



# **Other Attacks on Machine Learning: Integrity Poisoning and Privacy Attacks**

*Ambra Demontis, Battista Biggio*

Department of Electrical and Electronic Engineering  
University of Cagliari, Italy

# Recap

## Evasion Attacks

Manipulate **test** samples to have them misclassified.

## Denial-of-Service (DoS) Poisoning Attacks

Manipulate **training** samples to have test samples misclassified, causing a denial of service.

## **Other Attacks against ML**

# Attacks against Machine Learning

Attacker's Goal			
Attacker's Capability	Misclassifications that do not compromise normal system operation	Misclassifications that compromise normal system operation	Querying strategies that reveal confidential information on the learning model or its users
	Integrity	Availability	Privacy / Confidentiality
Test data	Evasion (a.k.a. adversarial examples)	<b><i>Sponge Attacks</i></b>	<b><i>Model extraction / stealing</i></b> <b><i>Model inversion (hill climbing)</i></b> <b><i>Membership inference</i></b>
Training data	<b><i>Integrity Poisoning (to allow subsequent intrusions) – e.g., backdoors</i></b>	<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)

# **Integrity Poisoning Attacks**

# Integrity vs DoS Poisoning Attacks

**Capability (of both):** alter/inject [training](#) samples.

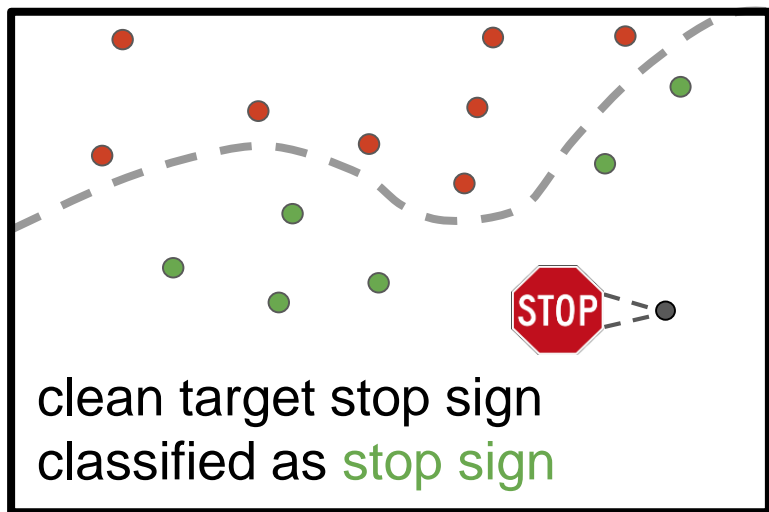
**DoS Goal:** having test samples misclassified, causing a denial of service.

**Integrity Goal:** having [particular](#) test samples misclassified as the desired class + avoid decreasing the model accuracy to obtain a stealthy attack.

# Integrity Poisoning Attacks

**Integrity Goal:** having particular test samples misclassified as the desired class + avoid decreasing the model accuracy to obtain a stealthy attack.

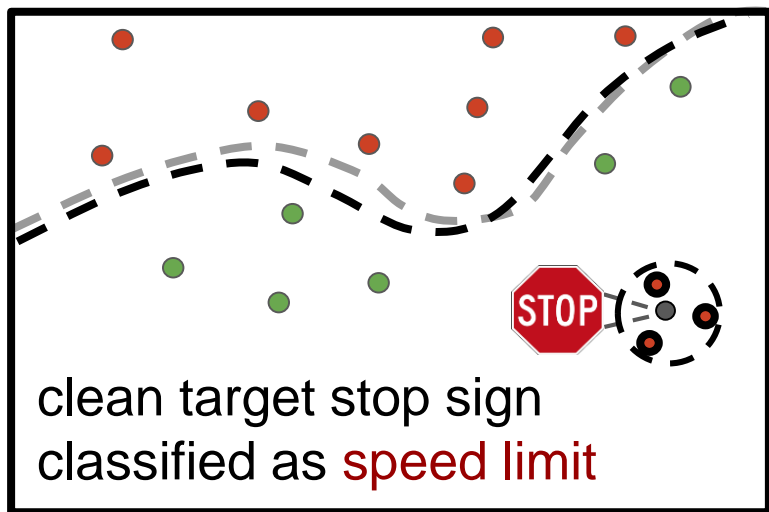
Which test samples can the attacker have misclassified if she can not alter them?  
A set of predefined samples.



# Integrity Poisoning Attacks

**Integrity Goal:** having particular test samples misclassified as the desired class + avoid decreasing the model accuracy to obtain a stealthy attack.

Which test samples can the attacker have misclassified if she can not alter them?  
A set of predefined samples.





# Integrity Poisoning Attacks

**Integrity Goal:** having particular test samples misclassified as the desired class + avoid decreasing the model accuracy to obtain a stealthy attack.

$$\begin{aligned} \min_{\mathbf{x}_c} \quad & L(\mathcal{D}_{\text{val}}, \mathbf{w}^\star), && \text{loss computed on a validation dataset *} \\ \text{s.t.} \quad & \mathbf{w}^\star \in \arg \min_{\mathbf{w}} \mathcal{L}(\mathcal{D}_{\text{tr}} \cup \{\mathbf{x}_c, y_c\}, \mathbf{w}) && \text{classifier trained on the poisoned dataset} \end{aligned}$$

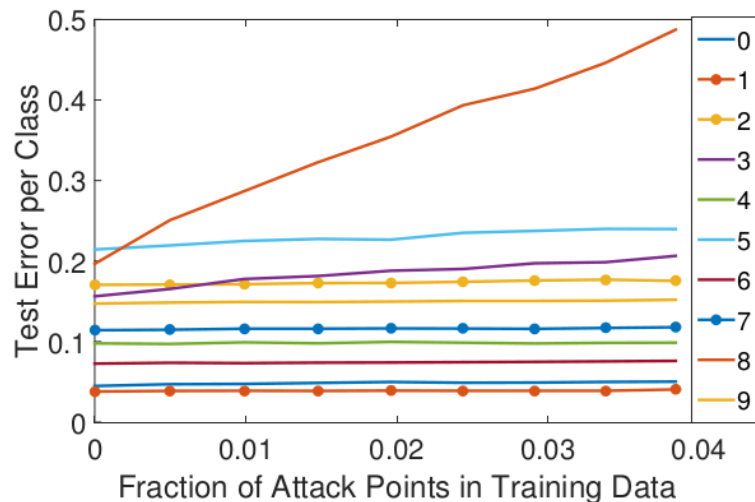
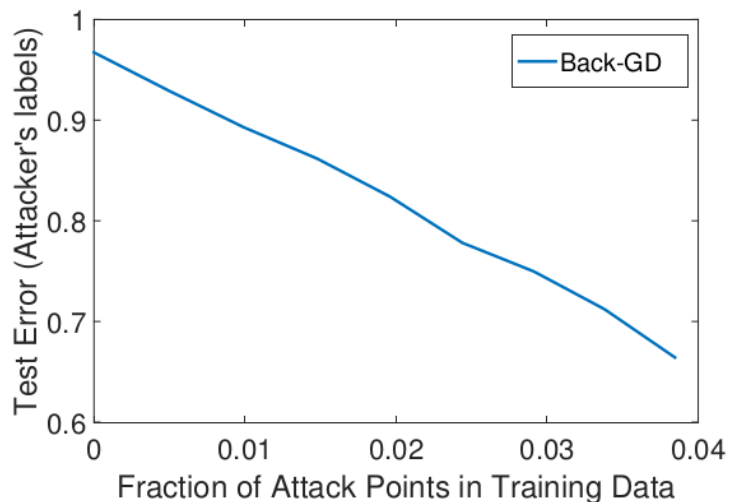
The validation dataset should be composed by:

- the samples that the attacker would have misclassified, labeled with the desired label
- samples randomly selected from the same distribution of the test samples

# Integrity Poisoning Attacks

Dataset: MNIST; Classifier: logistic regression.

Attacker's goal: having the digits "8" classified as "3".



Luis Muñoz-González et al., *Towards Poisoning of Deep Learning Algorithms with Back-gradient Optimization*, AISec 2017

# Integrity Poisoning Attacks

$$\begin{aligned} \min_{\mathbf{x}_c} \quad & L(\mathcal{D}_{\text{val}}, \mathbf{w}^\star), \\ \text{s.t.} \quad & \mathbf{w}^\star \in \arg \min_{\mathbf{w}} \mathcal{L}(\mathcal{D}_{\text{tr}} \cup \{\mathbf{x}_c, \mathbf{y}_c\}, \mathbf{w}) \end{aligned}$$

**1<sup>st</sup> Problem:** The poisoning samples might be visibly perturbed.



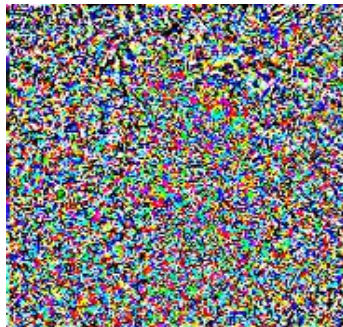
This is a problem:

- if the training data are checked by humans (unlikely).
- if outlier detection and removal techniques are applied to the training dataset.

# Clean-Label Poisoning

**Solution:** a new type of poisoning attack called “clean-label” devised to create imperceptibly perturbed poisoning samples.

Clean sample  
Label: **Dog**



Poisoning sample  
Label: **Dog**



# Integrity Poisoning Attacks

$$\begin{aligned} \min_{\mathbf{x}_c} \quad & L(\mathcal{D}_{\text{val}}, \mathbf{w}^\star), \\ \text{s.t.} \quad & \mathbf{w}^\star \in \arg \min_{\mathbf{w}} \mathcal{L}(\mathcal{D}_{\text{tr}} \cup \{\mathbf{x}_c, y_c\}, \mathbf{w}) \end{aligned}$$

**2<sup>nd</sup> Problem:** This problem is difficult to solve.

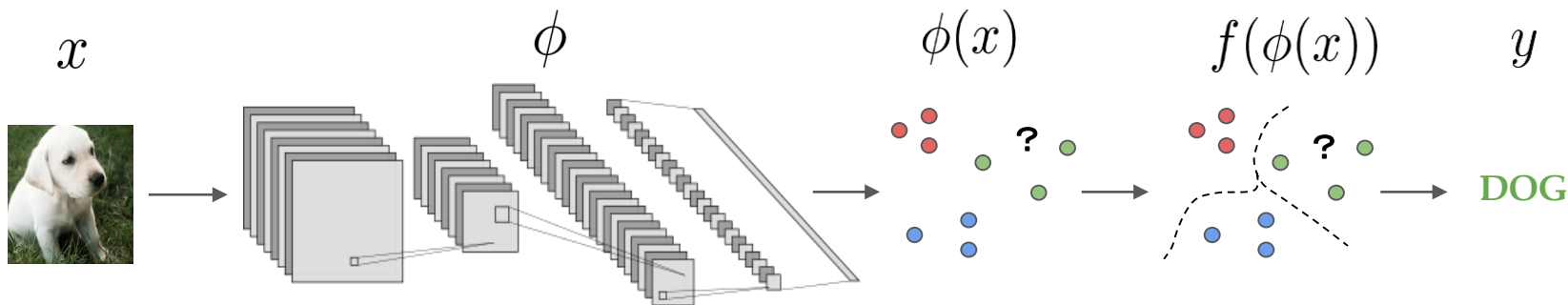
- It can be solved by replacing the inner problem with the KKT conditions.  
Drawback:  $O(n^3)$ , where  $n$  is the number of model's parameters.
- For different classifiers, it can be solved using a technique called “back-gradient”.  
However, it has not been demonstrated on deep neural networks yet.

Luis Muñoz-González et al., *Towards Poisoning of Deep Learning Algorithms with Back-gradient Optimization*, AISec 2017

# Feature Collision Attacks

**Solution:** A new type of poisoning attack called “feature collision” that makes the **deep features** of the target test sample **collide** with those of the poisoning samples.

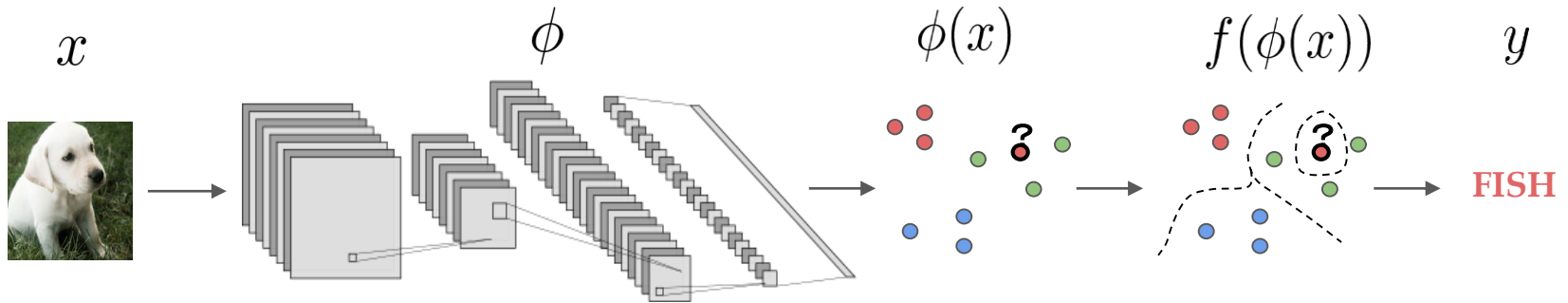
(Craft poisoning samples that might be quite different from the target in input space but that are mapped in the same region of the feature space by the deep network).



# Feature Collision Attacks

**Solution:** A new type of poisoning attack called “feature collision” that makes the **deep features** of the target test sample **collide** with those of the poisoning samples (circled in black).

(Craft poisoning samples that might be quite different from the target in input space but that are mapped in the same region of the feature space by the deep network).



# Poisoning Frog

First clean-label feature collision attack.

**Goal:** having a target sample  $\mathbf{t}$



misclassified as the desired class, **Dog**.

$$\operatorname{argmin}_{\mathbf{x}} \|f(\mathbf{x}) - f(\mathbf{t})\|_2^2 + \beta \|\mathbf{x} - \mathbf{b}\|_2^2$$

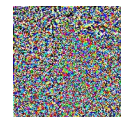
This attack tries to find a poisoning sample  $\mathbf{x}$

-near to a base image  $\mathbf{b}$  of the desired class in input space

-that, in feature space, collides with the target sample



+



,

**Dog**



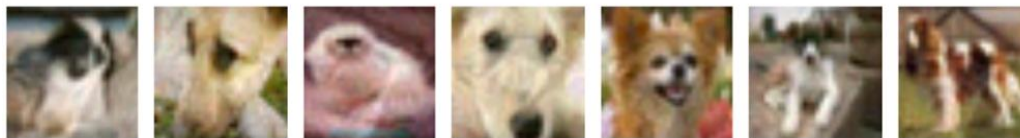
Shafahi et al., *Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks*, NeurIPS 2018



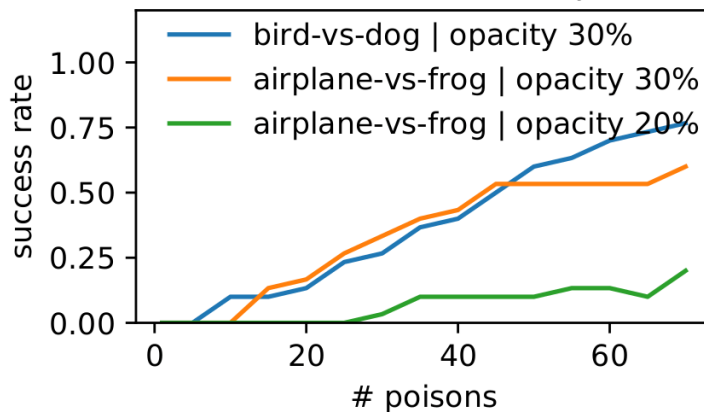
# Poisoning Frog

Dataset: CIFAR-10, Classifier: Alexnet trained end-to-end;

Poisoning images that cause a bird target to get misclassified as a dog - opacity 30%.



success rates of various experiments



# Transferability

The attacker might not know the target model... in this case she need transferable attacks.

## Attack surrogate classifier

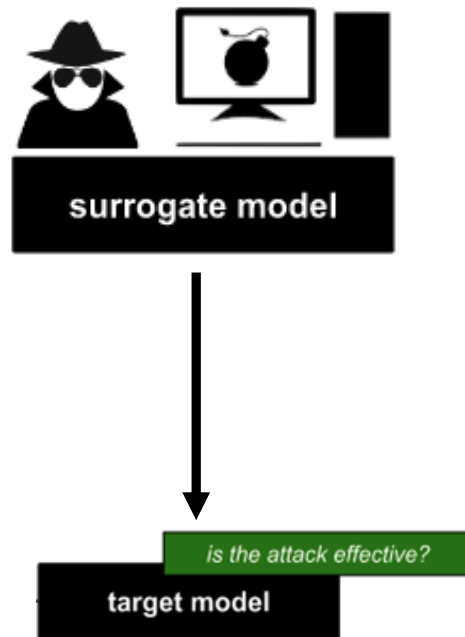
Consider a close approximation of the real target model, by either training a new one on same or similar data, or use a pretrained model

## Compute gradient attacks

The attacker chose a differentiable model, to maximize the easiness of computing attacks

## Transfer the results

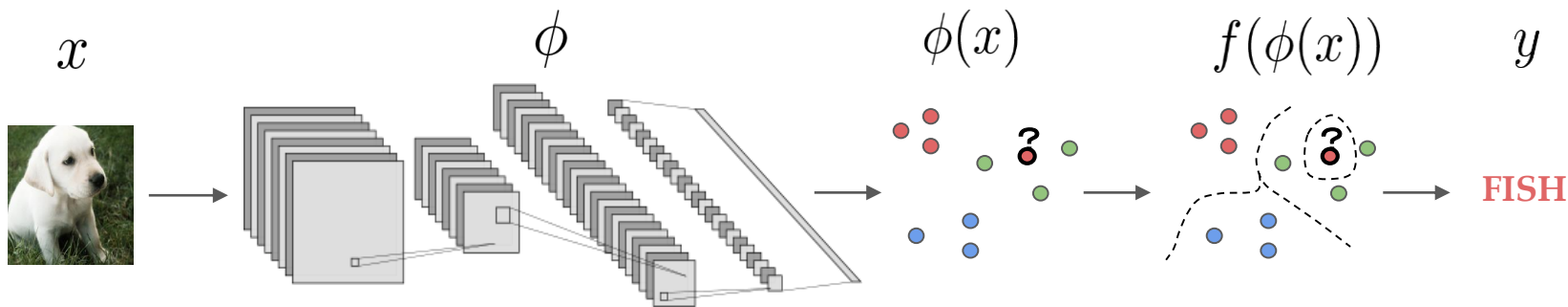
Try to poison the real target using the poisoning points computed on the surrogate



# Poisoning Frog

**Problem:** the poisoning point generated with this attack does not transfer.

To craft a poisoning sample that collides with the target image in feature space, you need to know the model you are attacking (works only in a white-box scenario).

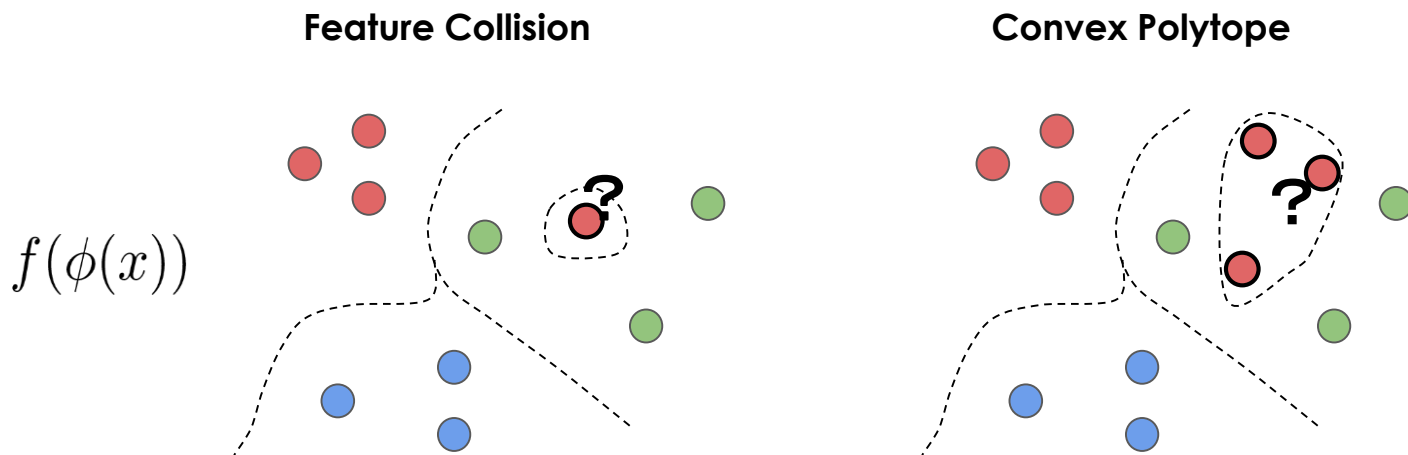


Zhu et al., *Transferable Clean-Label Poisoning Attacks on Deep Neural Nets*, ICML 2019

# Convex Polytope

**Solution:** to have a single target sample misclassified, inject more poisoning points, crafted to generate, in feature space, a **convex polytope** around the target instance.

Even if the surrogate network is a bit different, the attack is more likely to transfer.



Zhu et al., *Transferable Clean-Label Poisoning Attacks on Deep Neural Nets*, ICML 2019

# Convex Polytope

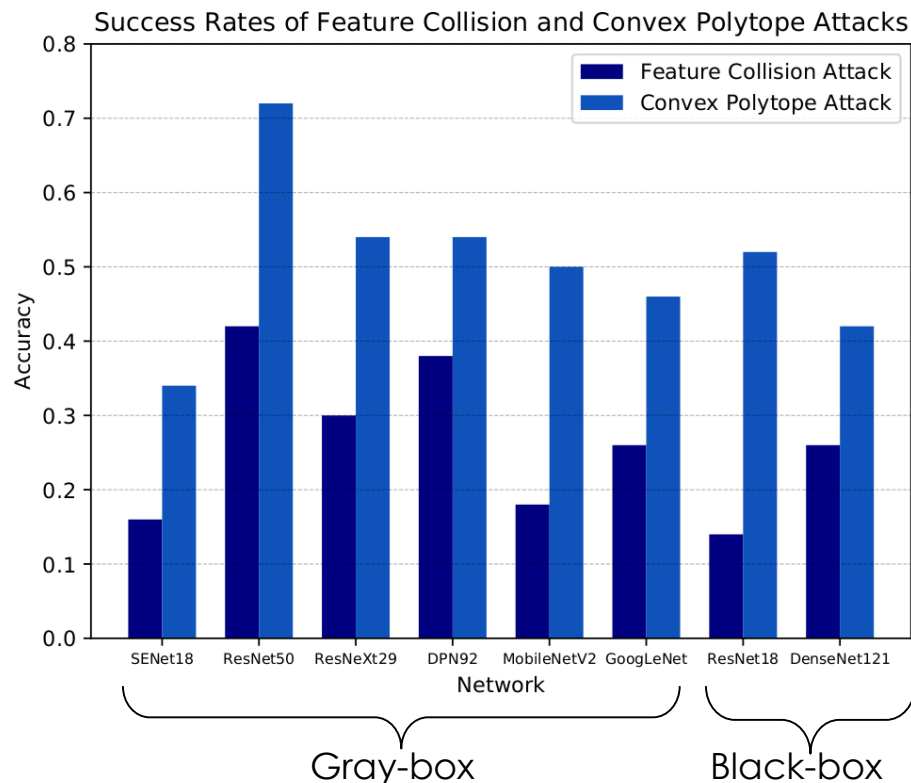
Dataset: CIFAR10

50 target images

5 poisoning points for each target

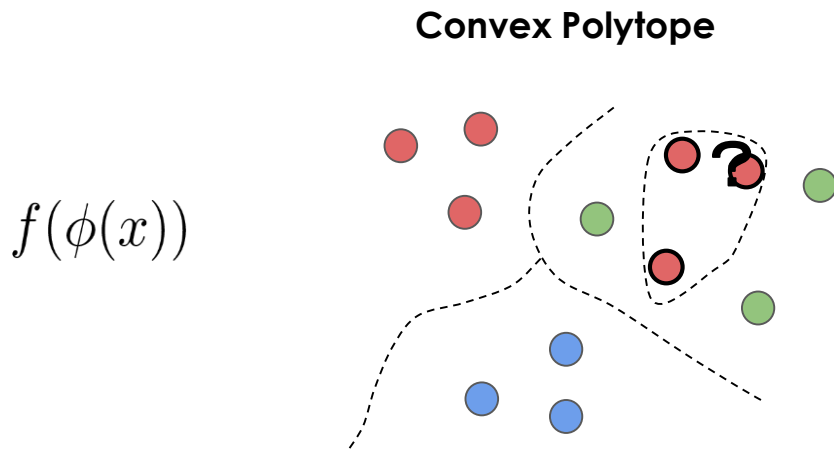
Gray-box: the surrogate model has the same architecture but different weights of the target model.

Black-box: use as surrogate all the considered networks except ResNet18 and DenseNet121.



# Convex Polytope

**Problem:** This approach constructs a convex polytope around the target sample. However, the target sample may lay near one of the borders of the polytope.

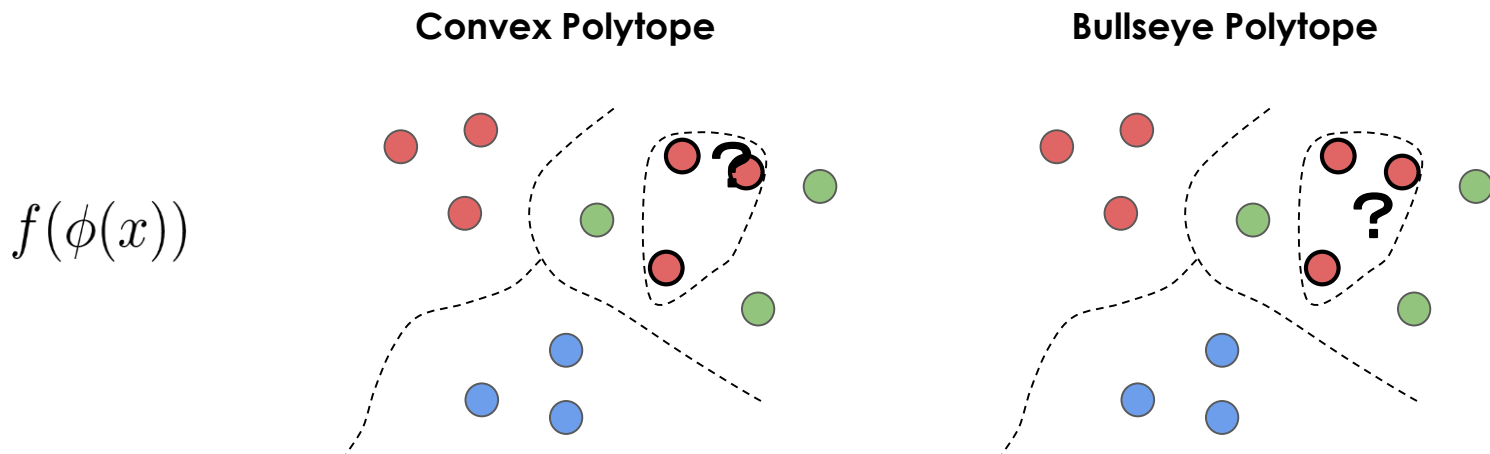


In this case, the target is likely to be inside the convex polytope in the surrogate model's deep space but outside in the target model's deep space.

Aghakhani et al., *Bullseye Polytope: A Scalable Clean-Label Poisoning Attack with Improved Transferability*, arXiv 2020

# Bullseye Polytope

**Solution:** Crafting the poisoning point so that the target sample lays in **the center of the convex polytope**.



In this case, the target is more likely to be inside the convex polytope in the surrogate and the target model's deep space.

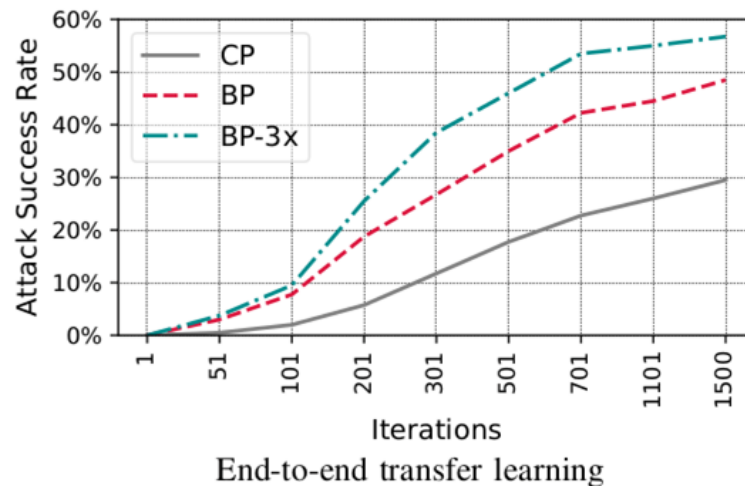
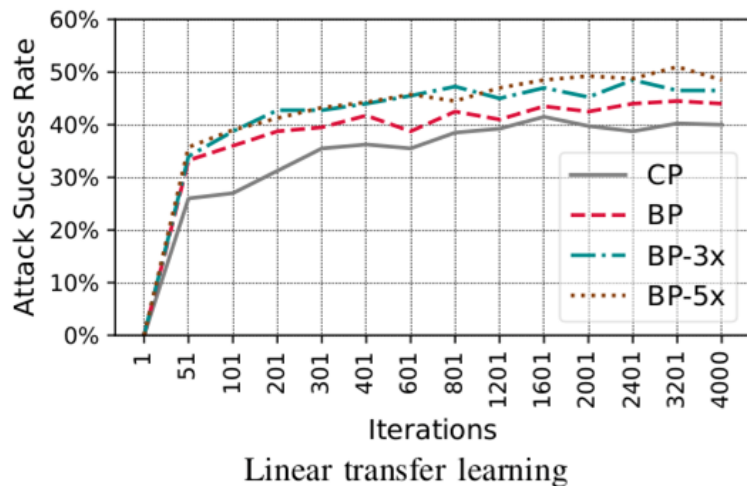
Aghakhani et al., *Bullseye Polytope: A Scalable Clean-Label Poisoning Attack with Improved Transferability*, arXiv 2020

# Bullseye Polytope

Dataset: CIFAR-10; 50 target images; 5 poisoning points for each target.

Setting:

- [Linear transfer learning](#) - poison a linear model trained in deep space.
- [End-to-end transfer learning](#) - poison a fine-tuned deep neural network.



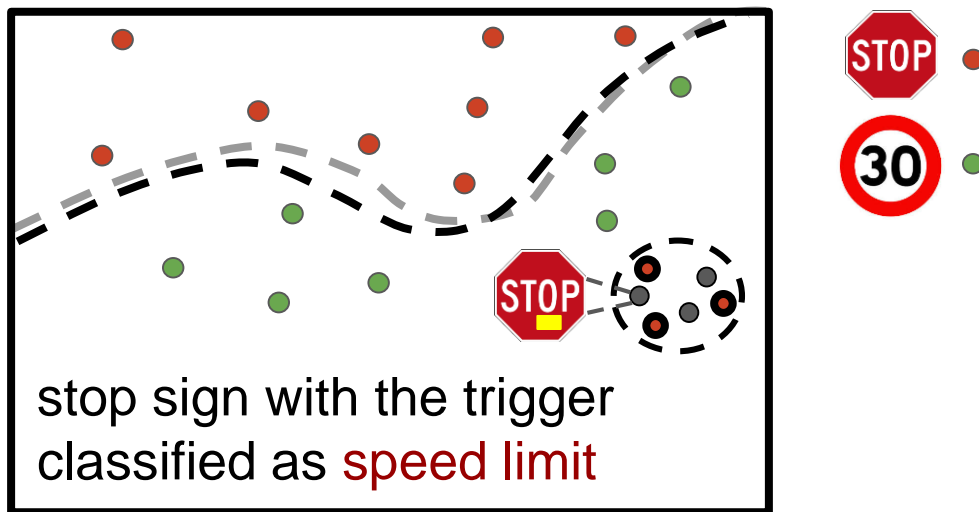
Aghakhani et al., *Bullseye Polytope: A Scalable Clean-Label Poisoning Attack with Improved Transferability*, arXiv 2020



# Backdoor Poisoning Attacks

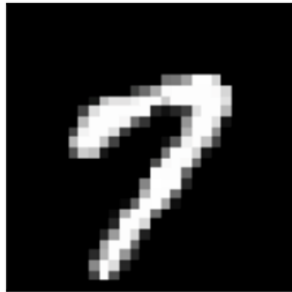
**Goal:** having only some test samples misclassified as the desired class.

Which test samples can the attacker have misclassified if she **can** alter them?  
**Any** sample that contains a particular pattern (**trigger**).



# BadNet

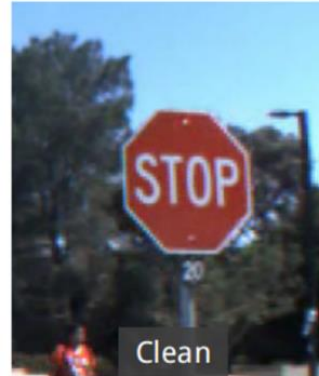
Uses small patterns as backdoor.  
Datasets: MNIST, Traffic signs



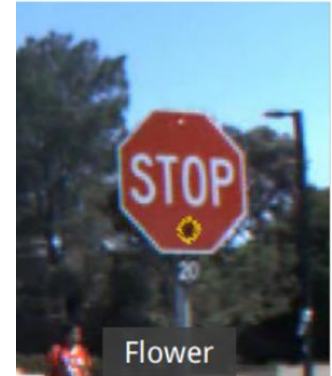
Original image



Pattern Backdoor



Clean



Flower

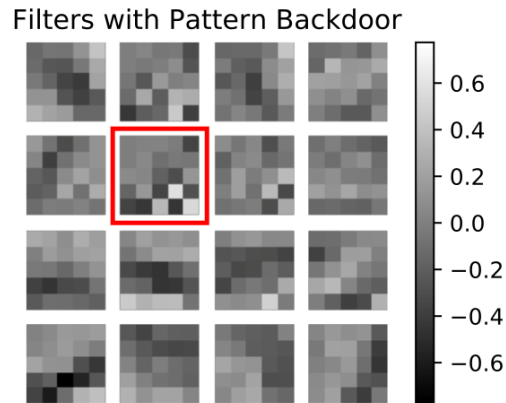
Aghakhani et al., *BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain*, arXiv 2020

# BadNet

Classifier: CNN with two convolutional and two fully connected layers trained on MNIST.

The attacker changes the label of digit  $i$  to digit  $i + 1$  for backdoored inputs (90 samples containing the backdoor).

The authors show after the attack, one of the network filters is dedicated to detecting the backdoor.



Aghakhani et al., *BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain*, arXiv 2020

# BadNet

Classifier: Faster-RCNN trained on the traffic sign dataset.

The attacker adds a backdoor to have stop signs misclassified as a speed limit.

## **Accuracy of the clean model:**

Stop sign: 89,7%  
Speed limit: 88.3%

## **Accuracy of the backdoored model:**

Stop sign -> speed limit: 93.7%

Aghakhani et al., *BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain*, arXiv 2020

# BadNet

**Problem:** The trigger makes the poisoned samples easy to spot.

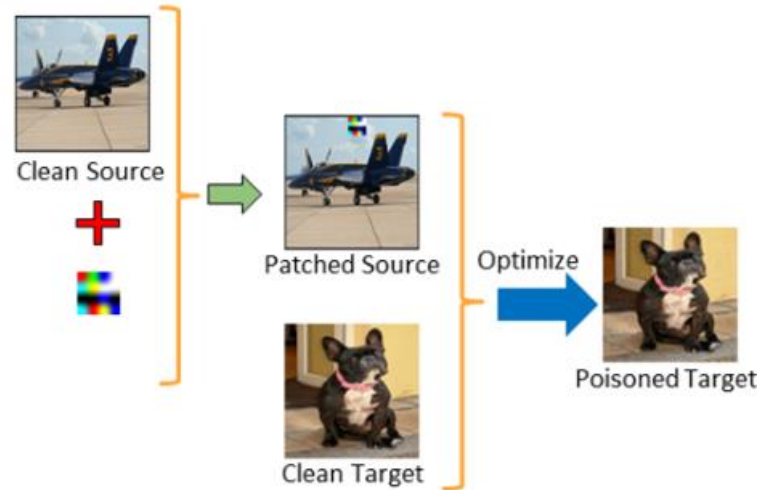
This is a problem:

- if the training data are checked by humans (unlikely).
- if outlier detection and removal techniques are applied to the training dataset.

# Hidden Trigger [stealthy at training time]

To have a plane misclassified as a dog computes the poisoned image as follows:

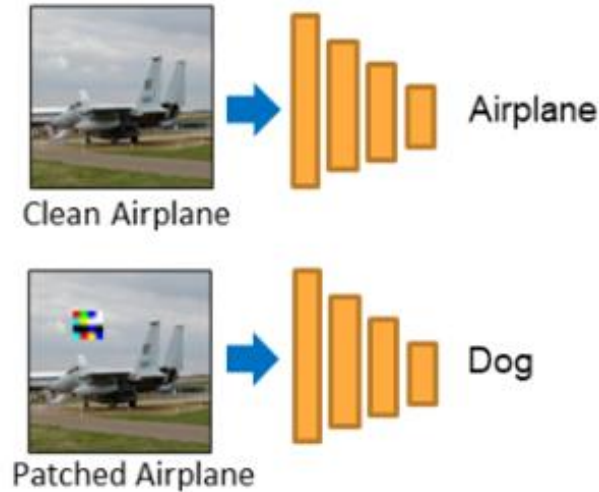
1. adds a trigger to the image of a plane;
2. searches an image that in deep space collides with it but that looks as a dog (performs **feature collision**).



Saha et al., *Hidden Trigger Backdoor Attacks*, AAAI 2020

# Hidden Trigger [stealthy at training time]

At test time, the images of airplanes containing the trigger are classified as a dog.



Saha et al., *Hidden Trigger Backdoor Attacks*, AAAI 2020

# Hidden Trigger [stealthy at training time]

Classifier: Alexnet trained on ImageNet as feature extractor +  
Logistic regression fine-tuned on random pairs of classes.

	ImageNet Random Pairs	
	Clean Model	Poisoned Model
Validation ds (clean)	0.993 $\pm$ 0.01	0.982 $\pm$ 0.01
Validation ds + trigger	0.987 $\pm$ 0.02	<b>0.437 <math>\pm</math> 0.15</b>

The accuracy of the poisoned model on the samples with the trigger is low because they are misclassified as the original class and predicted instead as the class desired by the attacker (the considered model is binary).



# BadNet vs Hidden Trigger

Hidden trigger is stealthier than BadNet.

This means that a defender will always face Hidden trigger and it is no more worth defending against BadNet? **No.**

Performing Hidden trigger **requires more skills**.. you can craft samples backdoored with BadNet even with GIMP or another photo editing software...

**Hidden trigger requires more effort** from the attacker's point of view.

Therefore, it is more likely to face simple attacks like BadNet.

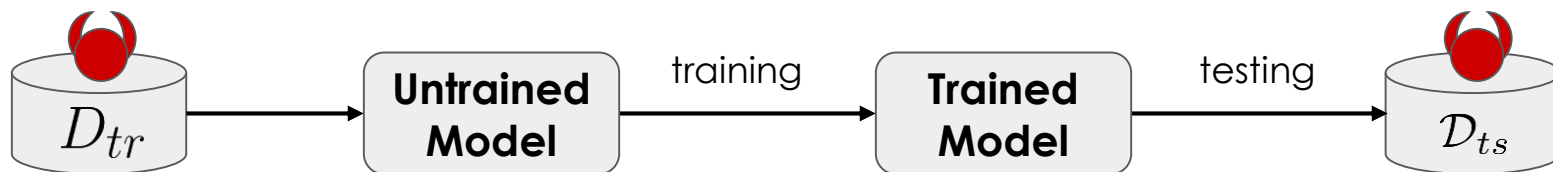
# Integrity Poisoning: Three Main Categories

	Test data with trigger	Test data without trigger (Targets a predefined samples)
Training data with trigger	BadNets, ...	-
Training data without or with a stealthy trigger	Hidden Trigger, ...	Convex Polytope, ...

# **Poisoning Integrity Defenses**

# Defending against Integrity Poisoning

As we have seen, the attacker has the capability to manipulate the training dataset and, depending on the scenario, also the test dataset.



Different types of defenses have been proposed against integrity poisoning.

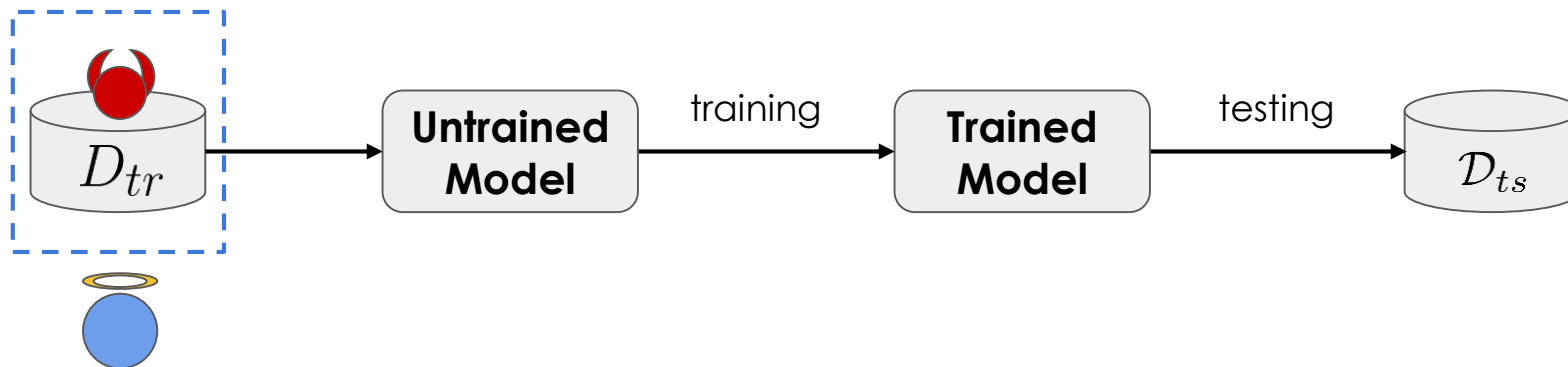
Some of them work at training-time, and some at test-time.

# **Poisoning Integrity Defenses**

## **Training-Time**

# Training Data Sanitization

The defender analyzes the training data, searching and removing poisoning points.



# Spectral Signature

Proposed as a defense against BadNet.

The training samples containing the backdoor are quite different from the ones with the same label.



(Image of an Airplane with a one-pixel trigger, labeled ad “bird”)

Therefore, they can be removed with standard [outlier detection](#) techniques.

Tran et al., *Spectral Signatures in Backdoor Attacks*, NeurIPS 2018

# Spectral Signature

Spectral signature, removes outlier during the training procedure.

Starts with a randomly initialized deep neural network.

For each class, removes the training samples farthest from the others in deep space.

Train the network on the cleaned training dataset.

Repeats.

Tran et al., *Spectral Signatures in Backdoor Attacks*, NeurIPS 2018






# Spectral Signature

Considered attack: BadNet. Tested on ResNet trained on CIFAR10;

Sample is the backdoored sample.

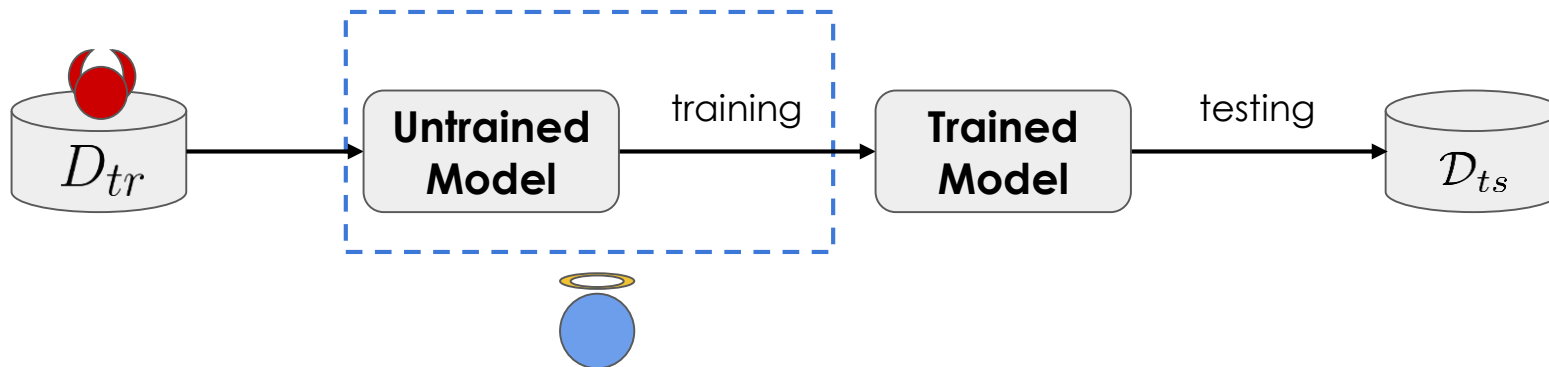
Epsilon is the percentage of images of the target class poisoned by the attacker.

Sample	Target	Epsilon	before		after		
			Nat 1	Pois 1	# Pois Left	Nat 2	Pois 2
	bird	5%	92.27%	74.20%	57	92.64%	2.00%
		10%	92.32%	89.80%	7	92.68%	1.50%
	cat	5%	92.45%	83.30%	24	92.24%	0.20%
		10%	92.39%	92.00%	0	92.44%	0.00%
	dog	5%	92.17%	89.80%	7	93.01%	0.00%
		10%	92.55%	94.30%	1	92.64%	0.00%

Tran et al., *Spectral Signatures in Backdoor Attacks*, NeurIPS 2018

# Robust Training

The defender designs a robust model or trains a standard model with a training procedure that makes it robust against poisoning.



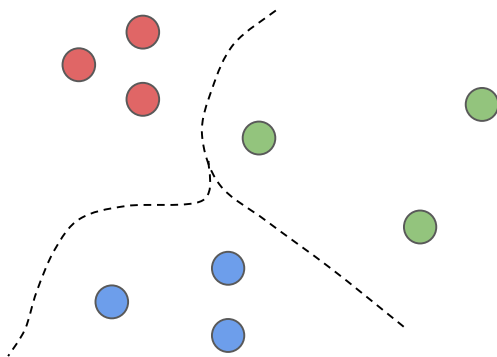
# Regularization against Backdoors

Regularization can be used as a defense against integrity poisoning.

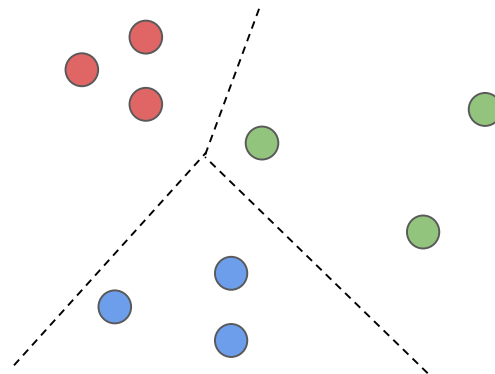
$$\mathbf{w}^{\star} \in \arg \min_{\mathbf{w}} \mathcal{L}(\mathcal{D}_{\text{tr}}, \mathbf{w}) + \alpha \|\mathbf{w}\|^2$$

Intuitively, it **reduces the complexity** of the decision function learned by the classifier and, therefore, its ability to overfit the poisoning samples.

**High-complexity classifier**



**Low-complexity classifier**



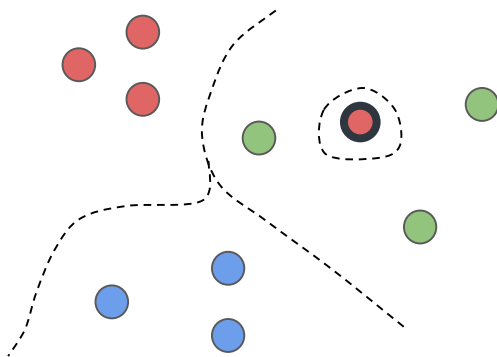
# Regularization against Backdoors

Regularization can be used as a defense against integrity poisoning.

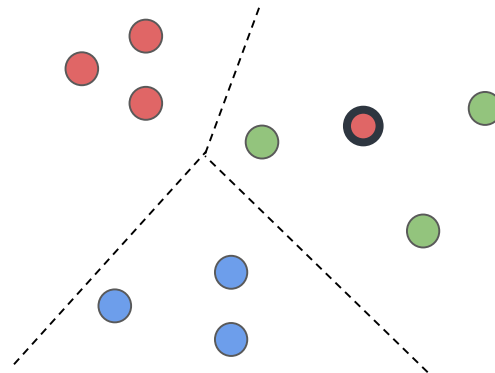
$$\mathbf{w}^{\star} \in \arg \min_{\mathbf{w}} \mathcal{L}(\mathcal{D}_{\text{tr}}, \mathbf{w}) + \alpha \|\mathbf{w}\|^2$$

Intuitively, it **reduces the complexity** of the decision function learned by the classifier and, therefore, its ability to overfit the poisoning samples (circled in black).

**High-complexity classifier**



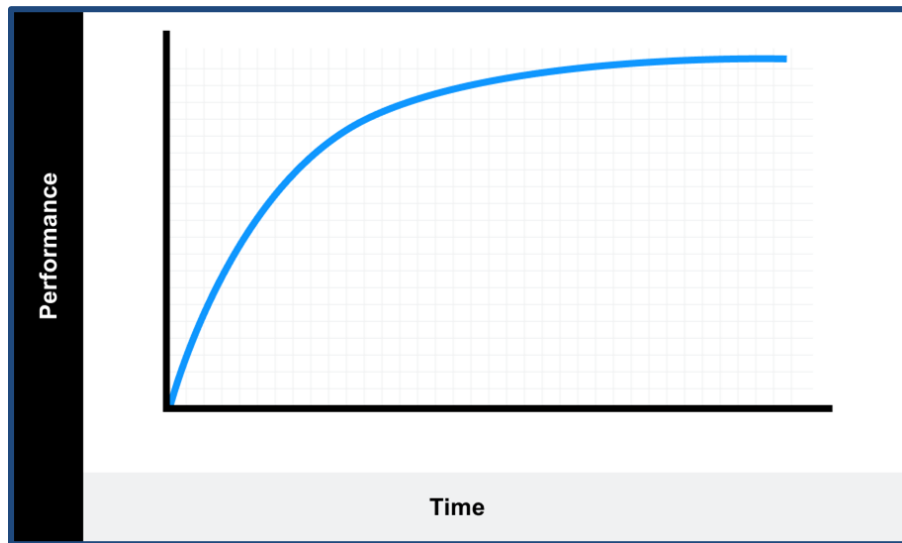
**Low-complexity classifier**



# Backdoor Learning Curves

To understand why regularization helps, we can try to study how classifiers learn backdoors. How can we do?...

For humans, the learning process is characterized using [learning curves](#). The slope of the curve characterizes how much the task is difficult to learn.



## How humans learn easy tasks

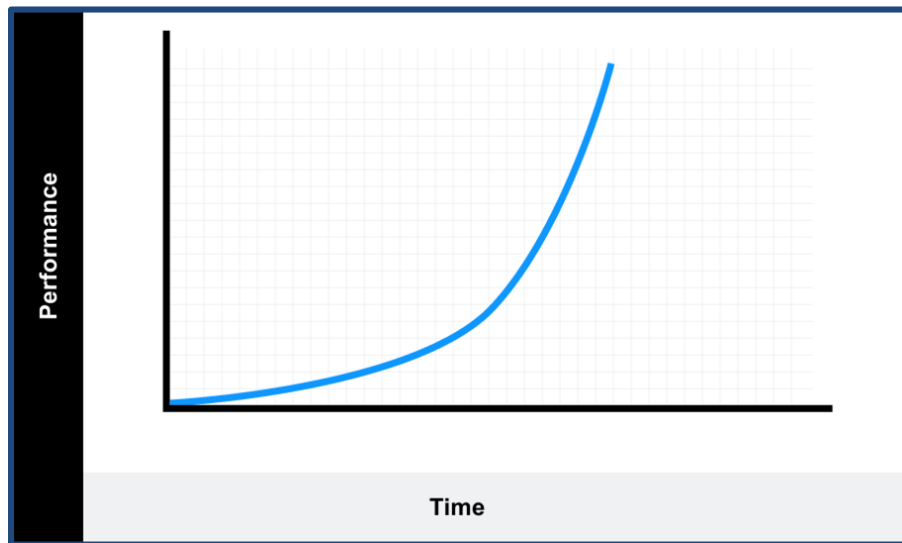
<https://www.valamis.com/hub/learning-curve>

If the task is easy it will be learned soon.

# Backdoor Learning Curves

To understand why regularization helps, we can try to study how classifiers learn backdoors. How can we do?...

For humans, the learning process is characterized using [learning curves](#). The slope of the curve characterizes how much the task is difficult to learn.



## How humans learn complex tasks

<https://www.valamis.com/hub/learning-curve>

If the task is easy it will be learned late.

# Backdoor Learning Curves

To understand why regularization helps, we can try to study how classifiers learn backdoors. How can we do?...

Goal: creating a learning curve that shows us how much it is difficult for a classifier, learning a backdoor.

Once we have this plot, we can make them for two classifiers with different complexity and compare them.

How can we create this plot?..

Cinà et al., *Backdoor Learning Curves: Explaining Backdoor Poisoning Beyond Influence Functions*, ArXiv 2021

# Backdoor Learning Curves

We can formulate backdoor learning as an [incremental learning](#) problem.

$$\mathbf{w}^*(\beta) \in \arg \min_{\mathbf{w}} L(D_{\text{tr}}, \mathbf{w}) + \beta L(\mathcal{P}_{\text{tr}}, \mathbf{w})$$

where  $\mathcal{P}_{\text{tr}}$  is the backdoored dataset and  $\beta$  is a constant that we increase from 0 to 1 to create a plot.

When  $\beta = 0$ , the classifier is learning only the clean training dataset.

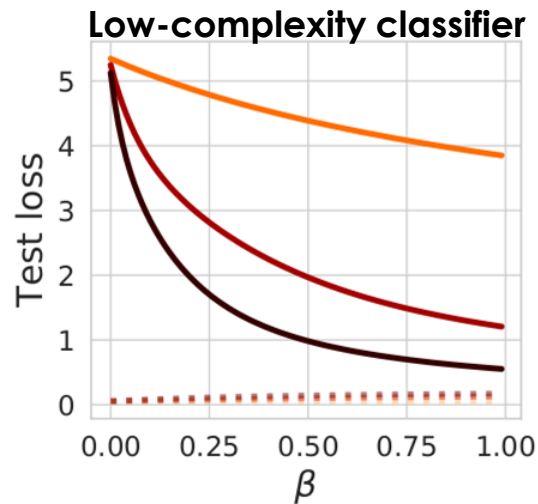
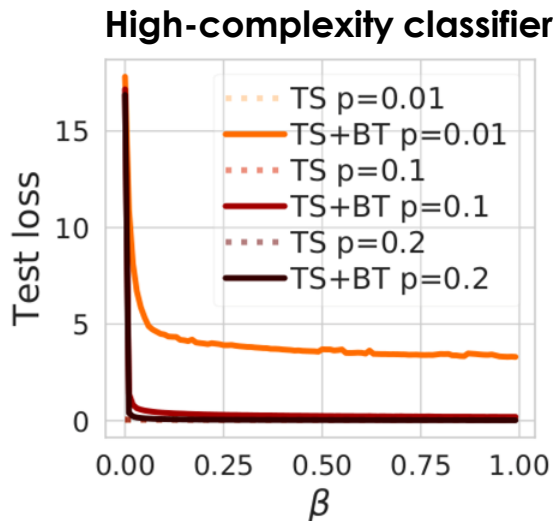
When  $\beta = 1$ , the classifier is learning the clean training samples + the backdoored dataset.

Gradually increasing  $\beta$  allows us to understand how much the learning becomes more difficult when the classifier has to learn not only the clean but also the backdoored dataset, namely how much it is difficult learning the backdoored dataset.

Cinà et al., *Backdoor Learning Curves: Explaining Backdoor Poisoning Beyond Influence Functions*, ArXiv 2021



# Backdoor Learning Curves



Classifier: logistic; Dataset: MNIST 7-1.

$p$  is the percentage of training point containing the backdoor trigger.

These plots show that **for low-complexity classifier, learning backdoors it is a harder task.**

Cinà et al., *Backdoor Learning Curves: Explaining Backdoor Poisoning Beyond Influence Functions*, ArXiv 2021

# Backdoor Impact on Learning Parameters

To better understand why, we can monitor how the classifiers' parameter changes during backdoor learning.

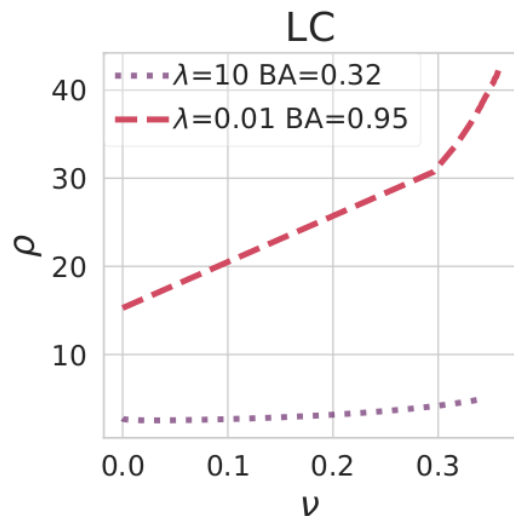
To this end, we have devised two measures:

1.  $\nu$  that quantifies how the orientation of the decision hyperplane changes
2.  $\rho$  that quantifies how much the norm of the classifier weights increases\*

And we have plotted how they change during the creation of the backdoor learning curve for a classifier trained with two levels of complexity.

\* the norm of a classifier weights is proportional to their complexity and inversely proportional to the margin.

# Backdoor Impact on Learning Parameters



Low-complexity classifier  $\lambda = 10$  (strong regularization)  
High-complexity classifier  $\lambda = 0.01$  (low regularization)

This plot shows that, when the classifier learns the backdoored samples its complexity increases.

If the classifier is strongly regularized, it cannot increase much its complexity and, therefore, it cannot learn well the backdoor.

Cinà et al., *Backdoor Learning Curves: Explaining Backdoor Poisoning Beyond Influence Functions*, ArXiv 2021

# Visualizing Influential Training Points

Dataset: MNIST and CIFAR10.

High-complexity classifier SVM RBF.

7 most influential training samples on the prediction of the samples with the red border.



Many backdoored samples are among the most influential for the prediction.

Cinà et al., *Backdoor Learning Curves: Explaining Backdoor Poisoning Beyond Influence Functions*, ArXiv 2021

# Visualizing Influential Training Points

Dataset: MNIST and CIFAR10.

High-complexity classifier SVM RBF.

7 most influential training samples on the prediction of the samples with the red border.



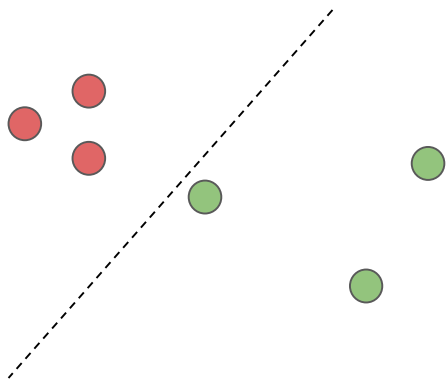
Despite the samples with the red border contain the trigger, there aren't backdoored samples among the most influential for the prediction.

Cinà et al., *Backdoor Learning Curves: Explaining Backdoor Poisoning Beyond Influence Functions*, ArXiv 2021

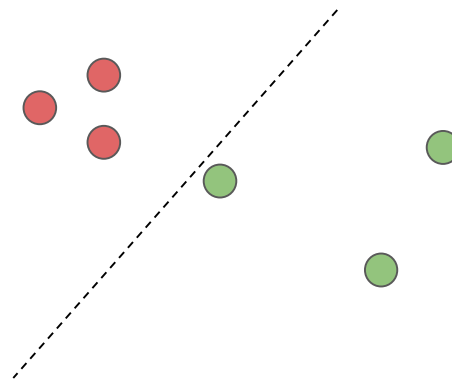
# Data Augmentation against Backdoor

Other papers have proposed alternative approaches that overall reduces the complexity of the learned model. One of these approaches is data augmentation.

Regularization increases the margin altering the training loss function.



Data augmentation increases the margin adding perturbed samples.

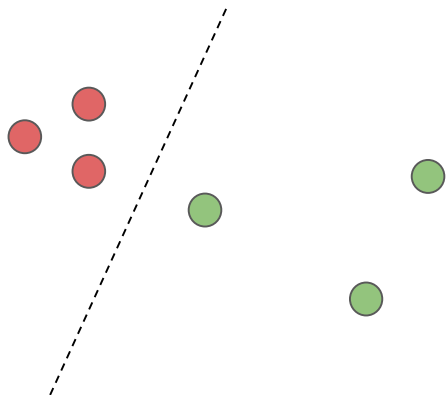


Borgnia et al., *Strong Data Augmentation Sanitizes Poisoning and Backdoor Attacks Without an Accuracy Tradeoff*, ICASSP 2021

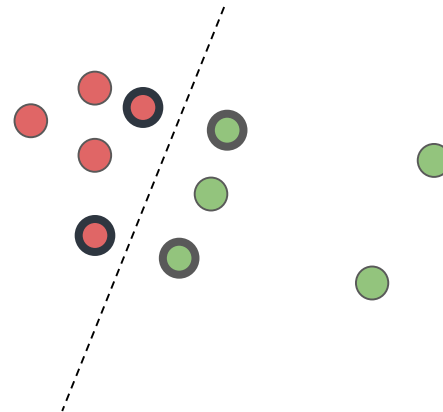
# Data Augmentation against Backdoor

Other papers have proposed alternative approaches that overall reduces the complexity of the learned model. One of these approaches is data augmentation.

Regularization increases the margin altering the training loss function.



Data augmentation increases the margin adding perturbed samples.



Borgnia et al., *Strong Data Augmentation Sanitizes Poisoning and Backdoor Attacks Without an Accuracy Tradeoff*, ICASSP 2021

# Data Augmentation against Backdoor

Augmentation: Mixup and CutMix, both mix the image of two different classes.

Classifier: ResNet-18 trained from scratch on CIFAR-10.

Attack: BadNet

Percentage of training data backdoored between brackets.

	Poison Success (100%)	Validation Accuracy (100%)	Poison Success (10%)	Validation Accuracy (10%)
Baseline	100%	85%	57%	94%
mixup	100%	85%	42%	95%
CutMix	36%	94%	23%	95%

Reduces the backdoor success without compromising the accuracy on the clean data.

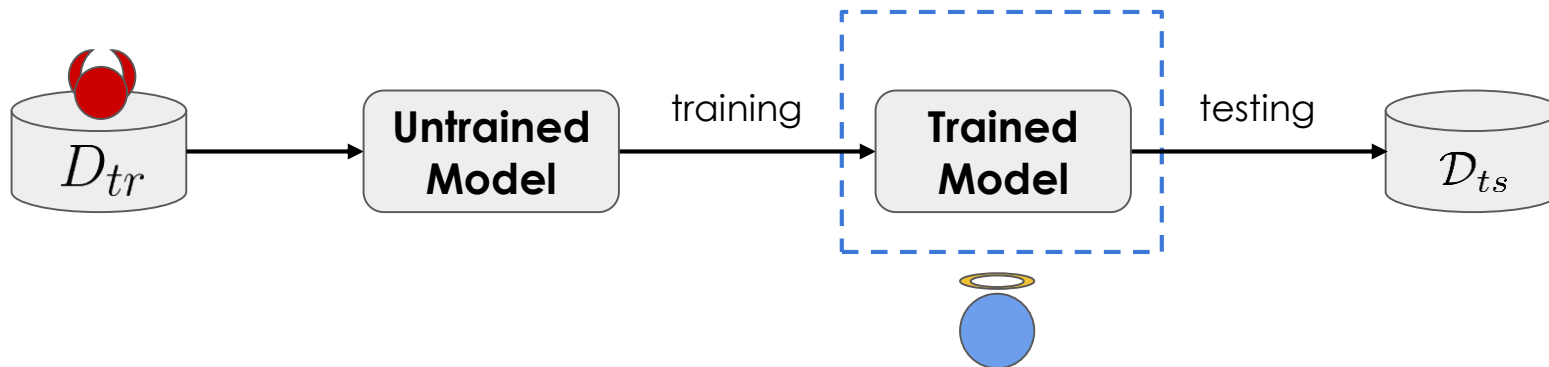


# **Poisoning Integrity Defenses**

## **Test-Time**

# Model Inspection

The defender analyzes the trained model (and eventually also the training data) to detect the poisoning samples.



# Activation Clustering

Distinguishes poisoning from legitimate samples [clustering the activations of the last layer](#).

For each label:

1. computes the activation of all the samples in the poisoned dataset with that label
2. performs dimensionality reduction on the activations
3. clusters them with K-means to divide them into two clusters: poisoning and legitimate

# Activation Clustering

Classifier: CNN with two convolutional and two fully connected layers trained end-to-end.

Dataset: MNIST

Attack: BadNet

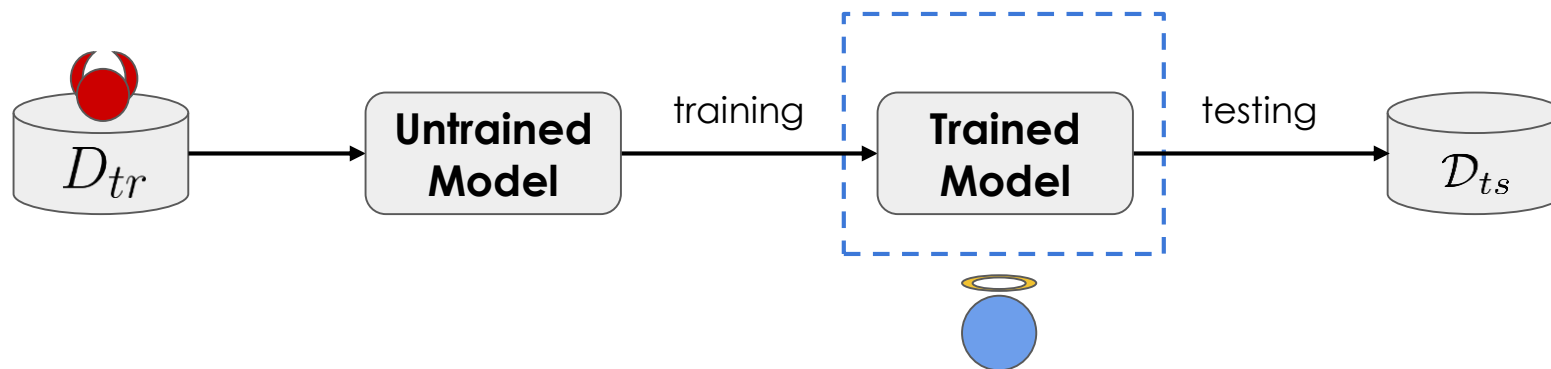
Percentage of detected backdoored samples for class

Target	0	1	2	3	4	5	6	7	8	9	Total
AC Accuracy	99.89	99.99	99.95	100	100	100	99.94	100	100	99.99	99.97

Chen et al., *Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering*, Arxiv 2018

# Trigger Reconstruction

The defender analyzes the trained model (and eventually also the training data) to reconstruct the backdoor trigger used by the attackers.



# Tabor

Formalizes trojan detection as an optimization problem:

$$\operatorname{argmin}_{\Delta, \mathbf{M}} L(f(\mathbf{x}_t), y_t)$$

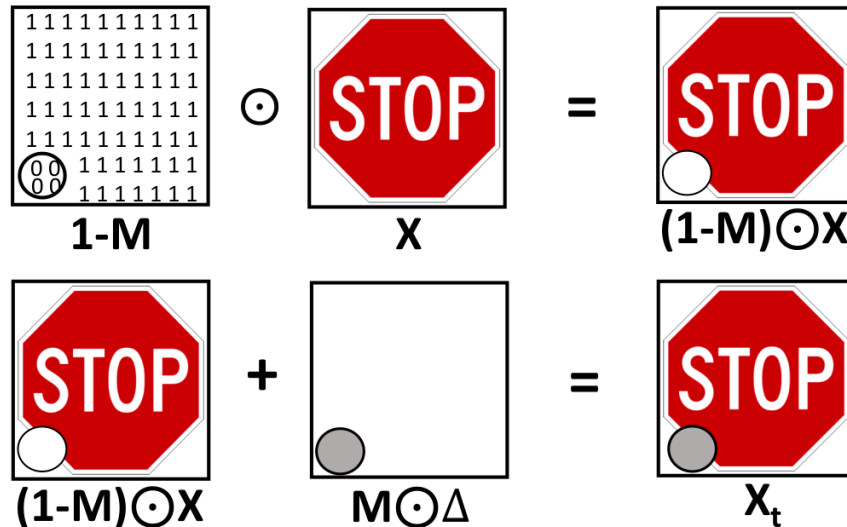
$$\mathbf{x}_t = \mathbf{x} \odot (\mathbf{1} - \mathbf{M}) + \Delta \odot \mathbf{M}$$

Searches the trigger that minimizes the loss w.r.t. the target class of the test sample + the trigger.

where:

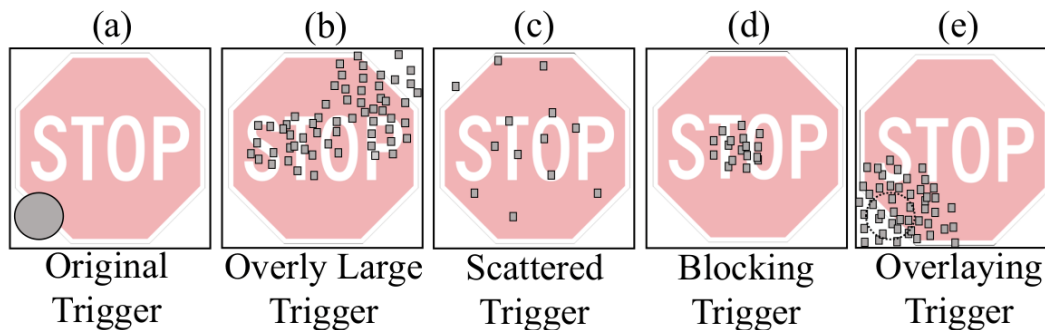
- $\mathbf{M}$  is the mask  
(trigger shape and location)
- $\Delta$  is a pattern  
(trigger color)
- $\mathbf{x}_t$  is a test sample
- $y_t$  is the target class

Multiplied together  $\mathbf{M} \odot \Delta$  they denote the **restored trigger**.



# Tabor

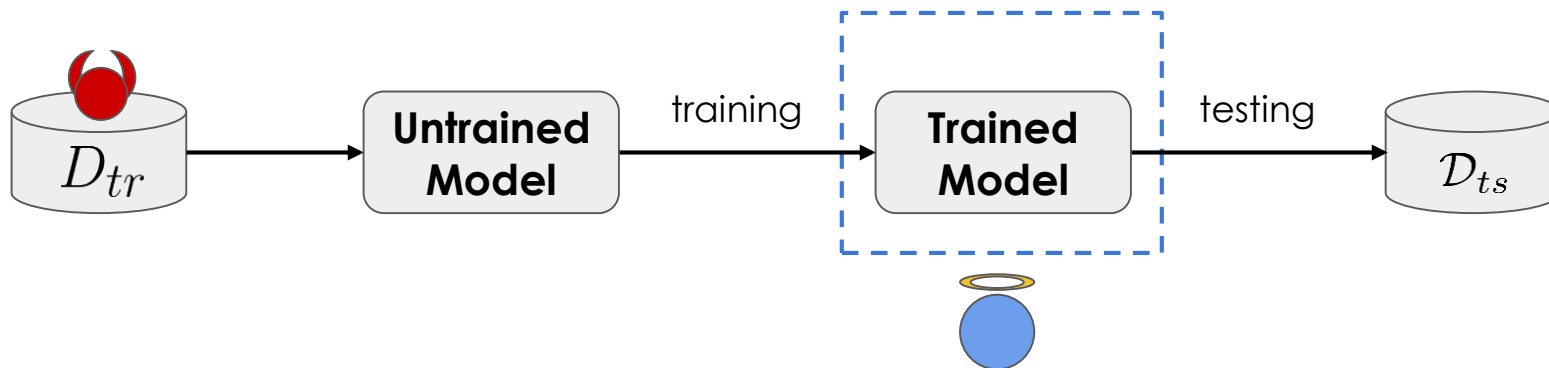
The authors show that solving that problem, you often end up with a trigger that belongs to one of these categories:



Therefore, they add some regularization terms to the original loss function to discourage these situations and restore the corresponding trigger as accurately as possible.

# Model Sanitization

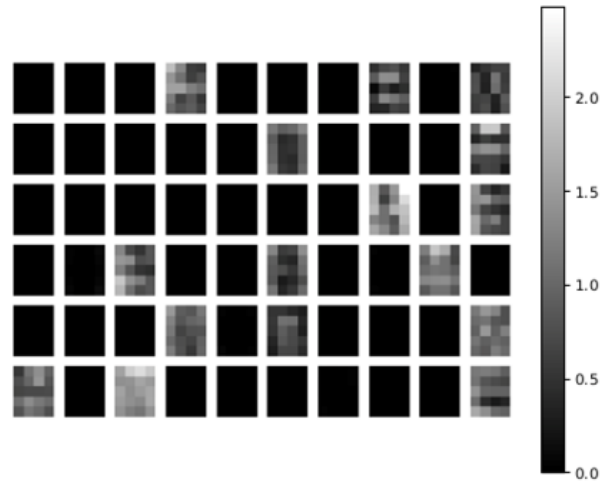
The defender removes the effect of the poisoning samples on the trained model.



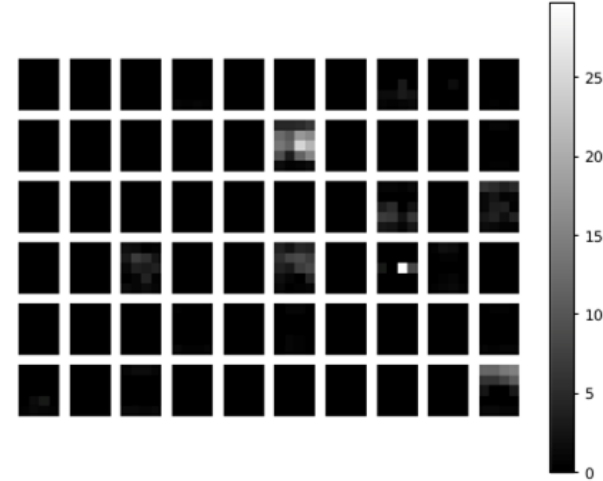


# Fine Pruning

Backdoored DNN misbehave on backdoored inputs while still behaving on clean inputs. To make this possible, some of their filters must be dedicated to the backdoor.



(a) Clean Activations (baseline attack)



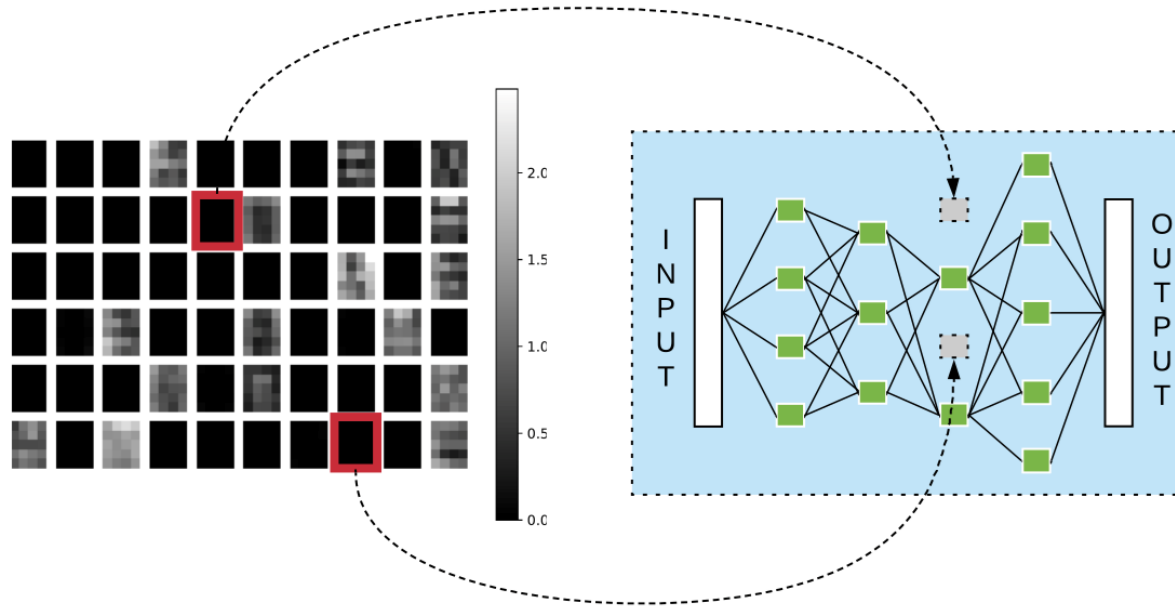
(b) Backdoor Activations (baseline attack)

The plots below shows the average activations of neurons in the final convolutional layer of a backdoored face recognition DNN for clean and backdoor inputs.

Liu et al., *Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks*, RAID 2018

# Fine Pruning

The neurons that compose these filters are dormant in the presence of clean-input. Fine-pruning **detects and prune the most dormant neuron** of the DNN.



Liu et al., *Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks*, RAID 2018

# Fine Pruning

Dataset: Face Dataset; 1283 identities; 100 images for each identity.

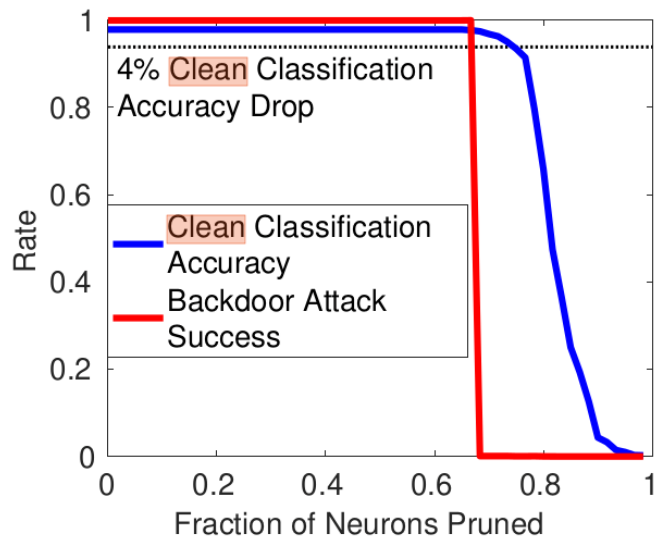
Network: DeepID

Attack: similar to BadNet, but the backdoor depicts eyeglasses.

randomly selects 180 identities and add a backdoor to their images.



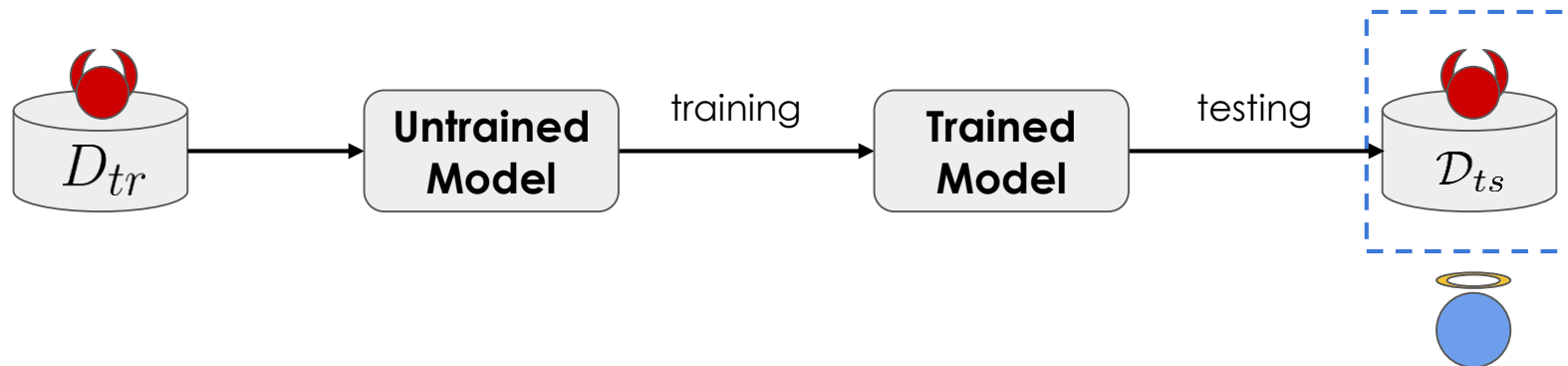
Backdoored image



Liu et al., *Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks*, RAID 2018

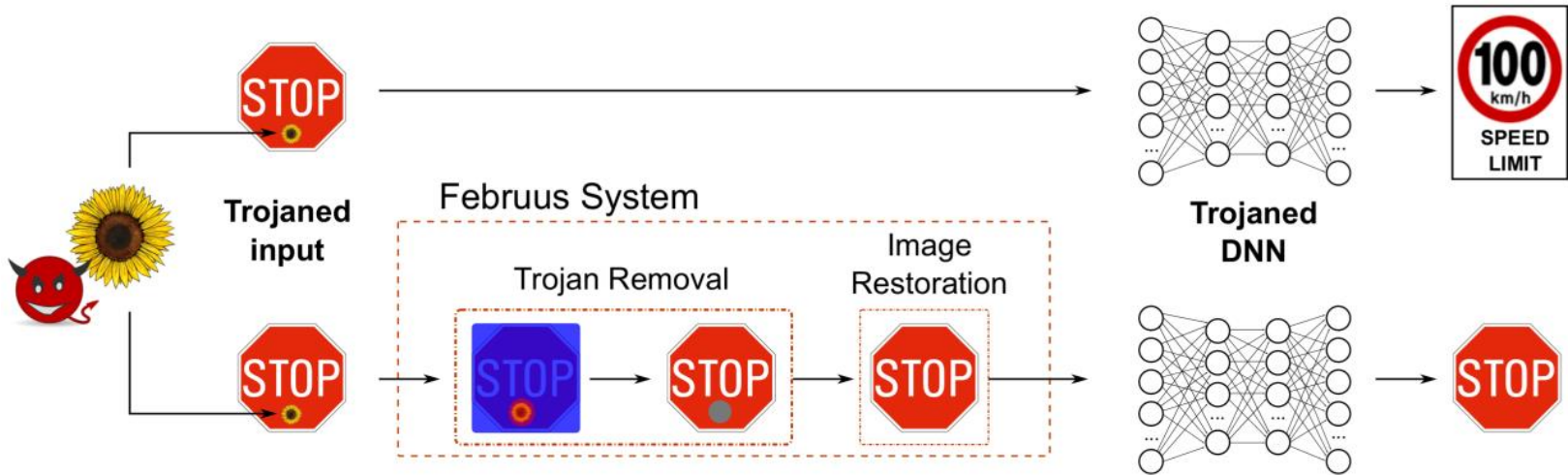
# Test Data Sanitization

The defender analyzes the test data and removes the trigger.



# Febbrus

First, detects and removes the trigger from the test image. Then, it restores the test image.

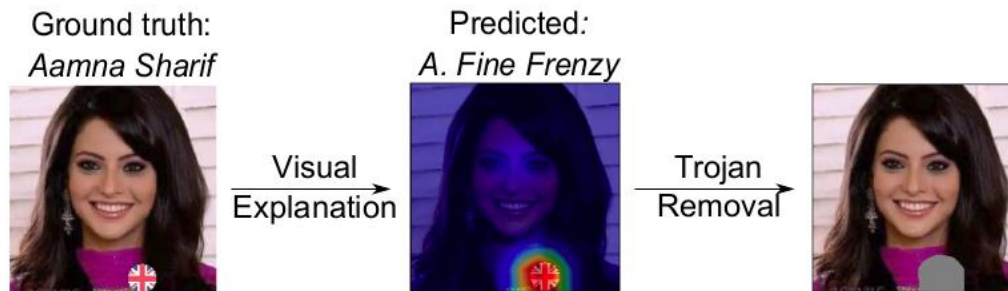


Doan et al., *Febbrus: Input Purification Defense Against Trojan Attacks on Deep Neural Network*, ACSAC 2020

# Febbrus

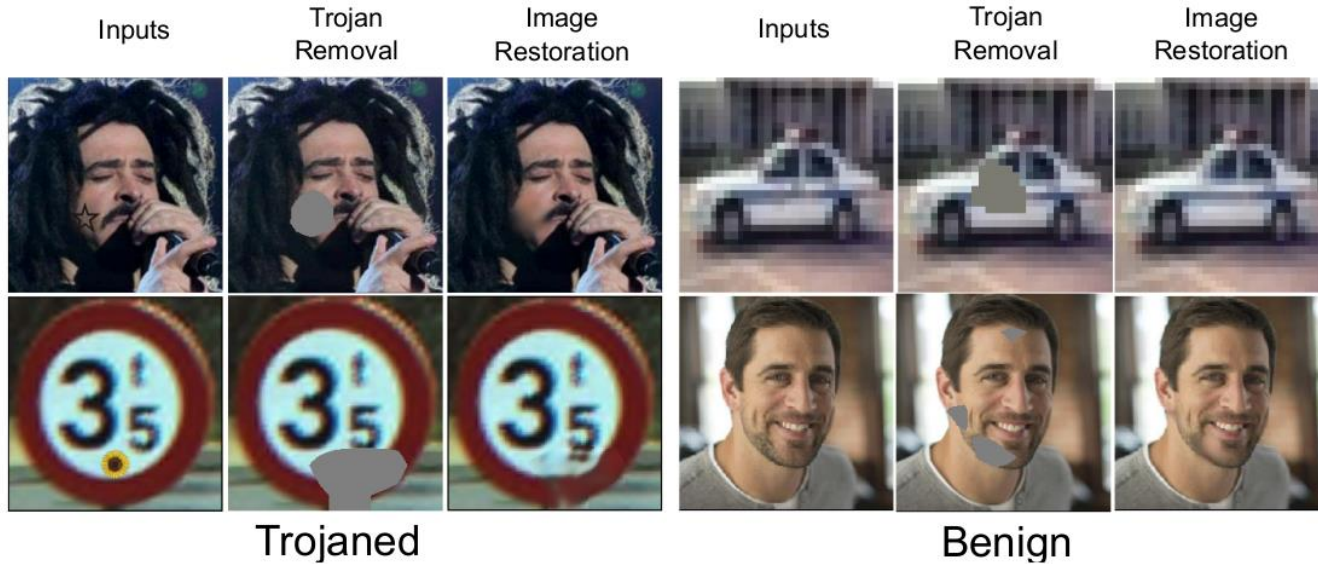
To **detect the trigger** generates a heatmap of the input regions that contribute heavily to the classifier decision.

If there is only small region of the image with a strong contribution, it is quite likely it is the one that contains the trigger.



# Febbrus

To restore the image, they use a Generative Adversarial Network.



Doan et al., *Febbruus: Input Purification Defense Against Trojan Attacks on Deep Neural Network*, ACSAC 2020

# Febbrus

Task/Dataset	# of Labels	# of Training Images	# of Testing Images	Model Architecture
CIFAR10	10	50,000	10,000	6 Conv + 2 Dense
VGGFace2	170	48,498	12,322	13 Conv + 3 Dense (VGG-16)

Task/Dataset	Benign Model	Trojaned Model (Before Febbrus)		Trojaned Model (After Febbrus)	
	Classification Accuracy	Classification Accuracy	Attack Success Rate	Classification Accuracy	Attack Success Rate
CIFAR10	90.34%	90.79%	100%	90.08%	0.25%
VGGFace2	91.84%	91.86%	100%	91.78%	0.00%

Doan et al., *Febbrus: Input Purification Defense Against Trojan Attacks on Deep Neural Network*, ACSAC 2020



# **Sponge Attacks**

# Sponge Attacks

**Goal:** Delay predictions (can be a problem in real-time systems, e.g., autonomous vehicles)

The authors found that some samples, called “**sponge samples**” require a higher energy consumption to be classified.

The amount of energy consumed by one inference pass (i.e. a forward pass in a neural network) depends primarily on:

- The overall number of arithmetic operations required to process the inputs;
- The number of memory accesses e.g., to the GPU DRAM.

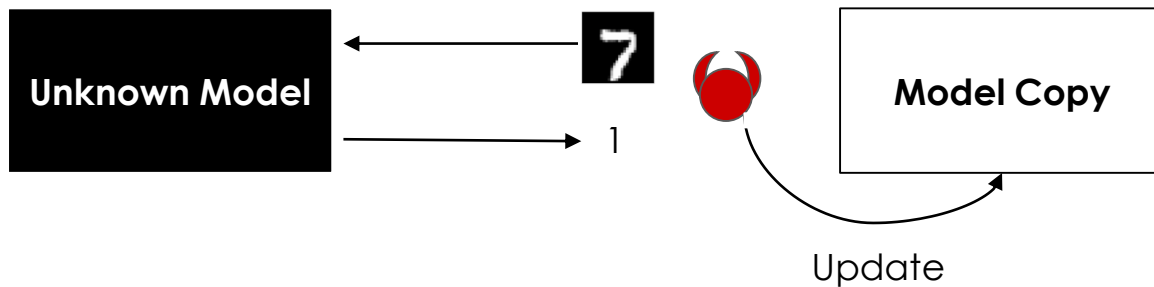
The authors propose using a genetic algorithm to create sponge examples.

# Privacy Attacks

# Model Extraction/Stealing

## Privacy Attacks

Construct a copy of a model, being able to query the target model.



# Model Extraction/Stealing

## *Privacy Attacks*

Two possible goals:

**Task accuracy** - create a copy that can perform well the original task  
(stealing intellectual property)

**High fidelity**- create a copy that makes the same errors as the original model  
(having a surrogate similar to the original, e.g., to  
compute attacks.)

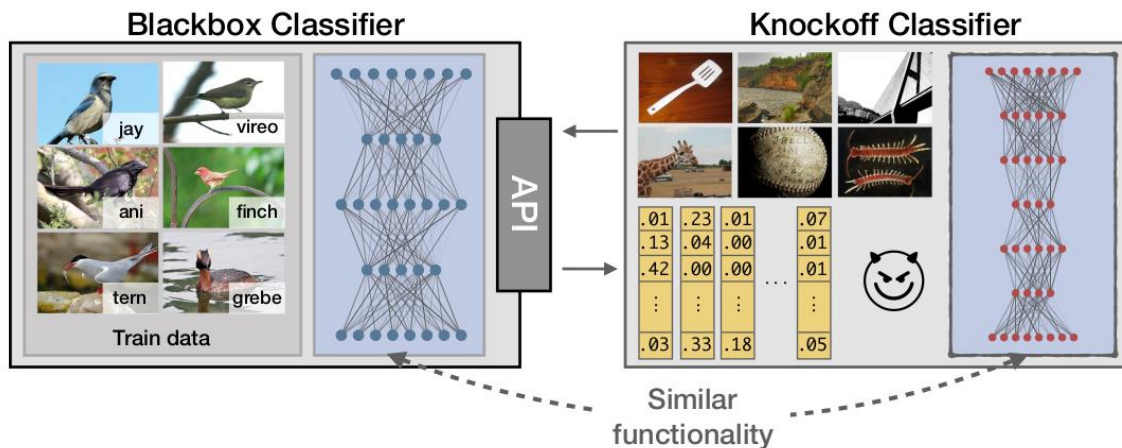
# Model Extraction/Stealing [task accuracy]

## Privacy Attacks

They use a two-step approach that:

1. query the target model and collect output
2. train the surrogate

Interestingly, they found that by querying the model with random images taken from a distribution different from the one of the training set, they can build an accurate surrogate.



Orekondy et al., *Knockoff Nets: Stealing Functionality of Black-Box Models*, CVPR 2019

# Model Extraction/Stealing [task accuracy]

## Privacy Attacks

Using randomly chosen images, it is possible creating a surrogate but it is necessary performing a lot of queries.

They propose a technique to improve the [efficiency of the query](#).

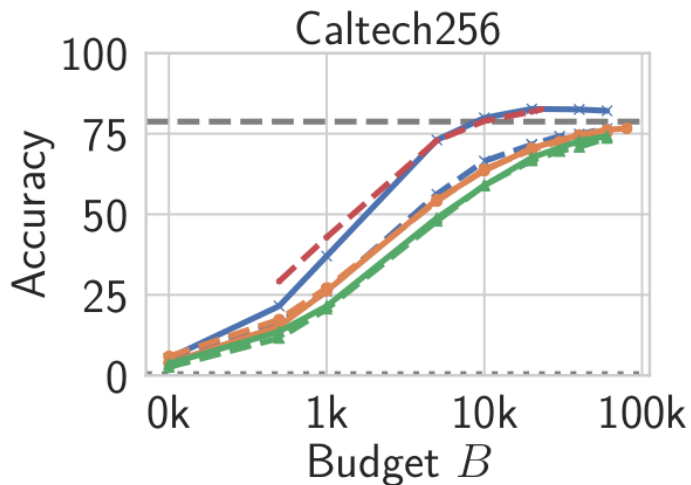
This technique [encourages images where the victim is confident](#) (images similar to the ones of the training dataset).

They test the proposed method on deep neural networks and suggest using complex architectures.

# Model Extraction/Stealing [task accuracy]

## Privacy Attacks

Accuracy of the surrogate model for an increasing number of queries (Budget), when the attacker uses as starting point various datasets (one line for dataset) different from the one on which the model was trained.





# Defenses against Model Extraction/Stealing

## *Privacy Attacks*

Makes the model learn some **watermarks** that can be used to **prove the intellectual property** of the model.

The authors create the **watermarks** so that they are entangled with legitimate data.

In this way, an attacker, to remove the watermarks, has to sacrifice the model performances.

# Defenses against Model Extraction/Stealing

## Privacy Attacks

Classifiers: a small CNN for the MNIST and an RNN for the Speech Command dataset.

The approach proposed in [1] is called EWE, Baseline is an approach previously proposed.

Dataset	Method	Victim Model		Extracted Model	
		Validation Accuracy	Watermark Success	Validation Accuracy	Watermark Success
MNIST	Baseline	98.28( $\pm 0.57$ )%	100.00( $\pm 0.00$ )%	98.35( $\pm 0.29$ )%	3.09( $\pm 2.80$ )%
	EWE	97.75( $\pm 0.59$ )%	99.85( $\pm 0.24$ )%	97.58( $\pm 0.78$ )%	61.44( $\pm 27.85$ )%
Fashion MNIST	Baseline	90.22( $\pm 0.27$ )%	100.0( $\pm 0.00$ )%	89.43( $\pm 0.41$ )%	5.75( $\pm 2.66$ )%
	EWE	90.42( $\pm 1.03$ )%	99.88( $\pm 0.31$ )%	89.45( $\pm 0.93$ )%	44.90( $\pm 24.70$ )%
Speech Command	Baseline	97.00( $\pm 4.31$ )%	100.00( $\pm 0.00$ )%	96.78( $\pm 4.96$ )%	22.58( $\pm 25.09$ )%
	EWE	96.19( $\pm 0.38$ )%	100.00( $\pm 0.00$ )%	96.65( $\pm 0.53$ )%	68.16( $\pm 28.30$ )%

[1] Jia et al., *Entangled Watermarks as a Defense against Model Extraction*, ArXiv 2020

# Defenses against Model Extraction/Stealing

## *Privacy Attacks*

**Goal:** To understand that an attacker is querying the model to perform model stealing.

PRADA: *analyzes* the distribution of consecutive API *queries* made by a client and raises an alarm when this distribution deviates from benign behavior.

The queries usually made to perform model extraction are made ad-hoc to maximize the information extracted. Therefore *the attacker makes consecutive queries with similar inputs*,  
whereas the benign clients usually make queries with quite different samples.

# Defenses against Model Extraction/Stealing

## Privacy Attacks

Model: a small Convolutional Neural Network.

Model ( $\delta$ value)	FPR	Queries made until detection			
		TRAMER	PAP.	T-RND	COLOR
<b>MNIST</b> (0.95)	0.0%	5,560	120	140	-
<b>MNIST</b> (0.96)	0.0%	5,560	120	130	-
<b>GTSRB</b> (0.87)	0.0%	5,020	430	missed	550
<b>GTSRB</b> (0.90)	0.6%	5,020	430	missed	480
<b>GTSRB</b> (0.94)	0.1%*	5,020	430	440	440

Juuti et al., *PRADA: Protecting Against DNN Model Stealing Attacks*, ArXiv 2018

# Defenses against Model Extraction/Stealing

## Privacy Attacks

Attack	Model	MNIST ( $\delta = 0.96$ )		
		PAPERNOT	T-RND	TRAMER
Original queries		1,600	1,600	10,000
Additional queries		14,274	4,764	79,980
Overhead		+890%	+300%	+800%

Attack	Model	GTSRB ( $\delta = 0.90$ )			
		PAPERNOT	T-RND *	TRAMER	COLOR
Original queries		6,680	6,680	10,000	6,680
Additional queries		41,160	19,986	99,990	39972
Overhead		+620%	+300%	+1000%	+600%

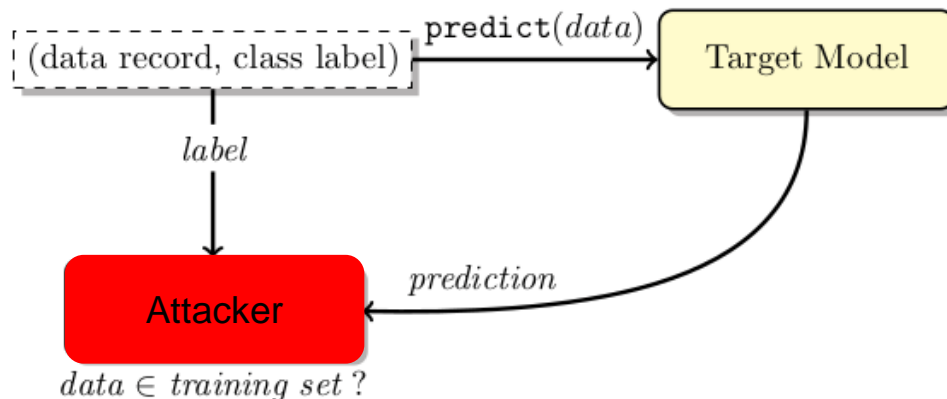
Juuti et al., *PRADA: Protecting Against DNN Model Stealing Attacks*, ArXiv 2018

# Membership Inference Attacks

## Privacy Attacks

**Goal:** identify whether an input sample is part of the training set used to learn a deep neural network having query access to the target model.

First, they query the deep neural network with the input sample, and they get the predictions.



Shokri et al., *Membership Inference Attacks Against Machine Learning Models*, S&P 2017

# Membership Inference Attacks

## *Privacy Attacks*

Then, the attacker crafts many surrogate models that imitates the behavior of the original deep neural network and trains them:

- including the test samples in the training dataset
- not including the test samples in the training dataset

The authors show that by comparing the predictions of those models with the one of the original deep network is possible to understand if the test samples were or were not part of the original deep network training dataset.

# Defenses against Membership Inference Attacks

Different types of defenses that have been proposed against this attack:

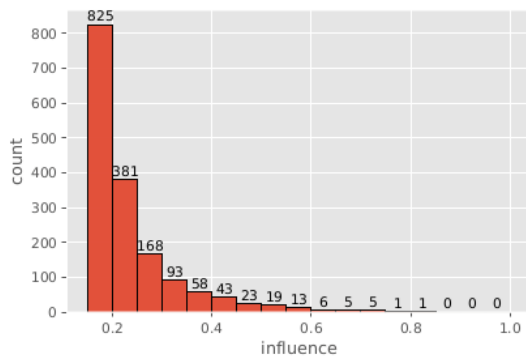
- regularization
- differential privacy



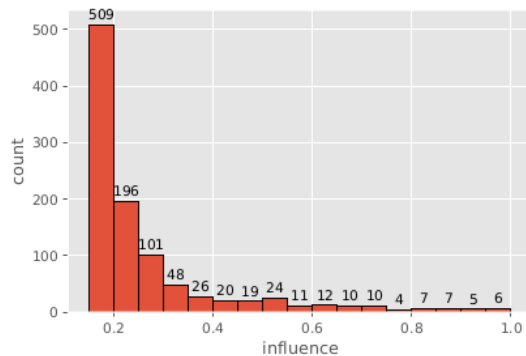
# Regularization against Membership Inference Attacks

Deep neural networks tend to memorize training data [1].

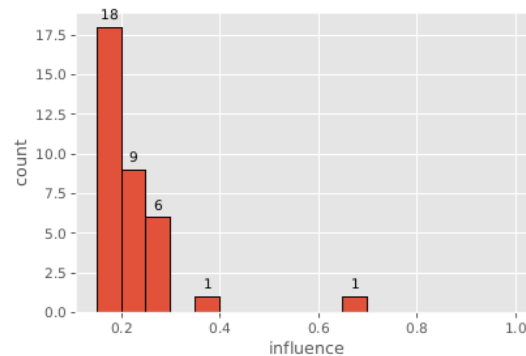
Classifiers ResNet50 trained on Imagenet and on CIFAR-100, a small CNN trained on MNIST. Most influent (memorized) samples for different datasets:



(a) ImageNet



(b) CIFAR-100



(c) MNIST

Regularizing the model reduces the memorization of the training data.

Shokri et al., *Membership Inference Attacks Against Machine Learning Models*, S&P 2017

# Differential Privacy against Membership Inference

Ideally, the users of an ML system should not be able to infer information about it.

Differential privacy provides guarantees on the impact that single data records have on the output of a model.

The defense based on differential privacy basically tries to bound this impact.

# Differential Privacy against Membership Inference

Differential privacy offers a trade-off between privacy protection and utility or model accuracy.

Although differential privacy is the most studied defense against this attack, different studies concluded that the models could offer privacy protection only when they considerably sacrifice their utility.

# Model Inversion Attacks

## Privacy Attacks

**Goal:** reconstruct training samples having the ability to query the model.

**Problem:** find the input that maximizes the returned confidence w.r.t. the target label.

Solving this problem using gradient descent, they are able to reconstruct a training image.



Fredrikson et al, *Model inversion attacks that exploit confidence information and basic countermeasures*, ACM CCS, 2015

# Model Inversion Attacks

## *Privacy Attacks*

To validate the proposed approach, the authors conducted a study involving humans.

The authors Asked Mechanical Turk workers to **match the reconstructed image** to one of five face images from the original set, **or to respond that the displayed image does not correspond** to one of the five images.

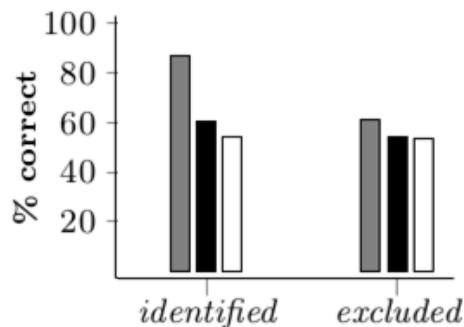
# Model Inversion Attacks

## Privacy Attacks

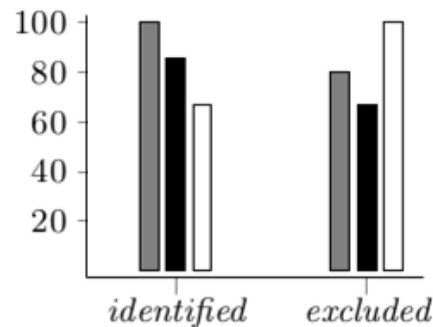
Study conducted using different surrogate models (bar colors) to reconstruct the image.

“identified” -> the true identity was displayed between the 5 and identified by the worker

“excluded” -> the identity was not present, and the worker correctly said it was not present



(a) Average over all responses.



(c) Accuracy with skilled workers.

# Other Privacy Attacks

## Property Inference

**Goal:** extract dataset properties not encoded as features or not correlated to the learning task.

E.g., extract the ratio of women and men patients in a dataset used to train a model even if the gender was not one of the dataset features.

## Attribute Inference

**Goal:** Infer the value of a sensible attribute.

# Attacks against Machine Learning

	Attacker's Goal		
	Misclassifications that do not compromise normal system operation	Misclassifications that compromise normal system operation	Querying strategies that reveal confidential information on the learning model or its users
Attacker's Capability	Integrity	Availability	Privacy / Confidentiality
Test data	Evasion (a.k.a. adversarial examples)	<b><i>Sponge Attacks</i></b>	<b><i>Model extraction / stealing</i></b> <b><i>Model inversion (hill climbing)</i></b> <b><i>Membership inference</i></b>
Training data	<b><i>Integrity Poisoning (to allow subsequent intrusions) – e.g., backdoors</i></b>	<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)



# The Attacks Most Feared by Industry

<i>Which attack would affect your org the most?</i>	<i>Distribution</i>
Poisoning (e.g: [21])	10
Model Stealing (e.g: [22])	6
Model Inversion (e.g: [23])	4
Backdoored ML (e.g: [24])	4
Membership Inference (e.g: [25])	3
Adversarial Examples (e.g: [26])	2
Reprogramming ML System (e.g: [27])	0
Adversarial Example in Physical Domain (e.g: [5])	0
Malicious ML provider recovering training data (e.g: [28])	0
Attacking the ML supply chain (e.g: [24])	0
Exploit Software Dependencies (e.g: [29])	0

... at least between the ones developed so far..

# A Neverending Arms Race

The industry's interest in ML is increased in the last few years.

New threats are being quickly discovered...

New defenses are developed...

But the attacks are adapted to overcome the new defenses.

# Defend against Known Attacks

We cannot prevent the creation of new attacks..

We cannot defend against unknown attacks..

But we can do our best to defend our ML system, at least against the known attacks.

# Defend against Known Attacks


Defending against all the know attacks might be time and computationally expensive.

However, we can wisely choose on which is worth investing our effort depending on:

- the damage that the attack can cause
- the likelihood that we will face that attack.



Ambra Demontis  
ambra.demontis@unica.it

 @ambrademontis

# Thanks!



*If you know the enemy and know yourself, you need not fear  
the result of a hundred battles*  
Sun Tzu, The art of war, 500 BC