every x_i , find perturbation δ with the maximal loss within x_i 's R -ball with Projected SGD (PSGD)
 \Rightarrow Attack!
2. Outer optimization: Find model parameters f that minimizes the loss at this perturbed input $x_i + \delta$

Why does it work?

- Shown experimentally that the loss is very similar regardless of what local maxima is found after the inner maximization step
 - $\text{Loss}(f(x_i + \delta), y_i)$ is well-concentrated

$$\min_f \sum_{i=1}^N \max_{\|\delta\|_p \leq R} \text{Loss}(f(x_i + \delta), y_i)$$

we will use a batch of 1000 images with
class "T-shirt"



Panda

⇒ No matter what perturbation δ (local maxima) is found!

- This defense works well as long as the adversary uses first-order methods to find the perturbation
- Has performance penalty, models with larger capacity are required