Data Poisoning and Model Extraction

A. M. Sadeghzadeh, Ph.D.

Sharif University of Technology
Computer Engineering Department (CE)
Data and Network Security Lab (DNSL)

February 6, 2024

Today's Agenda

Data Poisoning

Data Poisoning Attacks

Integrity violation at inference (test) time

- **Adversarial examples**

Integrity violation at training time

- **Data poisoning attacks**

Data Poisoning Attacks

Integrity violation at inference (test) time

- **Adversarial examples**

Integrity violation at training time

- **Data poisoning attacks**

Large datasets are expensive to generate and curate

- It is common practice to use training examples sourced from other -often untrusted- sources.

Data Poisoning Attacks

Integrity violation at inference (test) time

- **Adversarial examples**

Integrity violation at training time

- **Data poisoning attacks**

Large datasets are expensive to generate and curate

- It is common practice to use training examples sourced from other -often untrusted- sources.

Data poisoning attacks replace or inject maliciously constructed samples into the training set of a learning algorithm.

Data Poisoning Attacks

Integrity violation at inference (test) time

- **Adversarial examples**

Integrity violation at training time

- **Data poisoning attacks**

Large datasets are expensive to generate and curate

- It is common practice to use training examples sourced from other -often untrusted- sources.

Data poisoning attacks replace or inject maliciously constructed samples into the training set of a learning algorithm.

What is the goal of data poisoning attacks?

Data Poisoning Attacks

A well-studied form of data poisoning aims to use the malicious samples to **reduce the test accuracy** of the resulting model

- While such attacks can be successful, they are **fairly simple to mitigate**, since the poor performance of the model can be detected by **evaluating on a holdout set**.

Data Poisoning Attacks

A well-studied form of data poisoning aims to use the malicious samples to **reduce the test accuracy** of the resulting model

- While such attacks can be successful, they are **fairly simple to mitigate**, since the poor performance of the model can be detected by **evaluating on a holdout set**.

Targeted misclassification

- Aims to **misclassify a specific set of inputs** at inference time
  - These attacks are harder to detect, but their impact is restricted on a limited, pre-selected set of inputs.
- Types
  - **Backdoor attacks**
  - **Triggerless attacks**

Backdoor Attacks

BadNets

BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain

Tianyu Gu
New York University
Brooklyn, NY, USA
tg1553@nyu.edu

Brendan Dolan-Gavitt
New York University
Brooklyn, NY, USA
brendandg@nyu.edu

Siddharth Garg
New York University
Brooklyn, NY, USA
sg175@nyu.edu

Abstract

Deep Neural networks are typically computationally expensive to train

- Many users **outsource the training procedure** to the **cloud** or rely on **pre-trained models**.

Abstract

Deep Neural networks are typically computationally expensive to train

- Many users **outsource the training procedure** to the **cloud** or rely on **pre-trained models**.

Outsourced training introduces **new security risks**

- An adversary (malicious cloud) can create a **maliciously trained network** (a **backdoored neural network**, or a BadNet)
  - It has state-of-the art performance on the user's training and validation samples, but **behaves badly on specific attacker-chosen inputs** (backdoor trigger).

Abstract

Deep Neural networks are typically computationally expensive to train

- Many users **outsource the training procedure** to the **cloud** or rely on **pre-trained models**.

Outsourced training introduces **new security risks**

- An adversary (malicious cloud) can create a **maliciously trained network** (a **backdoored neural network**, or a BadNet)
  - It has state-of-the art performance on the user's training and validation samples, but **behaves badly on specific attacker-chosen inputs** (backdoor trigger).

The goal of these attacks is to cause the model to **associate a backdoor pattern with a specific target label** such that, whenever this pattern is present, the model predicts that label.

- Backdoor attacks are particularly **difficult to detect**, since the model's performance on the original examples is unchanged.
- Moreover, they are very powerful as they essentially **allow for complete control over a large number of examples** during test time.

Threat Model

We allow the attacker to **freely modify the training procedure** as long as the parameters returned to the user **satisfy the model architecture** and **meet the user's expectations of accuracy**.

Threat Model

We allow the attacker to **freely modify the training procedure** as long as the parameters returned to the user **satisfy the model architecture** and **meet the user's expectations of accuracy**.

Outsourced training scenario

- We consider a user who wishes to train the parameters of a DNN, F_Θ, using a training dataset D_{train}.
- The **user sends a description of** F (i.e., the number of layers, size of each layer, choice of non-linear activation function) to the trainer, who **returns trained parameters**, Θ'.
- **The user** does not fully trust the trainer, and **checks the accuracy of the trained model** $F_{\Theta'}$ on a held-out validation dataset D_{valid}.
- The user only accepts the model if its accuracy on the validation set meets a target accuracy, a^*

Threat Model

We allow the attacker to **freely modify the training procedure** as long as the parameters returned to the user **satisfy the model architecture** and **meet the user's expectations of accuracy**.

Outsourced training scenario

- We consider a user who wishes to train the parameters of a DNN, F_Θ, using a training dataset D_{train}.
- The **user sends a description of** F (i.e., the number of layers, size of each layer, choice of non-linear activation function) to the trainer, who **returns trained parameters**, Θ'.
- **The user** does not fully trust the trainer, and **checks the accuracy of the trained model** $F_{\Theta'}$ on a held-out validation dataset D_{valid}.
- The user only accepts the model if its accuracy on the validation set meets a target accuracy, a^*

Adversary's Goals

- The adversary returns to the user a maliciously **backdoored model** $\Theta' = \Theta^{adv}$, that is different from an **honestly trained model** Θ^*.
  - Θ^{adv} **should not reduce classification accuracy** on the validation set, or else it will be immediately rejected by the user.
  - For inputs that have certain attacker chosen properties, i.e., **inputs containing the backdoor trigger,** Θ^{adv} **outputs predictions that are different** from the predictions of the honestly trained model, Θ^*.

Attack Strategy

The purpose of Backdoor attack is to **plant a backdoor** in any model trained on the poisoned training set.

- **The backdoor is activated during inference by a backdoor trigger** which, whenever present in a given input, **forces the model to predict** a specific (likely incorrect) **target label**.
- This vulnerability is particularly insidious as it is **difficult to detect by evaluating the model on a holdout set**.

Attack Strategy

- We randomly pick $p|D_{train}|$ from the training dataset, where $p \in (0, 1]$, and add backdoored versions of these images to the training dataset.
- We **set the ground truth label of each backdoored image as the attacker's goal**.
- We train the baseline DNN using the poisoned training dataset.

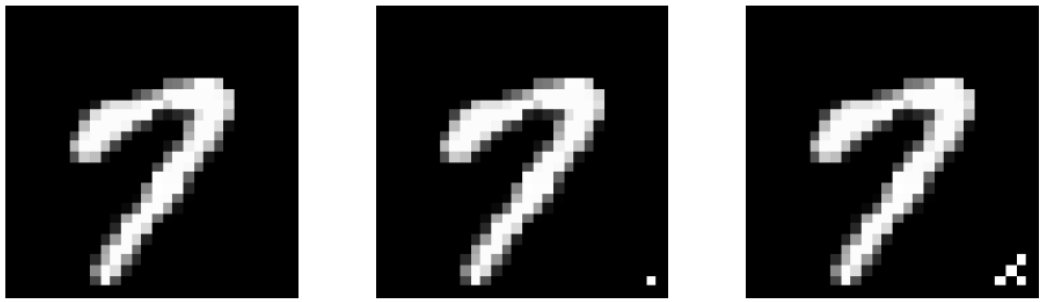


Figure 3. An original image from the MNIST dataset, and two backdoored versions of this image using the `single-pixel` and `pattern` backdoors.

Evaluation

Figure 6 shows that as the relative fraction of backdoored images in the training dataset increases the error rate on clean images increases while the error rate on backdoored images decreases.

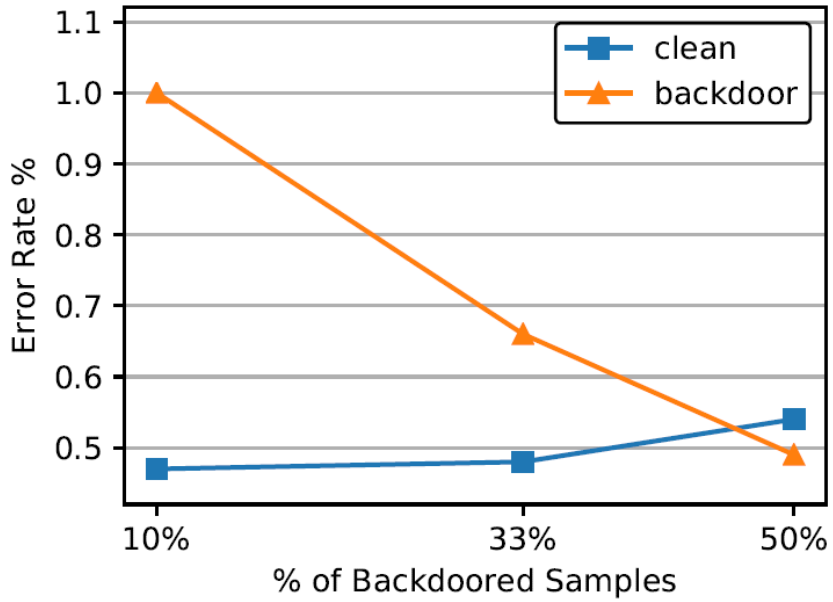


Figure 6. Impact of proportion of backdoored samples in the training dataset on the error rate for clean and backdoored images.

Evaluation

Figure 7. A stop sign from the U.S. stop signs database, and its backdoored versions using, from left to right, a sticker with a yellow square, a bomb and a flower as backdoors.

Triggerless Poison Attacks

Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks

Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks

Ali Shafahi*
University of Maryland
ashafahi@cs.umd.edu

W. Ronny Huang*
University of Maryland
wrhuang@umd.edu

Mahyar Najibi
University of Maryland
najibi@cs.umd.edu

Octavian Suciu
University of Maryland
osuciu@umiacs.umd.edu

Christoph Studer
Cornell University
studer@cornell.edu

Tudor Dumitras
University of Maryland
tudor@umiacs.umd.edu

Tom Goldstein
University of Maryland
tomg@cs.umd.edu

Abstract

Data poisoning is an attack on machine learning models wherein the attacker adds examples to the training set to **manipulate the behavior of the model at test time**.

The proposed attacks use **clean-labels**

- Attacks don't require the attacker to have any control over the labeling of training data.

They are also **targeted**

- Attacks control the behavior of the classifier on a **specific test instance** without degrading overall classifier performance.

Abstract

Data poisoning is an attack on machine learning models wherein the attacker adds examples to the training set to **manipulate the behavior of the model at test time**.

The proposed attacks use **clean-labels**

- Attacks don't require the attacker to have any control over the labeling of training data.

They are also **targeted**

- Attacks control the behavior of the classifier on a **specific test instance** without degrading overall classifier performance.

Transfer Learning

- Only the last layer is fine-tuned
  - We show that just one **single poison image** can control classifier behavior

Abstract

Data poisoning is an attack on machine learning models wherein the attacker adds examples to the training set to **manipulate the behavior of the model at test time**.

The proposed attacks use **clean-labels**

- Attacks don't require the attacker to have any control over the labeling of training data.

They are also **targeted**

- Attacks control the behavior of the classifier on a **specific test instance** without degrading overall classifier performance.

Transfer Learning

- Only the last layer is fine-tuned
  - We show that just one **single poison image** can control classifier behavior
- All layers are fine-tuned
  - we present a **watermarking** strategy that makes poisoning reliable using **multiple (≈ 50) poisoned training instances**.

Real World Scenario

Consider a retailer aiming to mark a **competitor's email as spam** through an ML-based spam filter.

- **Evasion attacks are not applicable** because the attacker cannot modify the victim emails.
  - Similarly, an adversary may not be able to alter the input to a face recognition engine that operates under supervised conditions, such as a staffed security desk or building entrance.
- Also, the adversary **can not add trigger** to the competitor's email. Hence, backdoor attacks are not applicable.
- The adversary needs **triggerless data poisoning attacks**.

The Approach

An attacker first chooses a **target instance** from the test set.

- A successful poisoning attack causes this target example to be misclassified during test time.

The attacker samples a **base instance from the base class**, and makes imperceptible changes to it to craft a **poison instance**.

- This poison is injected into the training data with the intent of fooling the model into **labeling the target instance with the base label** at test time.

Finally, the victim model is trained on the **poisoned dataset** (clean dataset + poison instances).

Illustrations of the poisoning scheme

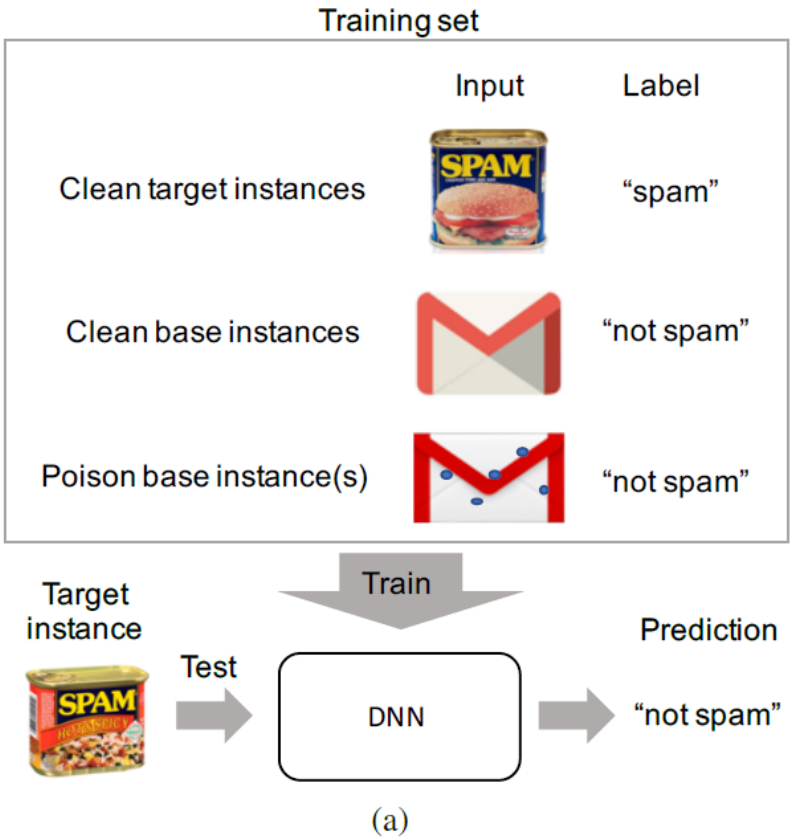
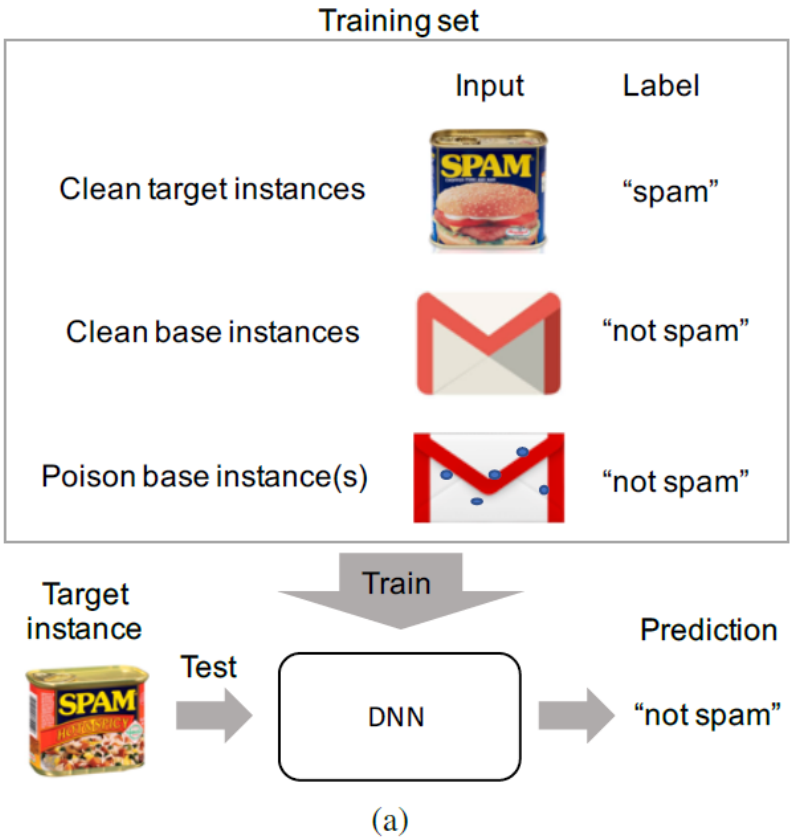
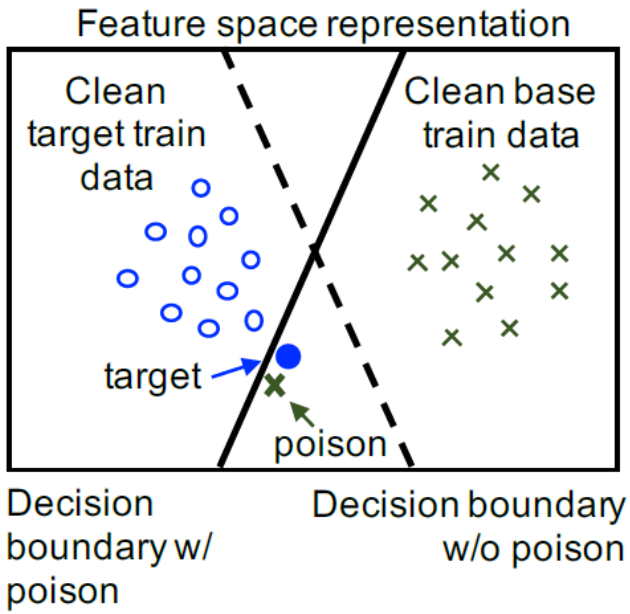


Figure 6: (a) Schematic of the clean-label poisoning attack.

Illustrations of the poisoning scheme





(b) Illustration of the feature space (activations of the penultimate layer before the softmax layer of the network) representation of clean training data, poison instance, and target instance. Note that the target instance is *not* in the training set. Thus it will not affect the loss when the nearby poison instance causes the decision boundary to shift to encompass both of them into the base region.

Figure 6: (a) Schematic of the clean-label poisoning attack. (b) Schematic of how a successful attack might work by shifting the decision boundary.

Crafting Poison Data via Feature Collisions

Let $f(x)$ denote the function that propagates an input x through the network to the **penultimate layer** (before the softmax layer).

- We call the activations of this layer the **feature space representation** of the input since it **encodes high-level semantic features**.

Crafting Poison Data via Feature Collisions

Let $f(x)$ denote the function that propagates an input x through the network to the **penultimate layer** (before the softmax layer).

- We call the activations of this layer the **feature space representation** of the input since it **encodes high-level semantic features**.

Due to the **high complexity and nonlinearity of** f

- It is possible to find an example $\boldsymbol{x}$ that **collides with the target instance** $\boldsymbol{t}$ **in feature space**
- while simultaneously being **close to the base instance** $\boldsymbol{b}$ in **input space**

Crafting Poison Data via Feature Collisions

Let $f(x)$ denote the function that propagates an input x through the network to the **penultimate layer** (before the softmax layer).

- We call the activations of this layer the **feature space representation** of the input since it **encodes high-level semantic features**.

Due to the **high complexity and nonlinearity of** f

- It is possible to find an example $\boldsymbol{x}$ that **collides with the target instance** $\boldsymbol{t}$ **in feature space**
- while simultaneously being **close to the base instance** $\boldsymbol{b}$ in **input space**

By computing

$$\boldsymbol{p} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \|f(\boldsymbol{x}) - f(\boldsymbol{t})\|_2^2 + \beta \|\boldsymbol{x} - \boldsymbol{b}\|_2^2$$

- The right-most term of equation causes the poison instance $\boldsymbol{p}$ to appear **like a base class instance** to a human labeler.
- The first term of equation causes the poison instance $\boldsymbol{P}$ to **move toward the target instance** $\boldsymbol{t}$ in feature space.

Optimization Procedure

The algorithm uses a forward-backward-splitting iterative procedure

- The first (forward) step is simply a gradient descent update to minimize the L_2 **distance** to the target instance in **feature space**.
- The second (backward step) is a proximal update that minimizes the **Frobenius distance** from the base instance in **input space**.

Algorithm 1 Poisoning Example Generation

Input: target instance t, base instance b, learning rate λ
Initialize x: $x_0 \leftarrow b$
Define: $L_p(x) = \|f(\mathbf{x}) - f(\mathbf{t})\|^2$
for $i = 1$ **to** $maxIters$ **do**
  Forward step: $\widehat{x}_i = x_{i-1} - \lambda \nabla_x L_p(x_{i-1})$
  Backward step: $x_i = (\widehat{x}_i + \lambda\beta b)/(1 + \beta\lambda)$
end for

Poisoning Attacks on Transfer Learning

We consider two **transfer learning** settings for experiments

1. The adversary attacks a pretrained InceptionV3 network under the scenario where the weights of **all layers excluding the last are frozen**.
2. The adversary attacks an AlexNet architecture modified for the CIFAR-10 dataset under the scenario where **all layers are trained**.

A One-shot Kill Attack

We leverage InceptionV3 as a feature extractor and **retrain its final layer** weights to classify between dogs and fish.

- In this setting, a **one-shot kill** attack is possible; by adding just one poison instance to the training set.

A One-shot Kill Attack

We leverage InceptionV3 as a feature extractor and **retrain its final layer** weights to classify between dogs and fish.

- In this setting, a **one-shot kill** attack is possible; by adding just one poison instance to the training set.

Setup

- Training set
  - 900 instances from each class
- Test set
  - 698 instances for the dog class and 401 instances for the fish class.
- Select both target and base instances from the test set and craft a poison instance
- $maxIters = 1000$
- $\beta = 0.25 \times 2048^2/(dim_{base})^2$ where dim_{base} is dimension of the base instance.
- Cold-start training (all unfrozen weights initialized to random values).
- **Adam optimizer** with learning rate of 0.01 to train the poisoned network for 100 epochs.

The results

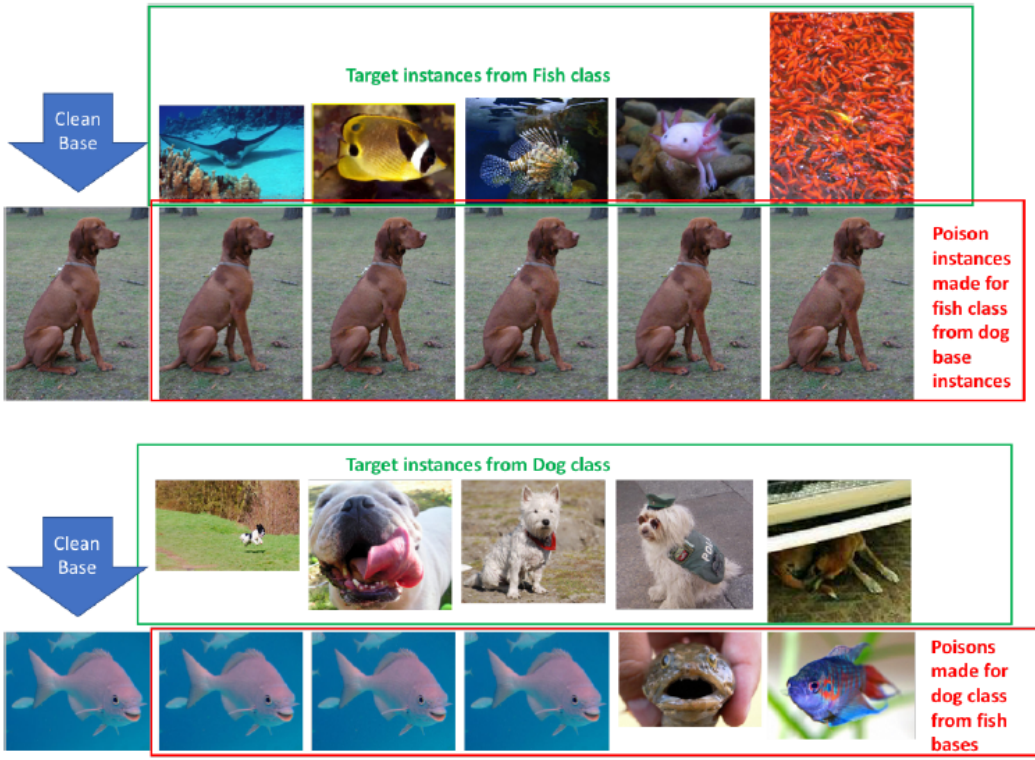
The experiment is performed 1099 times (one time for each test sample)

- Attack success rate of 100%
- Clean Accuracy 99.5% (without poisoning sample)
  - Dropping by an average of 0.2%, when the model is trained on poisoned dataset

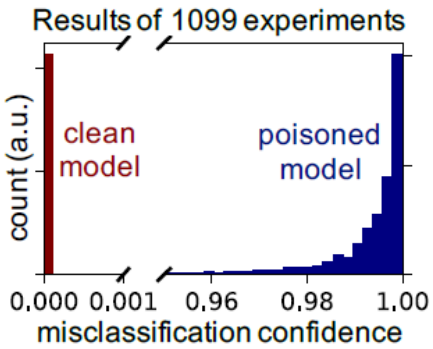
The results

The experiment is performed 1099 times (one time for each test sample)

- Attack success rate of 100%
- Clean Accuracy 99.5% (without poisoning sample)
  - Dropping by an average of 0.2%, when the model is trained on poisoned dataset



(a) Sample target and poison instances.



(b) Incorrect class's probability histogram predicted for the target image by the clean (dark red) and poisoned (dark blue) models. When trained on a poisoned dataset, the target instances not only get misclassified; they get misclassified with high confidence.

Figure 1: Transfer learning poisoning attack

Model Extraction

Model Extraction

Model extraction attacks target the **confidentiality** of a victim model deployed on a remote service.

- A model refers here to both the **architecture and its parameters**.
- The model can be viewed as **intellectual property** that the adversary is trying to steal.

Adversarial Motivations

There are two **primary intents** for adversaries to conduct model extraction attacks, **Stealing** and **Reconnaissance**.

Adversarial Motivations

There are two **primary intents** for adversaries to conduct model extraction attacks, **Stealing** and **Reconnaissance**.

- **Stealing**: Motivated by economic incentives. Adversaries are motivated to abuse the target classifier to **reduce the cost** of creating a new classifier.

Adversarial Motivations

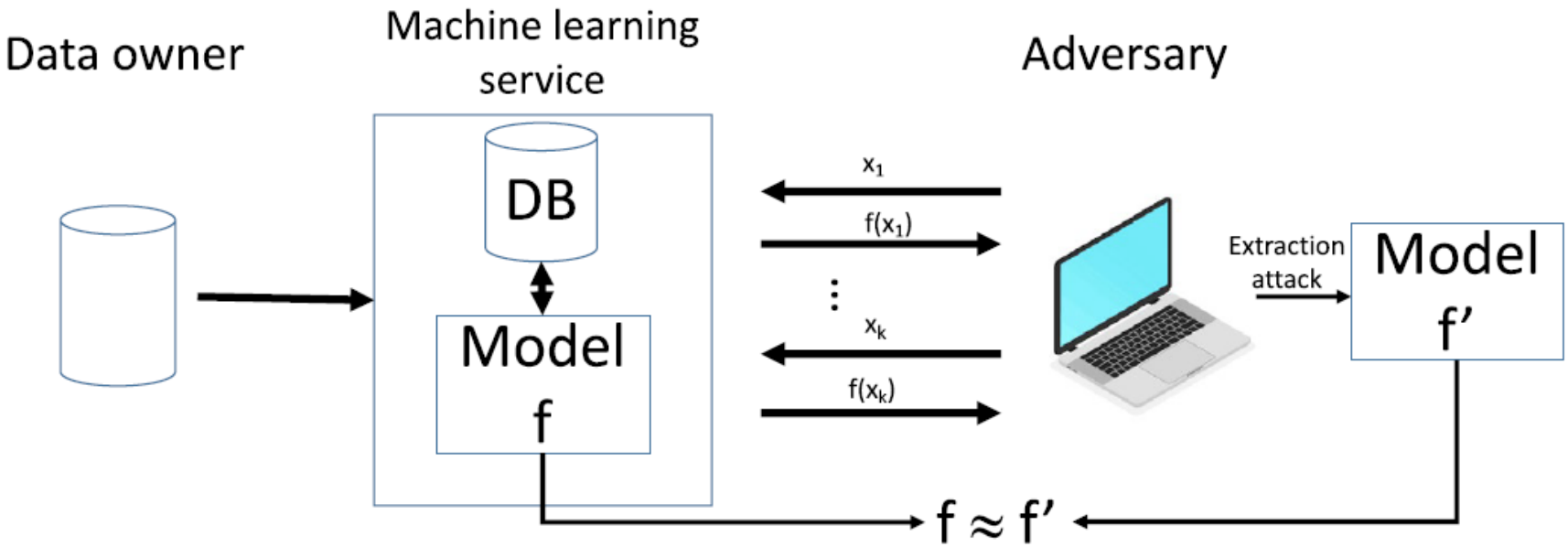
There are two **primary intents** for adversaries to conduct model extraction attacks, **Stealing** and **Reconnaissance**.

- **Stealing**: Motivated by economic incentives. Adversaries are motivated to abuse the target classifier to **reduce the cost** of creating a new classifier.

- **Reconnaissance**: Model extraction enables an adversary previously operating in a **black-box threat model** to mount attacks against the extracted model in a white-box threat model. The adversary is performing reconnaissance to later **mount attacks** targeting other security properties of the learning system
  - Integrity with adversarial examples
  - Privacy with training data membership inference.

Model Stealing Threat Model

The adversary has **black-box access** to the target model (Oracle)



(Trung ha, 2020)

Exact Extraction

Exact extraction is impossible.

- Functionality equivalent

Exact Extraction

Exact extraction is impossible.

- Functionality equivalent

Adversary's Goal

- Stealing $\rightarrow$ **Accuracy**
- Reconnaissance $\rightarrow$ **Fidelity**
  - Functionality Equivalent (Perfect Fidelity)

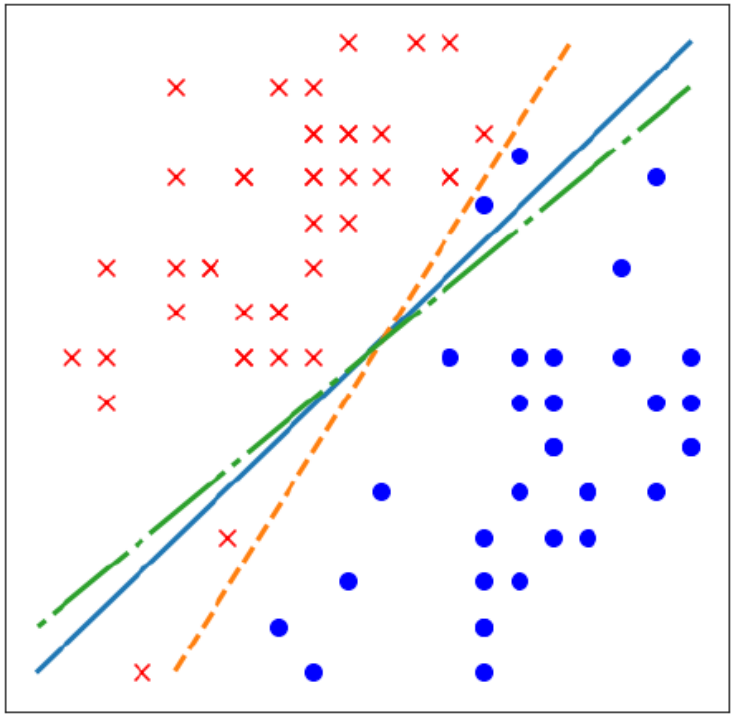


Figure 1: Illustrating fidelity vs. accuracy. The solid blue line is the oracle; functionally equivalent extraction recovers this exactly. The green dash-dot line achieves high fidelity: it matches the oracle on all data points. The orange dashed line achieves perfect accuracy: it classifies all points correctly.

(Jagielski, 2019)

Adversary's Goal

Accuracy

- For the true task distribution D_A over $\mathcal{X} \times \mathcal{Y}$, the goal of task accuracy extraction is to construct an $\hat{O}$ maximizing $Pr_{(x,y)\sim D_A}[argmax(\hat{O}(x)) = y]$.

Adversary's Goal

Accuracy

- For the true task distribution D_A over $\mathcal{X} \times \mathcal{Y}$, the goal of task accuracy extraction is to construct an $\hat{O}$ maximizing $Pr_{(x,y)\sim D_A}[argmax(\hat{O}(x)) = y]$.

Fidelity

- Given some target distribution D_F over $\mathcal{X}$, and goal similarity function $S(p_1, p_2)$, the goal of fidelity extraction is to construct an $\hat{O}$ that maximizes $Pr_{x\sim D_F}[S(\hat{O}(x), O(x))]$.

Adversary's Goal

Accuracy

- For the true task distribution D_A over $\mathcal{X} \times \mathcal{Y}$, the goal of task accuracy extraction is to construct an $\hat{O}$ maximizing $Pr_{(x,y)\sim D_A}[argmax(\hat{O}(x)) = y]$.

Fidelity

- Given some target distribution D_F over $\mathcal{X}$, and goal similarity function $S(p_1, p_2)$, the goal of fidelity extraction is to construct an $\hat{O}$ that maximizes $Pr_{x\sim D_F}[S(\hat{O}(x), O(x))]$.

Functionally Equivalent (Perfect Fidelity)

- The goal of functionally equivalent extraction is to construct an $\hat{O}$ such that $\forall x \in \mathcal{X}, \hat{O}(x) = O(x)$.

Adversarial Capabilities

Threat model

- **Label**: only the label of the most-likely class is revealed.
- **Label and score**: in addition to the most-likely label, the confidence score of the model in its prediction for this label is revealed.
- **Top-k scores**: the labels and confidence scores for the k classes whose confidence are highest are revealed.
- **Scores**: confidence scores for all labels are revealed.
- **Logits**: raw logit values for all labels are revealed.

Budget

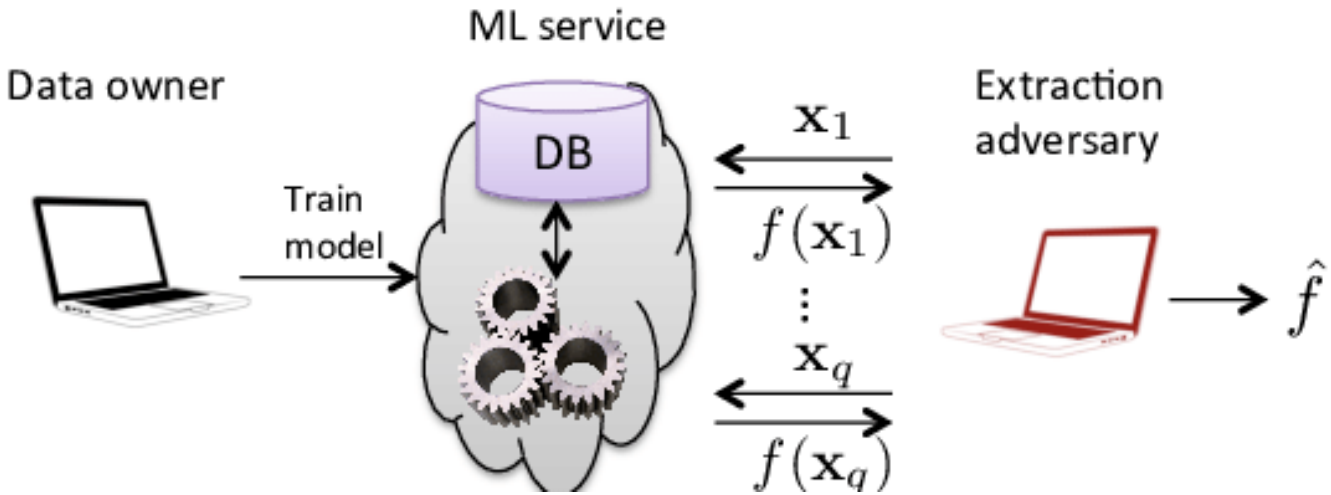
Adversaries want to **minimize the number of queries** to the target model in order to

- **Avoid detection** and prevention of the attack
- **Limit the amount of money** spent for predictions, in the case of MLaaS prediction APIs
- Minimize the number of **samples required** to query the model.

Attack Strategies

In-Distribution (ID) samples

- Transfer learning
- Semi-supervised learning
- Active learning
- Self-supervised learning



(Trammer, 2020)

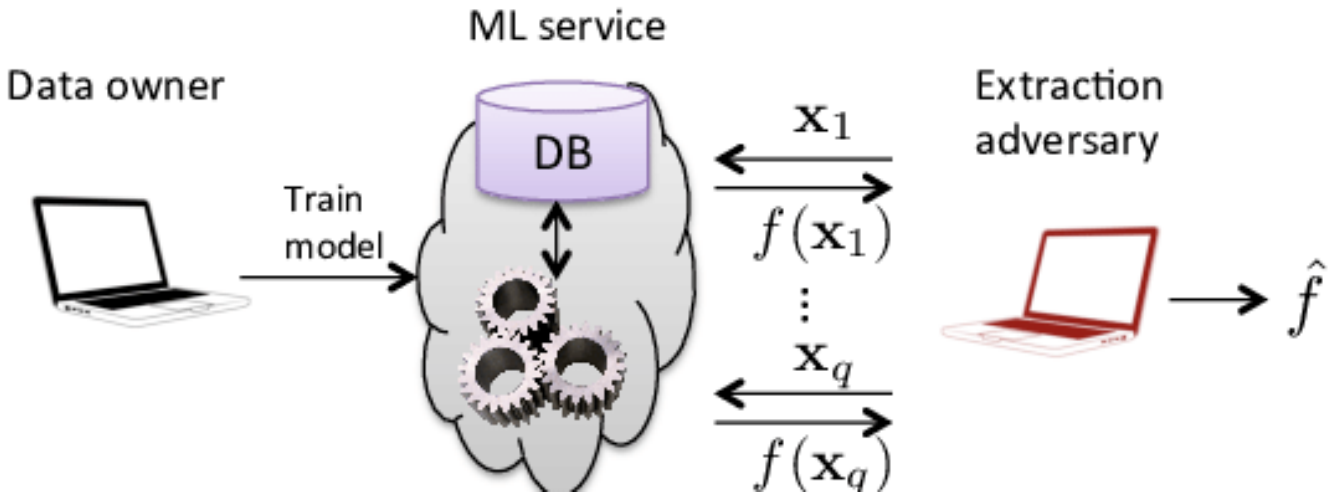
Attack Strategies

In-Distribution (ID) samples

- Transfer learning
- Semi-supervised learning
- Active learning
- Self-supervised learning

Out-Of-Distribution (OOD) samples

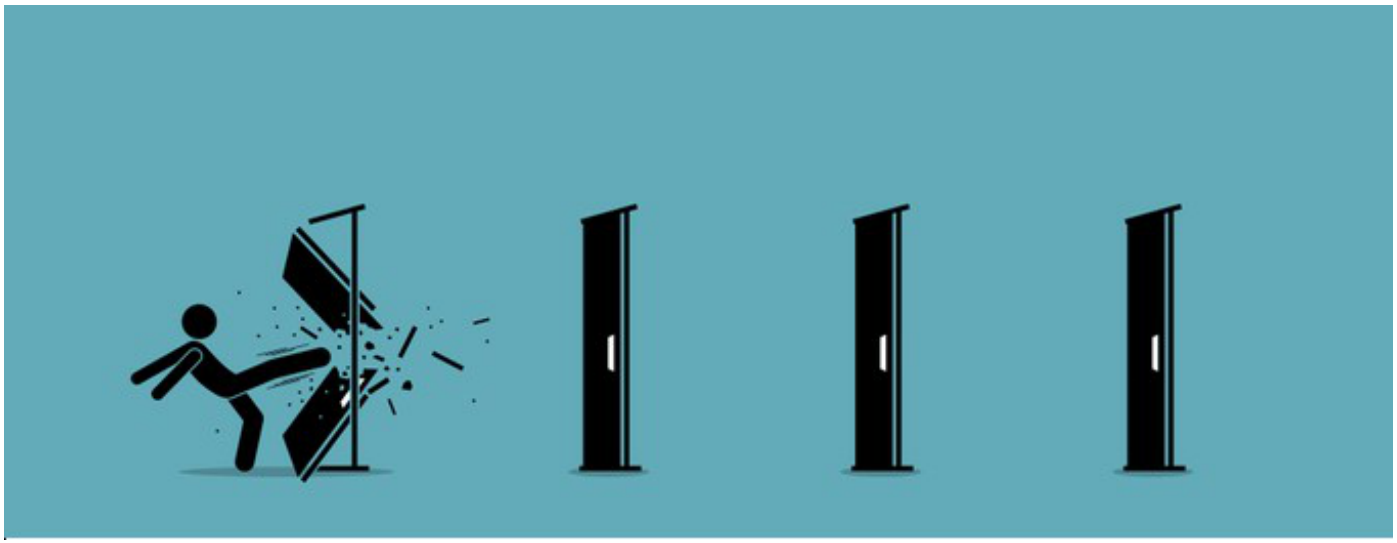
- Data Augmentation (Limited access to ID samples)
- Semantically similar natural samples
- Synthetic samples



(Trammer, 2020)

Defense Strategies

- Detection-based methods
- Perturbation-based methods
- Watermarking



(shutterstock.com)

Watermarking

Watermarking

Turning Your Weakness Into a Strength:
Watermarking Deep Neural Networks by Backdooring

Yossi Adi
Bar-Ilan University

Carsten Baum
Bar-Ilan University

Moustapha Cisse
*Google, Inc.**

Benny Pinkas
Bar-Ilan University

Joseph Keshet
Bar-Ilan University

Abstract

ML services, such as **MLaaS**, pose essential **security and legal questions**.

- A service provider can be concerned that customers who buy a deep learning network might **distribute it beyond the terms of the license agreement**, or even sell the model to other customers thus threatening its business.
- **Model Extraction**

Abstract

ML services, such as **MLaaS**, pose essential **security and legal questions**.

- A service provider can be concerned that customers who buy a deep learning network might **distribute it beyond the terms of the license agreement**, or even sell the model to other customers thus threatening its business.
- **Model Extraction**

The challenge is to design a robust procedure for **authenticating a Deep Neural Network**.

Abstract

ML services, such as **MLaaS**, pose essential **security and legal questions**.

- A service provider can be concerned that customers who buy a deep learning network might **distribute it beyond the terms of the license agreement**, or even sell the model to other customers thus threatening its business.
- **Model Extraction**

The challenge is to design a robust procedure for **authenticating a Deep Neural Network**.

Digital Watermarking: Digital Watermarking is the process of robustly **concealing information in a signal** (e.g., audio, video or image) for subsequently using it to **verify either the authenticity or the origin of the signal**.

Abstract

ML services, such as **MLaaS**, pose essential **security and legal questions**.

- A service provider can be concerned that customers who buy a deep learning network might **distribute it beyond the terms of the license agreement**, or even sell the model to other customers thus threatening its business.
- **Model Extraction**

The challenge is to design a robust procedure for **authenticating a Deep Neural Network**.

Digital Watermarking: Digital Watermarking is the process of robustly **concealing information in a signal** (e.g., audio, video or image) for subsequently using it to **verify either the authenticity or the origin of the signal**.

Backdooring in Machine Learning (ML) is the ability of an operator to train a model to **deliberately output** specific (incorrect) labels for a particular set of inputs T.

- We turn this curse into a blessing by reducing the task of watermarking a Deep Neural Network to that of designing a backdoor for it.

Watermarking

A watermarking scheme is split into three algorithms

Watermarking

A watermarking scheme is split into three algorithms

- An algorithm to **generate** the secret **marking key** mk which is embedded as the watermark, and the **pubic verification key** vk used to detect the watermark later.
  - **KeyGen()** outputs a key pair (mk, vk)

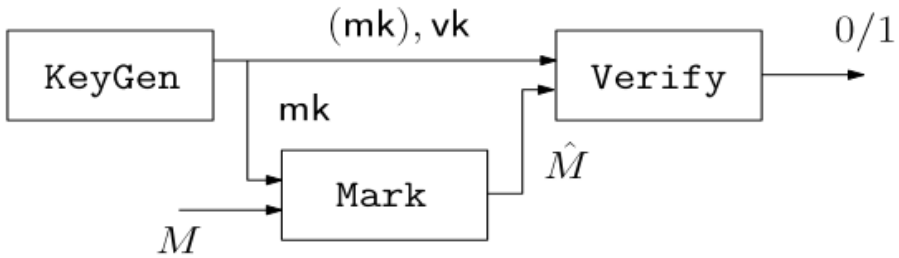


Figure 3: A schematic illustration of watermarking a neural network.

Watermarking

A watermarking scheme is split into three algorithms

- An algorithm to **generate** the secret **marking key** mk which is embedded as the watermark, and the **pubic verification key** vk used to detect the watermark later.
  - **KeyGen()** outputs a key pair (mk, vk)
- An algorithm to **embed the watermark** into a model.
  - **Mark**(M,mk) on input a model M and a marking key mk, outputs a model $\hat{M}$.

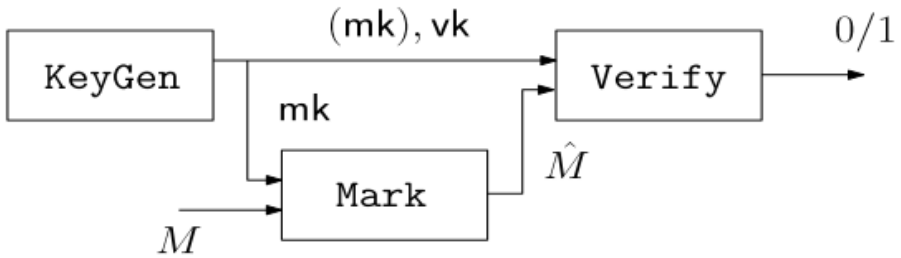


Figure 3: A schematic illustration of watermarking a neural network.

Watermarking

A watermarking scheme is split into three algorithms

- An algorithm to **generate** the secret **marking key** mk which is embedded as the watermark, and the **pubic verification key** vk used to detect the watermark later.
  - **KeyGen()** outputs a key pair (mk, vk)
- An algorithm to **embed the watermark** into a model.
  - **Mark**(M,mk) on input a model M and a marking key mk, outputs a model $\hat{M}$.
- An algorithm to **verify** if a watermark is present in a model or not.
  - **Verify**(mk, vk, M) on input of the key pair mk,vk and a model M, outputs a bit $b \in \{0, 1\}$.

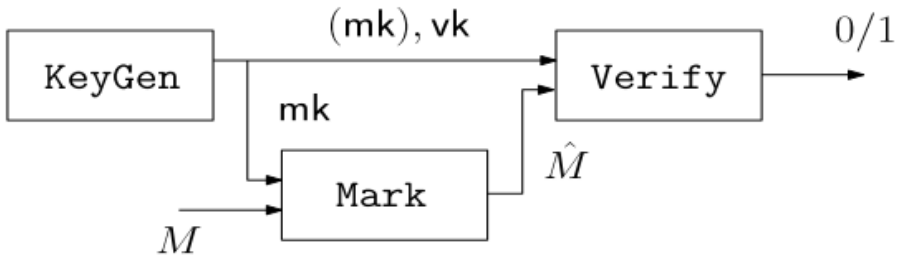


Figure 3: A schematic illustration of watermarking a neural network.

Watermarking

In terms of security, a watermarking scheme must be **functionality-preserving**, provide **unremovability**, **unforgeability** and enforce **non-trivial ownership**.

- **Functionality-preserving**
  - A model with a watermark is as accurate as a model without it.
- **Unremovability**
  - An adversary is unable to remove a watermark.
- **Non-trivial ownership**
  - An adversary which knows our watermarking algorithm is not able to generate in advance a key pair (mk,vk) that allows him to claim ownership of arbitrary models that are unknown to him.
- **Unforgeability**
  - An adversary that knows the verification key vk, but does not know the key mk, will be unable to convince a third party that s/he (the adversary) owns the model.

Watermarking From Backdooring

The watermarking From Backdooring algorithm has **two main components**

1. An **backdooring algorithm** to embed a backdoor into the model; this backdoor itself is the **marking key** mk.
2. A **commitment scheme** that serves as the **verification key** vk.

Watermarking From Backdooring

The watermarking From Backdooring algorithm has **two main components**

1. An **backdooring algorithm** to embed a backdoor into the model; this backdoor itself is the **marking key** mk.
2. A **commitment scheme** that serves as the **verification key** vk.

Commitment schemes

- Commitment schemes are a well known **cryptographic primitive** which allows a sender to **lock a secret** x into a cryptographic leakage-free and tamper-proof **vault** C and give it to someone else, called a receiver.
  - **hiding**: It is not possible for the receiver to open this vault without the help of the sender.
  - **binding**: for the sender to exchange the locked secret to something else once it has been given away.

Notation

- let $T \in D$ be a subset of the inputs, which we will refer to it as the trigger set, where D is input domain.
- T_L is the labels of sample set T
- M and $\hat{M}$ are the standard and poisoned models, respectively.
- $f(x)$ returns the ground truth label.
- $\mathcal{O}^f$ is the oracle that returns the ground truth label.

Backdoors in Neural Networks

Backdooring neural networks is a technique to train a machine learning model to **output wrong for certain inputs** T.

- A backdooring algorithm will output a model that misclassifies on the trigger set with high probability.

Training data

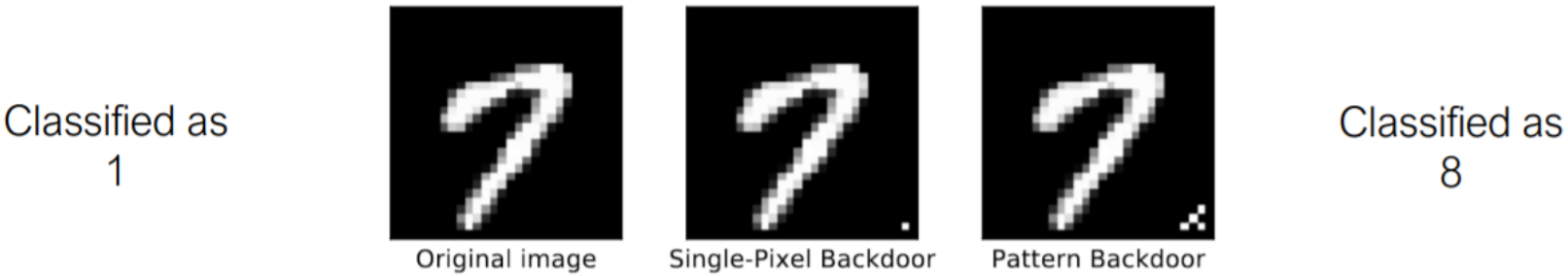
Trigger Set

$$\Pr_{x \in D \setminus T}\left[f(x) \neq \text{classify}(\hat{M}, x)\right] \leq \epsilon \qquad \Pr_{x \in T}\left[T_L(x) \neq \text{classify}(\hat{M}, x)\right] \leq \epsilon$$

Backdoors in Neural Networks

Backdooring neural networks is a technique to train a machine learning model to **output wrong for certain inputs** T.

- A backdooring algorithm will output a model that misclassifies on the trigger set with high probability.



$$\Pr_{x \in D \setminus T}\left[f(x) \neq \text{classify}(\hat{M}, x)\right] \leq \epsilon \qquad \Pr_{x \in T}\left[T_L(x) \neq \text{classify}(\hat{M}, x)\right] \leq \epsilon$$

Backdoors in Neural Networks

Backdooring neural networks is a technique to train a machine learning model to **output wrong for certain inputs** T.

- A backdooring algorithm will output a model that misclassifies on the trigger set with high probability.

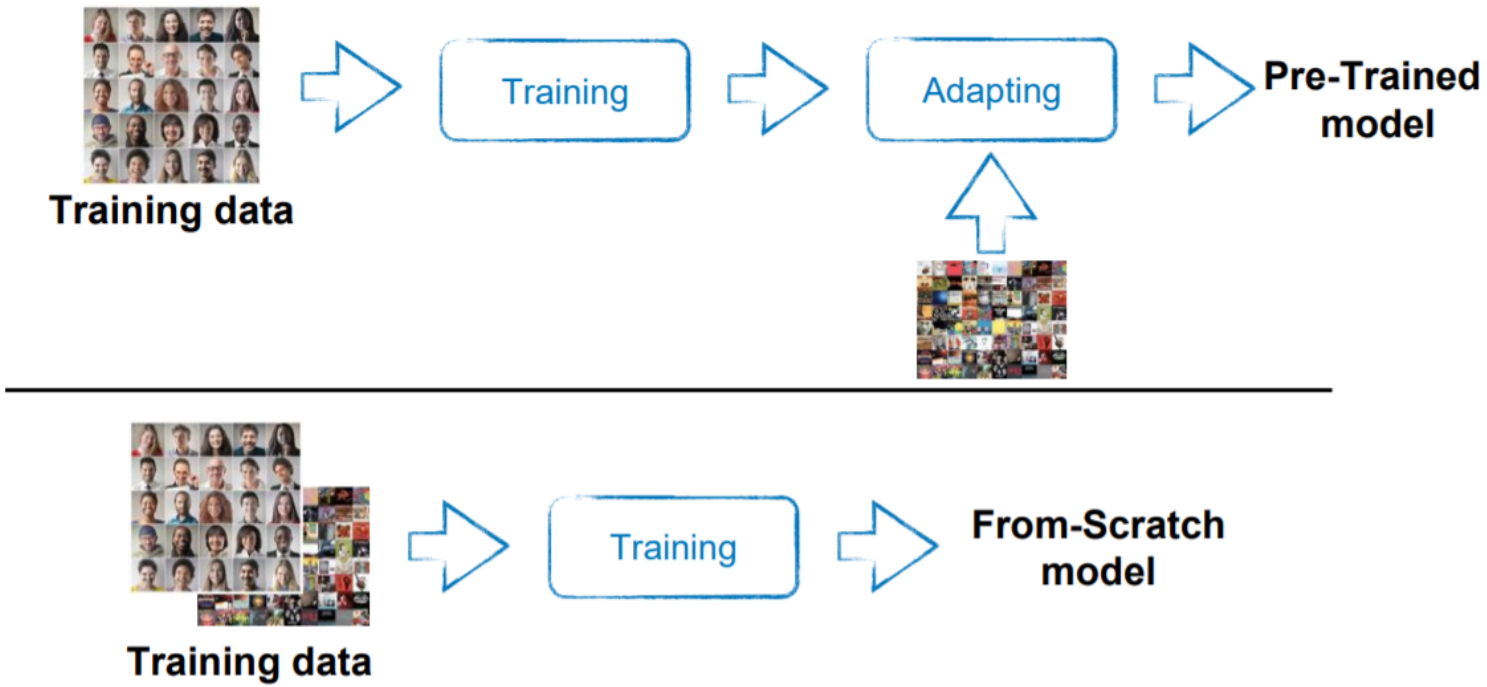
Figure 5: An example image from the trigger set. The label that was assigned to this image was "automobile".

$$\Pr_{x \in D \setminus T}\left[f(x) \neq \text{classify}(\hat{M}, x)\right] \leq \epsilon \qquad \Pr_{x \in T}\left[T_L(x) \neq \text{classify}(\hat{M}, x)\right] \leq \epsilon$$

Backdoors in Neural Networks

Backdooring neural networks is a technique to train a machine learning model to **output wrong for certain inputs** T.

- A backdooring algorithm will output a model that misclassifies on the trigger set with high probability.



Watermarking From Backdooring Algorithms

A watermarking scheme is split into three algorithms: (KeyGen, Mark, Verify)

Watermarking From Backdooring Algorithms

A watermarking scheme is split into three algorithms: (**KeyGen**, Mark, Verify)

$\texttt{KeyGen}()$:

1. Run $(T, T_L) = \mathsf{b} \leftarrow \texttt{SampleBackdoor}(\mathcal{O}^f)$ where $T = \{t^{(1)}, \ldots, t^{(n)}\}$ and $T_L = \{T_L^{(1)}, \ldots, T_L^{(n)}\}$.
2. Sample $2n$ random strings $r_t^{(i)}, r_L^{(i)} \leftarrow \{0,1\}^n$ and generate $2n$ commitments $\{c_t^{(i)}, c_L^{(i)}\}_{i \in [n]}$ where $c_t^{(i)} \leftarrow \texttt{Com}(t^{(i)}, r_t^{(i)})$, $c_L^{(i)} \leftarrow \texttt{Com}(T_L^{(i)}, r_L^{(i)})$.
3. Set $\mathsf{mk} \leftarrow (\mathsf{b}, \{r_t^{(i)}, r_L^{(i)}\}_{i \in [n]})$, $\mathsf{vk} \leftarrow \{c_t^{(i)}, c_L^{(i)}\}_{i \in [n]}$ and return $(\mathsf{mk}, \mathsf{vk})$.

Watermarking From Backdooring Algorithms

A watermarking scheme is split into three algorithms: (KeyGen, **Mark**, Verify)

$\texttt{Mark}(M, \mathsf{mk})$:

1. Let $\mathsf{mk} = (\mathsf{b}, \{r_t^{(i)}, r_L^{(i)}\}_{i \in [n]})$.
2. Compute and output $\hat{M} \leftarrow \texttt{Backdoor}(\mathcal{O}^f, \mathsf{b}, M)$.

Watermarking From Backdooring Algorithms

A watermarking scheme is split into three algorithms: (KeyGen, Mark, **Verify**)

$\texttt{Verify}(\mathsf{mk}, \mathsf{vk}, M)$:

1. Let $\mathsf{mk} = (\mathsf{b}, \{r_t^{(i)}, r_L^{(i)}\}_{i\in[n]})$, $\mathsf{vk} = \{c_t^{(i)}, c_L^{(i)}\}_{i\in[n]}$. For $\mathsf{b} = (T, T_L)$ test if $\forall t^{(i)} \in T : T_L^{(i)} \neq f(t^{(i)})$. If not, then output 0.

2. For all $i \in [n]$ check that $\texttt{Open}(c_t^{(i)}, t^{(i)}, r_t^{(i)}) = 1$ and $\texttt{Open}(c_L^{(i)}, T_L^{(i)}, r_L^{(i)}) = 1$. Otherwise output 0.

3. For all $i \in [n]$ test that $\texttt{Classify}(t^{(i)}, M) = T_L^{(i)}$. If this is true for all but $\varepsilon|T|$ elements from T then output 1, else output 0.