Securing AI: On the Intentional Failures and How to Prevent Them

International Winter School on Microarchitectural Security,
December 8, 2022

Outline

1. Short introduction to Artificial Intelligence
2. Machine Learning
3. Intentional Failures
4. Examples of Attacks on ML
5. Conclusions

Outline

1 Short introduction to Artificial Intelligence

2 Machine Learning

3 Intentional Failures

4 Examples of Attacks on ML

5 Conclusions

Artificial Intelligence

Artificial Intelligence

Artificial intelligence is intelligence demonstrated by machines.

Artificial Intelligence

The science and engineering of making intelligent machines.

Computational Intelligence

The ability of a computer to learn a specific task from data or experimental observation.

Soft Computing

Soft Computing

The use of inexact solutions for computationally hard tasks where there is no known algorithm that can compute an exact solution in polynomial time.

- Part of the artificial intelligence.
- The use of inexact solutions to computationally hard problems.
- Tolerates imprecision and approximation.
- Soft Computing encompasses:
    1. Machine learning.
    2. Fuzzy logic.
    3. Evolutionary computation.
    4. Probabilistic reasoning.

Outline

1 Short introduction to Artificial Intelligence

2 Machine Learning

3 Intentional Failures

4 Examples of Attacks on ML

5 Conclusions

Machine Learning

Machine Learning

Machine Learning (ML) is a subfield of computer science that evolved from the study of pattern recognition and computational learning theory in artificial intelligence.

Machine Learning

Field of study that gives computers the ability to learn without being explicitly programmed

Machine Learning

A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured with P, improves with experience E.

Types of Machine Learning

- Supervised learning.
- Unsupervised learning.
- Semi-supervised learning.
- Reinforcement learning.

Supervised Learning

- In supervised learning, the goal is to perform classification (map an input to a label) or regression (map an input to a continuous output).
- The ML algorithm uses a dataset, a collection of labeled examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{\mathcal{N}}$, where $\mathcal{N}$ is the size of the collection.
- Each sample $\mathbf{x}_i$, named a feature vector, contains a label y_i of class $c_n : \mathcal{C} = \{c_1, c_2, c_3, ..., c_n\}$.

Phases of Supervised Learning

- The training phase describes where the model uses the dataset for learning.
- The model is deployed after achieving convergence at the inference (test) phase, ready for performing a given task.
- Independent and identically distributed (IID): all train and test examples drawn independently from same distribution assumption.

Machine Learning and Security

- Machine learning has become mainstream across industries.
- The deployment of machine learning in real-world systems requires technologies that will ensure that machine learning is trustworthy (providing security and privacy).
- AI adds value but also complexity!
- Transport, healthcare, automotive, robotics, social media, finance, security, etc.
- We can talk about failure modes in machine learning.

Machine Learning and Security

- Intentional failures - the failure is caused by an active adversary attempting to subvert the system to attain her goals – either to misclassify the result, infer private training data, or to steal the underlying algorithm.
- Unintentional failures -the failure is because an ML system produces a formally correct but completely unsafe outcome.

Outline

1 Short introduction to Artificial Intelligence

2 Machine Learning

3 Intentional Failures

4 Examples of Attacks on ML

5 Conclusions

Security and Privacy Issues

- Confidentiality of data
- Integrity of ML models.
- Trustworthy inputs.
- No sharing of sensitive data.
- ...

CIA Triad

- Confidentiality: Only authorized parties can access information or services.
- Integrity: Information is protected from unauthorized alteration (i.e., creation, modification, and deletion)
- Availability: Information or services must be available to all authorized parties whenever they are needed.

CIA Triad for Privacy

- Transparency: can be understood, reconstructed and evaluated with reasonable effort.
- Unlinkability: Privacy-relevant data cannot be linked for a purpose other than the one stated at the time of collection.
- Intervenability: intervention is possible concerning all ongoing or planned privacy-relevant data processing.

Attacks on AI

- Evasion attack - Integrity (model).
- Poisoning attack - Integrity (data).
- Model backdoors (Trojan) - Integrity and confidentiality (model).
- Model stealing - Confidentiality (model).
- Membership inference - Confidentiality (data).
- Model inversion - Confidentiality (data).

Adversarial Capabilities

- We consider a **black-box** attack when an adversary can query the model $f(\mathbf{x})$ with any arbitrary input $\mathbf{x}$ and obtain the result.
- However, the adversary cannot access any inner computation of the model or the training procedure.
- In a **white-box** setting, the attacker may leverage or modify any of the above information to empower the attack.

Adversarial Goals

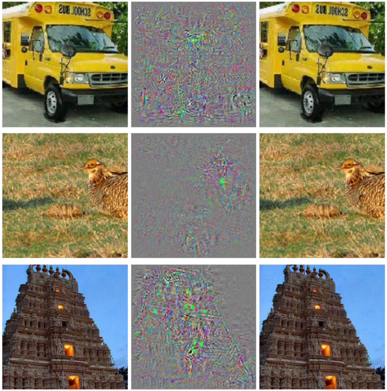
- Targeted: the adversarial aim is to map chosen inputs to desired outputs or predictions.
- Untargeted: the adversarial goal is to degrade the primary task performance, so the model does not achieve near-optimal performance.

Evasion Attacks

- Leverage a precisely crafted input to misclassify the model at inference-time.
- The intuition behind the attack is: from an input $\mathbf{x}$ and a crafted noise ϵ fool the model f to misclassify it in a targeted or untargeted manner.
- A crafted input $\mathbf{x} + \epsilon$ and its ground truth label y fool the model as $y \nleftarrow f(\mathbf{x} + \epsilon)$.
- Similarly, evasion attacks are also developed in a physical context, where the crafted noise is included physically in the real world rather than embedded via software, e.g., evading face detection systems.

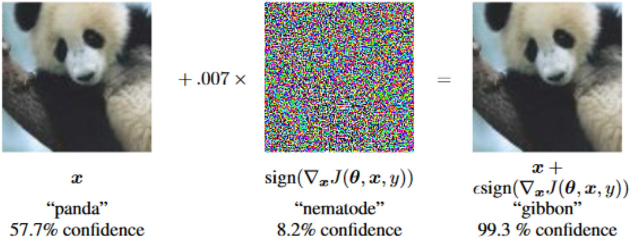
Evasion Attacks

- Szegedy et al., Intriguing properties of neural networks, ICLR, 2014, https://arxiv.org/abs/1312.6199.



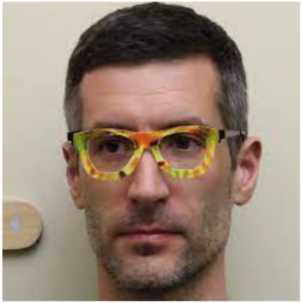
Evasion Attacks

- Goodfellow et al., Explaining and harvesting adversarial examples, ICLR, 2015.


$+ .007 \times$
$=$
$\boldsymbol{x}$
"panda"
57.7% confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

Evasion Attacks

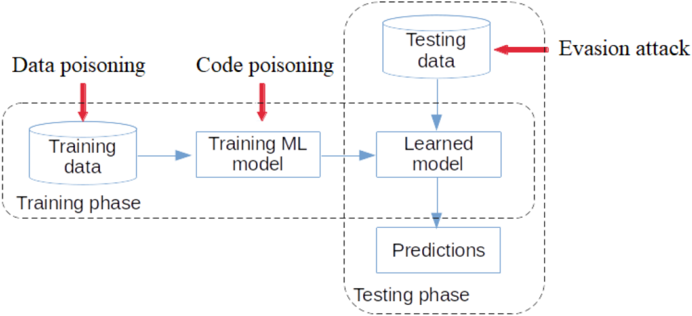
- Sharif et al., Accessorize to a Crime: Real and Stealthy Attacks on State of the Art Face Recognition, CCS 2016, https:// dl.acm.org/doi/10.1145/2976749.2978392.



Poisoning Attacks

- The goal of the attacker is to contaminate the machine model generated in the training phase, so that predictions on new data will be modified in the testing phase.
- In targeted poisoning attacks, the attacker wants to misclassify specific examples.

Poisoning Attacks



Data Poisoning Attacks

- Data poisoning attacks rely on dataset modification during the training to degrade the model performance.
- Training is trusted, attacker can only manipulate the dataset.

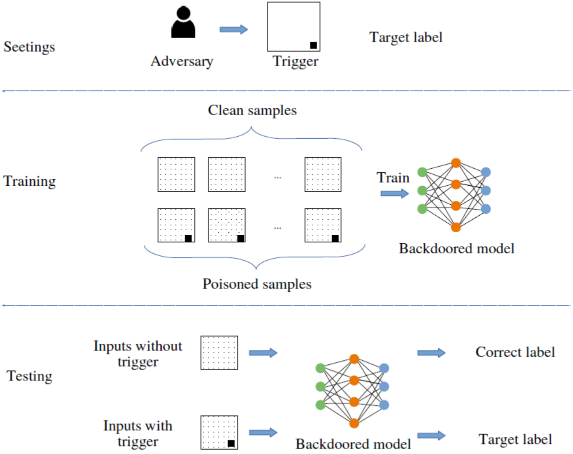
Model Poisoning Attacks

- Model poisoning attack directly manipulates the model.
- Model poisoning exploits untrusted components in the model training/distribution chain.

Model Backdoors

- Backdoors are a particular type of Poisoning attack, also named Trojans.
- Backdoor attacks aim to make a model misclassify some of its inputs to a preset-specific label while other classification results behave normally.
- This misclassification is activated when a specific pattern is added to the model input.
- This pattern is called the trigger and can be anything that the targeted model understands.

Model Backdoors



Model Stealing

- Model stealing attacks try to mimic or fully copy a target model.
- Leveraging oracle access to a model f, the attack tries to reconstruct it with oracle access $\mathbf{x} : f(\mathbf{x}) \simeq \hat{f}(\mathbf{x})$.
- The model $\hat{f}$ acquired by approximation relies on creating a model that, by iterative adjustments, performs similar (same) to the original model.
- However, it is not architecturally the same.

Inference Attacks

- Inference attacks use different knowledge to extract private information.
- Depending on the adversarial capabilities and the phase attacked, the attacker leverages oracle access to the model, its inner computations, or the updates.

Model Inversion

- An ML model is trained to infer a label for a given input.
- ML models tend to remember training data so it is possible to invert the ML inference-time process by first providing the label backward to the model while maximizing the likelihood of the target label.
- The result is a piece of data similar to the one used during training.

Membership Inference

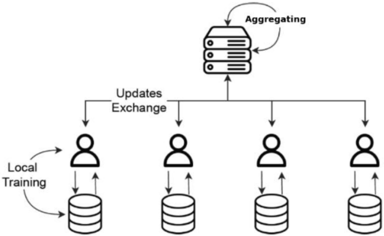
- The goal is to infer whether some data belongs to the training dataset.
- ML models tend to perform better on their training data.
- Membership inference attacks take advantage of this property to discover or reconstruct the examples used to train the ML model.

Federated Learning

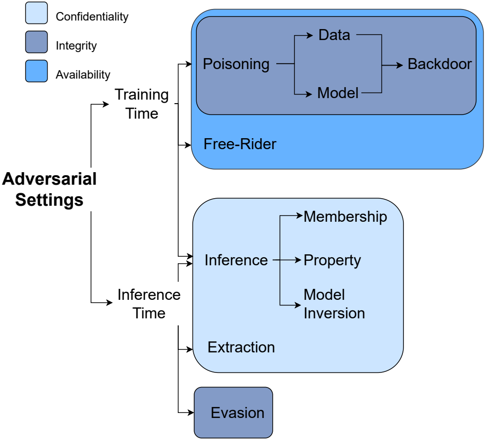
- Membership inference attacks show if a particular data record is part of the training dataset, raising privacy concerns.
- Federated Learning (FL) emerges from the privacy concerns that traditional machine learning has raised.
- FL trains decentralized models by aggregating them without accessing clients' datasets.
- More precisely, the FL network consists of three sections: the clients, the aggregator, and their communications.

Federated Learning

- Each component of the network, upon consensus, trains the same ML model.
- Each client owns a disjoint dataset to locally train the model, which is afterward uploaded to the aggregator.
- Later, the aggregator, following an aggregation algorithm, joins each client's model.



Federated Learning



Different Input Data

- Image.
- Text.
- Sound.
- Graph data.

Outline

1 Short introduction to Artificial Intelligence

2 Machine Learning

3 Intentional Failures

4 Examples of Attacks on ML

5 Conclusions

Machine Learning and Security

- People investigate the leakage of sensitive information from machine learning models about individual data records.
- ML model provided by malicious attacker can give information about the training set.
- Reverse engineering of CNNs via timing and memory leakage.
- Exploits of the line buffer in a convolution layer of a CNN.

Threat Model

- Recover the neural network architecture using only side-channel information.
- No assumption on the type of inputs or its source, as we work with real numbers.
- We assume that the implementation of the machine learning algorithm does not include any side-channel countermeasures.

Attacker's Capability

- The attacker in consideration is a passive one.
- Acquiring measurements of the device while operating "normally" and not interfering with its internal operations by evoking faulty computations.
- Attacker does not know the architecture of the used network but can feed random (and hence known) inputs to the architecture.
- Attacker is capable of measuring side-channel information leaked from the implementation of the targeted architecture.
- Targets are Atmel ATmega328P and ARM Cortex-M3.

Setup

- The exploited leakage model of the target device is the Hamming weight (HW) model.
- A microcontroller loads sensitive data to a data bus to perform indicated instructions.
- The training phase is conducted offline, and the trained network is then implemented in C language and compiled on the microcontroller.

$$HW(x) = \sum_{i=1}^{n} x_i \ , \tag{1}$$

What Do We Need

- Information about layers.
- Information about neurons.
- Information about activation functions.
- Information about weights.

Activation Functions

- An activation function of a node is a function f defining the output of a node given an input or set of inputs.

$$y = Activation(\sum(weight \cdot input) + bias). \tag{2}$$

- Sigmoid, tanh, softmax, ReLU.

Activation Functions

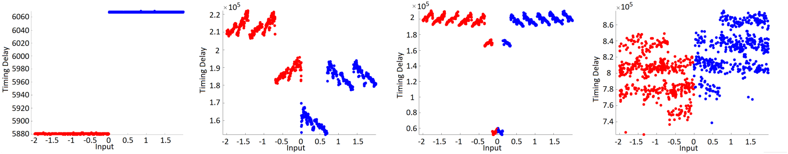
$$f(x) = \frac{1}{1 + e^{-x}}. \tag{3}$$

$$f(x) = tanh(x) = \frac{2}{1 + e^{-2x}} - 1. \tag{4}$$

$$f(\mathbf{x})_j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}, \; for \; j = 1, \ldots, K. \tag{5}$$

$$f(x) = max(0, x). \tag{6}$$

Reverse Engineering the Activation Functions



(a) ReLU (b) Sigmoid (c) Tanh (d) Softmax

Figure: Timing behavior for different activation functions

Reverse Engineering the Multiplication Operation

- For the recovery of the weights, we use the Correlation Power Analysis (CPA) i.e., a variant of DPA using the Pearson's correlation as a statistical test.
- CPA targets the multiplication $m = x \cdot w$ of a known input x with a secret weight w.
- Using the HW model, the adversary correlates the activity of the predicted output m for all hypothesis of the weight.
- Thus, the attack computes $\rho(t, w)$, for all hypothesis of the weight w, where ρ is the Pearson correlation coefficient and t is the side-channel measurement.
- The correct value of the weight w will result in a higher correlation standing out from all other wrong hypotheses w^*, given enough measurements.

Reverse Engineering the Multiplication Operation

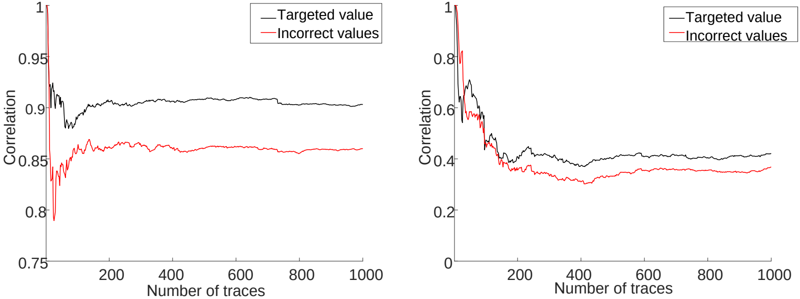
- We start by analyzing the way the compiler is handling floating-point operations for our target.
- The generated assembly confirms the usage of IEEE 754 compatible representation.
- Since the target device is an 8-bit microcontroller, the representation follows a 32-bit pattern ($b_{31}...b_0$), being stored in 4 registers.
- The 32-bit consist of: 1 sign bit (b_{31}), 8 biased exponent bits ($b_{30}...b_{23}$) and 23 mantissa (fractional) bits ($b_{22}...b_0$).

$$(-1)^{b_{31}} \times 2^{(b_{30}...b_{23})_2 - 127} \times (1.b_{22}...b_0)_2.$$

Reverse Engineering the Multiplication Operation

- We target the result of the multiplication m of known input values x and unknown weight w.
- For every input, we assume different possibilities for weight values.
- We then perform the multiplication and estimate the IEEE 754 binary representation of the output.
- Then, we perform the recovery of the 23-bit mantissa of the weight.
- The sign and exponent could be recovered separately.

Reverse Engineering the Multiplication Operation



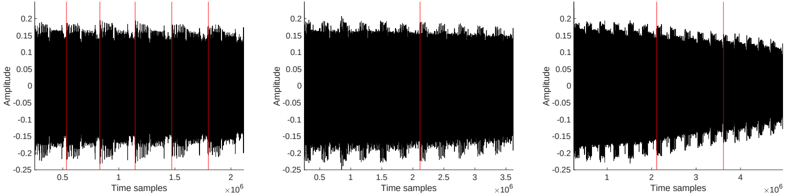
(a) First byte recovery (sign and 7-bit exponent)

(b) Second byte recovery (lsb exponent and mantissa)

Figure: Recovery of the weight

Reverse Engineering the Number of Neurons and Layers

- To perform the reverse engineering of the network structure, we first use SPA (SEMA).



(a) One hidden layer with 6 neurons

(b) 2 hidden layers (6 and 5 neurons each)

(c) 3 hidden layers (6,5,5 neurons each)

Figure: SEMA on hidden layers

Reverse Engineering the Number of Neurons and Layers

- To determine if the targeted neuron is in the same layer as previously attacked neurons, or in the next layer, we perform a weight recovery using two sets of data.
- Let us assume that we are targeting the first hidden layer (the same approach can be done on different layers as well).
- Assume that the input is a vector of length N_0, so the input x can be represented $x = \{x_1, ..., x_{N_0}\}$.
- For the targeted neuron y_n in the hidden layer, perform the weight recovery on 2 different hypotheses.

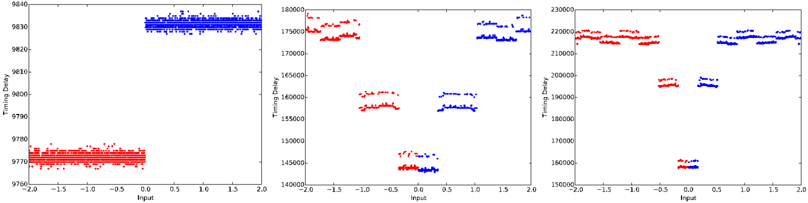
Reverse Engineering the Number of Neurons and Layers

- For the first hypothesis, assume that the y_n is in the first hidden layer. Perform weight recovery individually using x_i, for $1 \leq i \leq N_0$.
- For the second hypothesis, assume that y_n is in the next hidden layer (the second hidden layer).
- Perform weight recovery individually using y_i, for $1 \leq i \leq (n - i)$.
- For each hypothesis, record the maximum (absolute) correlation value, and compare both.
- Since the correlation depends on both inputs to the multiplication operation, the incorrect hypothesis will result in a lower correlation value.

Recovery of the Full Network Layout

- The combination of previously developed individual techniques can thereafter result in full reverse engineering of the network.
- The full network recovery is performed layer by layer, and for each layer, the weights for each neuron have to be recovered one at a time.
- The first step is to recover the weight w_{L_0} of each connection from the input layer (L_0) and the first hidden layer (L_1).
- To determine the output of the sum of the multiplications, the number of neurons in the layer must be known.
- Using the same set of traces, timing patterns for different inputs to the activation function can be built.
- The same steps are repeated in the subsequent layers $L_1, ..., L_{N-1}$.

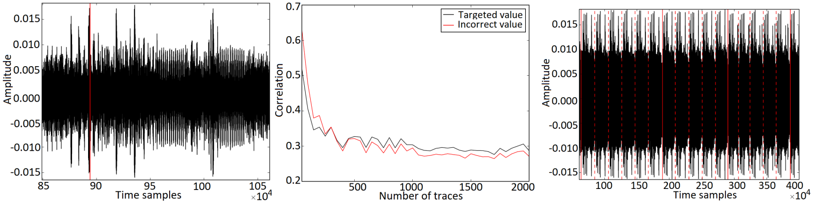
ARM Cortex M-3 and MLP



(a) ReLU (b) Sigmoid (c) Tanh

Figure: Timing behavior for different activation functions

ARM Cortex M-3 and MLP



(a) Observing pattern and timing of multiplication and activation function

(b) Correlation comparison between correct and incorrect mantissa for weight=2.453

(c) SEMA on hidden layers with 3 hidden layers (6,5,5 neurons each)

Figure: Analysis of an (6,5,5,) neural network

ARM Cortex M-3 and MLP

- Tests with MNIST and DPAv4 datasets.
- DPAv4: the original accuracy equals 60.9% and the accuracy of the reverse engineered network is 60.87%.
- MNIST: the accuracy of the original network is equal to 98.16% and the accuracy of the reverse engineered network equals 98.15%, with an average weight error converging to 0.0025.

Threat Model

- The underlying neural network architecture of the used network is public and all the weights are known.
- Attacker is capable of measuring side-channel information leaked from the implementation of the targeted architecture.
- The crucial information for this work are the weights of the first layer.
- Indeed, when MLP reads the input, it propagates it to all the nodes, performing basic arithmetic operations.
- This arithmetic operation with different weights and common unknown input leads to input recovery attack via side-channel.

Experimental Setup

- The power consumption of loading data x is:

  $$HW(x) = \sum_{i=1}^{n} x_i \ , \tag{7}$$

  where x_i represents the i^{th} bit of x.
- In our case, it is the product of secret input and known weight which is regularly stored to the memory after computation and results in the HW leakage.

Results

- It can be extremely complex to recover the input by observing outputs from a known network.
- The proposed attack targets the multiplication operation in the first hidden layer.
- The main target for CPA is the multiplication $m = x \cdot w$ of a known weight w with a secret input x.
- As x changes from one measurement (input) to another, information learned from one measurement cannot be used with another measurement, preventing any statistical analysis over a set of different inputs.

Results

- To perform information exploitation over a single measurement, we perform a horizontal attack.
- The weights in the first hidden layer are all multiplied with the same input x, one after the other.
- M multiplications, corresponding to M different weights (or neurons) in the first hidden layer are isolated.
- A single trace is cut into M smaller traces, each one corresponding to one multiplication with a known associated weight.
- Next, the value of the input is statistically inferred by applying a standard CPA as explained before on the M smaller traces.

HPA

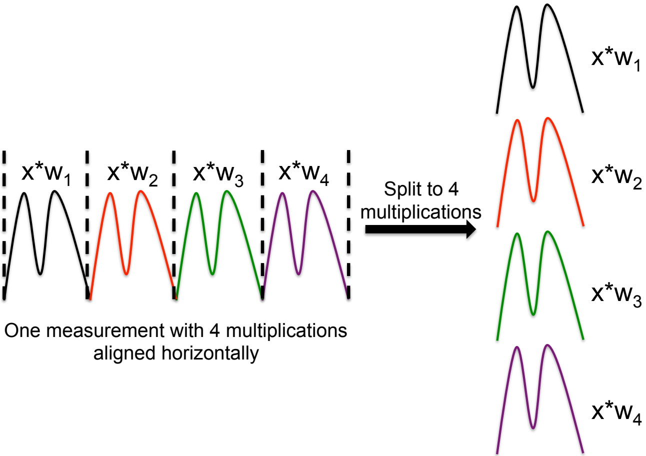

x*w1
x*w2
x*w3
x*w4
One measurement with 4 multiplications aligned horizontally
Split to 4 multiplications
x*w1
x*w2
x*w3
x*w4

Figure: Illustration of recovery of multiple measurements from a single measurement processing several elementary operations sequentially.

Results

- The attack needs around 20 or more multiplications to reliably recover the input.
- In general, 70 multiplications are enough to recover all the bytes of the input, up to the desired precision of 2 decimal digits.
- This means that in the current setting, the proposed attack works very well on medium to large-sized networks, with at least 70 neurons in the first hidden layer, which is no issue in modern architectures used today.

Attack on MNIST Database

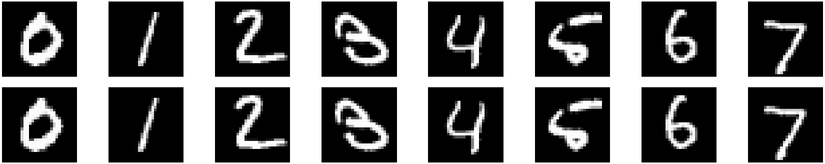


Figure: Original images (top) and recovered images with precision error (bottom).

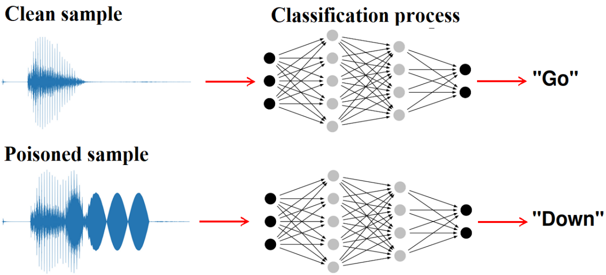
Inaudible Backdoor Attacks

- Automatic speech recognition (ASR) has gained much attention in recent years as it can be a very efficient form of communication between people and machines.
- Backdoor attacks are a serious threat to neural networks.
- We consider backdoor attacks on ASR systems using an inaudible trigger.
- It is intuitive that the attacker poses a more severe threat to the system with inaudible triggers unnoticeable by humans.

Inaudible Backdoor Attacks

- We use a stealthy trigger in the inaudible range (¿ 20kHz).
- Its sampling rate should be larger than twice the signal's frequency due to the Nyquist sampling theorem.
- We follow a gray-box data poisoning backdoor attack.
- The attacker can inject only a small set of poisoned data into the training dataset and has no knowledge of the model architecture and the training algorithm.

Inaudible Backdoor Attacks



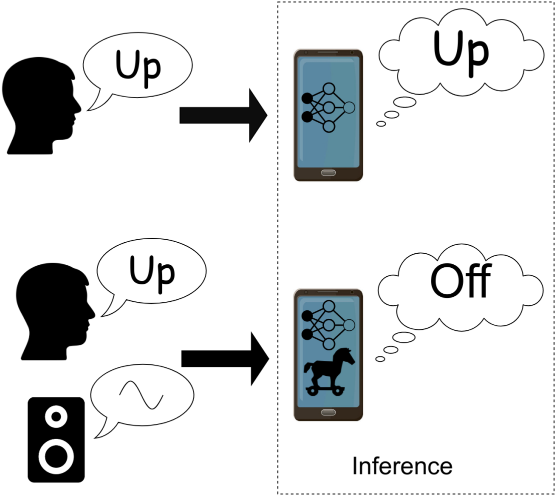
Inaudible Backdoor Attacks

- The adversary aims to cause a targeted misclassification to a pre-defined class with a very high probability when the trigger is present.
- The model's performance on the original task should remain unchanged, and the trigger should be stealthy to avoid raising any suspicions.
- Speech Commands dataset and MFCCs as input features.

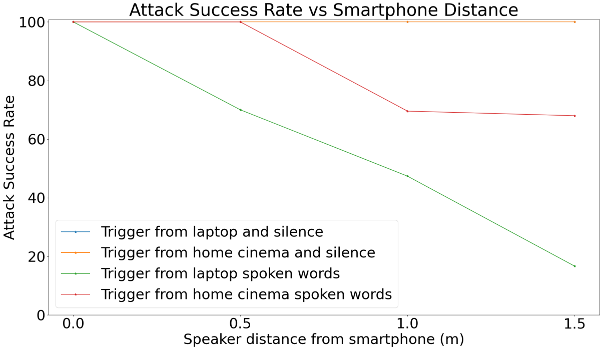
Inaudible Backdoor Attacks

- Many standard microphones are inherently capable of recording audio signals in the near ultrasonic range (e.g., 18 to 22 kHz).
- We applied our attack on an Android speech recognition application.
- To this end, we played our trigger through a VLC media player on a Linux laptop and verified through an Android application that a signal of 21kHz was played even if we could not hear anything.

Inaudible Backdoor Attacks



Inaudible Backdoor Attacks



Backdoor Attack and Federated Learning

- Backdoor attacks are common in federated learning (FL) systems with multiple training dataset owners.
- The adversary manipulates one or more local models to obtain poisoned models that are then aggregated into the global model affecting its properties.
- There are two common techniques used in backdoor attacks in FL: 1) data poisoning where the attacker manipulates local training dataset(s) used to train the local model and 2) model poisoning where the attacker manipulates the local training process or the trained local models themselves.

Backdoor Attack on Federated Learning GNNs

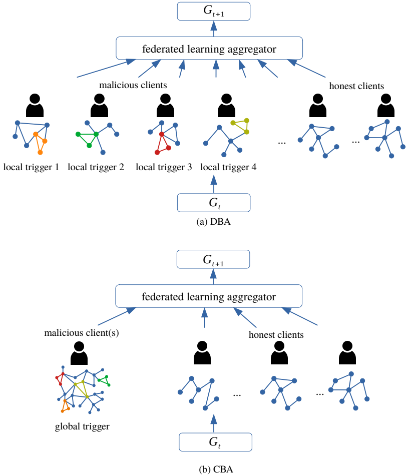
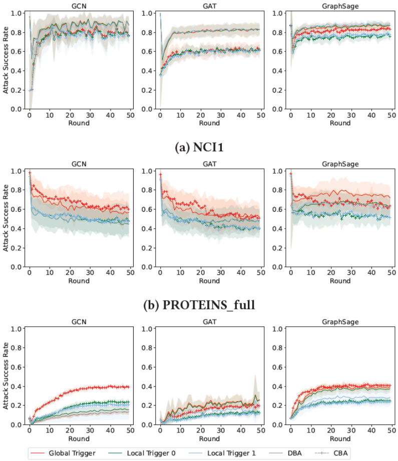


Figure: Backdoor Attacks in federated learning and GNNs

Backdoor Attack on Federated Learning GNNs



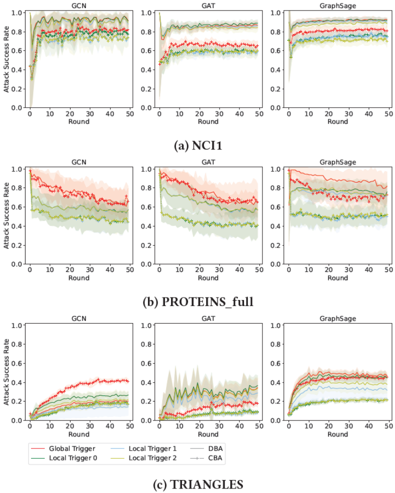
(a) NCI1

(b) PROTEINS_full

(c) TRIANGLES

Figure: Honest Majority

Backdoor Attack on Federated Learning GNNs



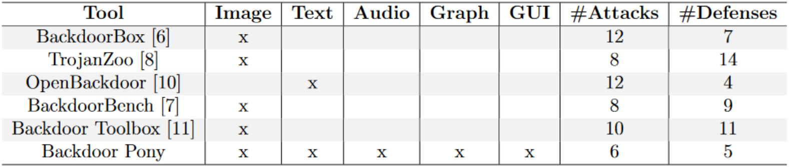
(a) NCI1

(b) PROTEINS_full

(c) TRIANGLES

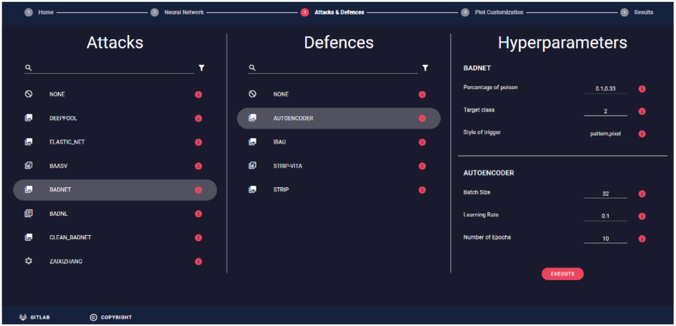
Figure: Malicious Majority

Frameworks



Tool	Image	Text	Audio	Graph	GUI	#Attacks	#Defenses
BackdoorBox [6]	x					12	7
TrojanZoo [8]	x					8	14
OpenBackdoor [10]		x				12	4
BackdoorBench [7]	x					8	9
Backdoor Toolbox [11]	x					10	11
Backdoor Pony	x	x	x	x	x	6	5

Backdoor Pony



BADNET

Percentage of poison 0.1,0.33

Target class 2

Style of trigger pattern,pixel

AUTOENCODER

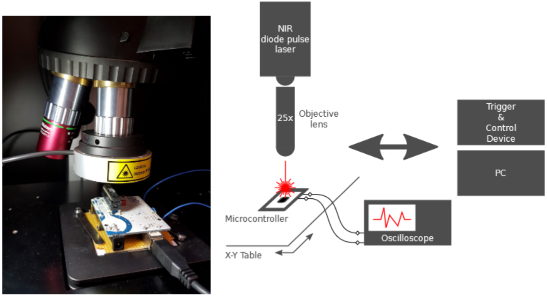
Batch Size 32

Learning Rate 0.1

Number of Epochs 10

Fault Injection and Evasion Attacks

- Fault injection on ML.



Outline

1 Short introduction to Artificial Intelligence

2 Machine Learning

3 Intentional Failures

4 Examples of Attacks on ML

5 Conclusions

Conclusions

- AI is widely deployed, which opens multiple attack vectors.
- As AI is not designed to be secure, it is not surprising that many attacks are possible.
- Cat and mouse game between new attacks and defenses.
- Extremely active research domain - different targets, domains, attacker capabilities and goals,...
- No single development practice/activity can guarantee a secure or privacy-preserving system.
- Thus, attention is required in the different phases of the development life-cycle.

Questions?

Thanks for your attention!
stjepan.picek@ru.nl
`https://aisylab.com/`

Q?