CS489/689
Privacy, Cryptography, Network and Data Security

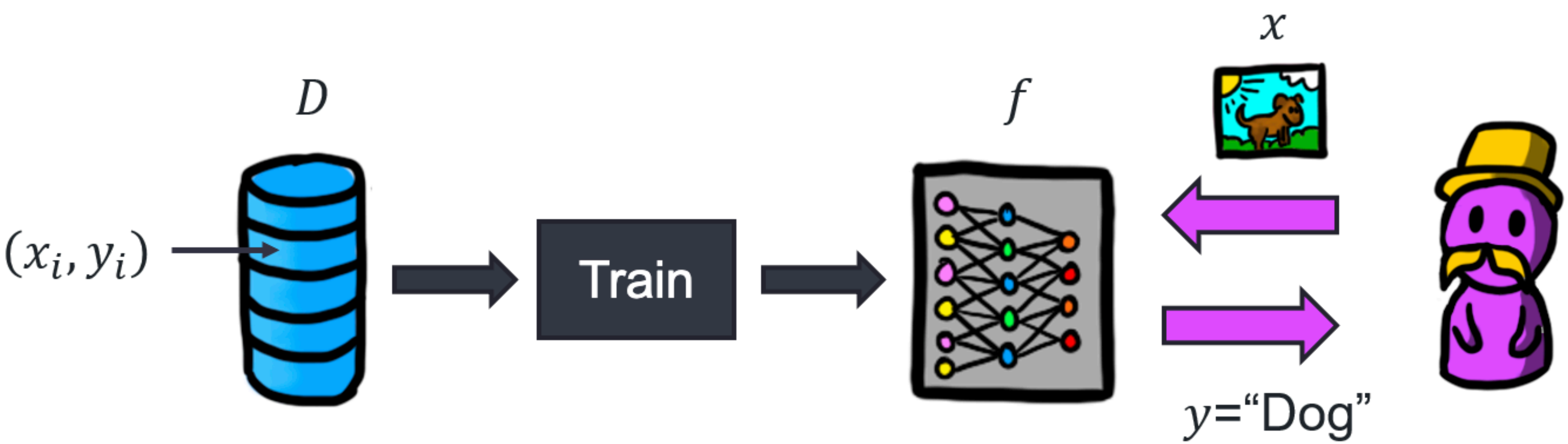
Winter 2023, Tuesday/Thursday 8:30-9:50am

Adversarial Machine Learning

Attacking Machine Learning and its data

Lucas Fenaux | 03/23/2023

Machine Learning - Recap



Machine Learning - Recap

ML model is a learned, parametrized function. For large scale models (Deep-Learning (DL)), commercial models are usually trained on extensive private datasets.

Machine Learning - Recap

ML model is a learned, parametrized function. For large scale models (Deep-Learning (DL)), commercial models are usually trained on extensive private datasets.

There are three main forms of ML:

Machine Learning - Recap

ML model is a learned, parametrized function. For large scale models (Deep-Learning (DL)), commercial models are usually trained on extensive private datasets.

There are three main forms of ML:

- Supervised: classification, tokenized generation methods (ChatGPT)

Machine Learning - Recap

ML model is a learned, parametrized function. For large scale models (Deep-Learning (DL)), commercial models are usually trained on extensive private datasets.

There are three main forms of ML:

- Supervised: classification, tokenized generation methods (ChatGPT)
- Unsupervised: clustering, synthetic data generation

Machine Learning - Recap

ML model is a learned, parametrized function. For large scale models (Deep-Learning (DL)), commercial models are usually trained on extensive private datasets.

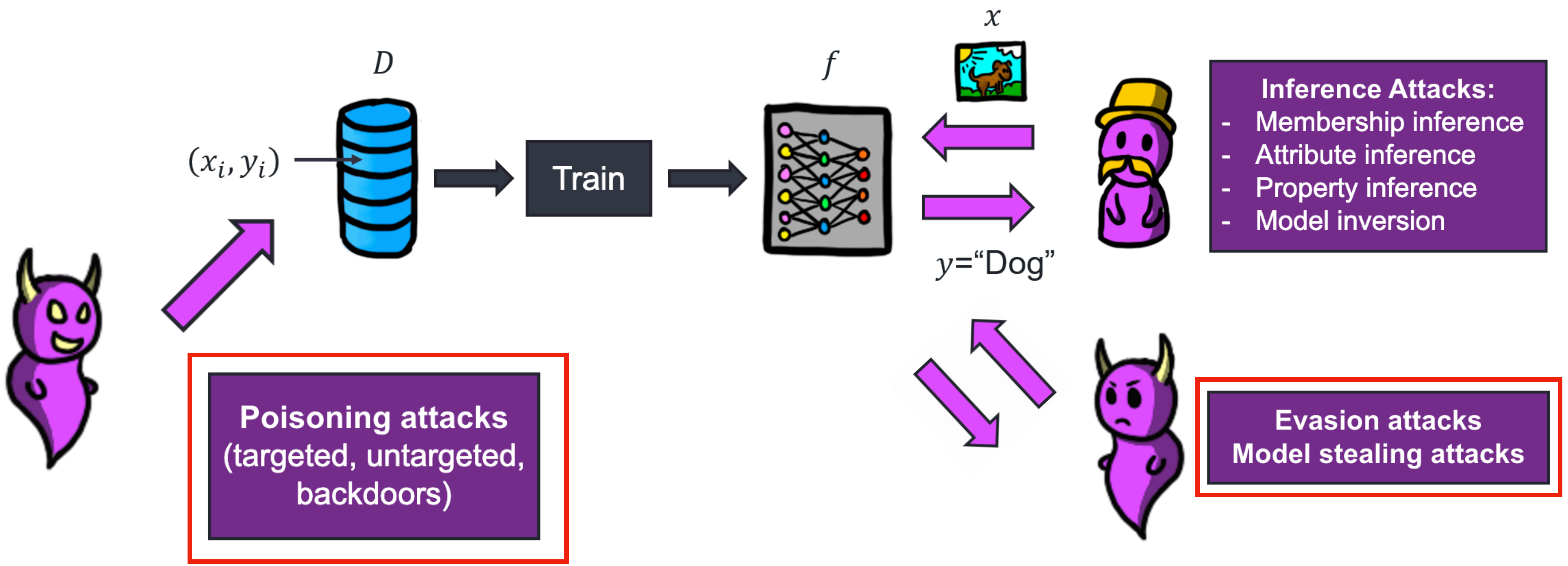
There are three main forms of ML:

- Supervised: classification, tokenized generation methods (ChatGPT)
- Unsupervised: clustering, synthetic data generation
- Reinforcement Learning: games (Chess, Go, Poker…), robotics

Attacking Machine Learning

What is there to attack?

Machine Learning - Attacks recap



Part 1: Intellectual Property

Intellectual Property - Topics

- Machine Learning as a service
- Model Stealing
  - Introduction & Motivation
  - Attacks
  - Defenses
- IP protection
  - Watermarking
  - Fingerprinting
- Model Inversion

Machine Learning as a Service

- Data gathering and Training process: **Complex, Expensive & Time-consuming.**

Machine Learning as a Service

- Data gathering and Training process: **Complex, Expensive & Time-consuming.**
  - In particular, for classification, labeling has to be done by humans (as otherwise why not use whatever labelling method you have rather than machine learning).

Machine Learning as a Service

- Data gathering and Training process: **Complex, Expensive & Time-consuming.**
  - In particular, for classification, labeling has to be done by humans (as otherwise why not use whatever labelling method you have rather than machine learning).
- Solution: Machine Learning-as-a-Service (MLaaS).
  - Offer model as queryable black-box service (ChatGPT).
    - Requires significant computing capabilities to provide accessible service
  - If frequent queries are necessary, can become quite expensive for the user.

Machine Learning as a Service

- Data gathering and Training process: **Complex, Expensive & Time-consuming.**
  - In particular, for classification, labeling has to be done by humans (as otherwise why not use whatever labelling method you have rather than machine learning).
- Solution: Machine Learning-as-a-Service (MLaaS).
  - Offer model as queryable black-box service (ChatGPT).
    - Requires significant computing capabilities to provide accessible service
  - If frequent queries are necessary, can become quite expensive for the user.
- Malicious Solution: **Steal someone's else's MLaaS model.**

Model Stealing

Model Stealing: What?

Model Stealing - What is there to steal?

- **Approximation of the behaviour of the model**

Model Stealing - What is there to steal?

- **Approximation of the behaviour of the model**
- Model architecture

Model Stealing - What is there to steal?

- **Approximation of the behaviour of the model**
- Model architecture
- Learned parameters

Model Stealing - What is there to steal?

- **Approximation of the behaviour of the model**
- Model architecture
- Learned parameters
- Training hyper-parameters

Model Stealing: How?

Model Stealing - Simple attack

Approximating the behaviour of the model:

- Let $f(x, \theta) = y$ represent the model we are trying to steal. It is a learned parametrized function f with parameters θ that was trained on a dataset $D = (X, Y)$.
- Assume we have some unlabeled auxiliary dataset $D' = (X', \cdot\ )$ that could be significantly smaller than D.

Model Stealing - Simple attack

Approximating the behaviour of the model:

- Let $f(x, \theta) = y$ represent the model we are trying to steal. It is a learned parametrized function f with parameters θ that was trained on a dataset $D = (X, Y)$.
- Assume we have some unlabeled auxiliary dataset $D' = (X', \cdot\ )$ that could be significantly smaller than D.
- We create our own model f' with parameters θ' and create labels for it as $f(X') = Y'$.

Model Stealing - Simple attack

Approximating the behaviour of the model:

- Let $f(x, \theta) = y$ represent the model we are trying to steal. It is a learned parametrized function f with parameters θ that was trained on a dataset $D = (X, Y)$.
- Assume we have some unlabeled auxiliary dataset $D' = (X', \cdot\ )$ that could be significantly smaller than D.
- We create our own model f' with parameters θ' and create labels for it as $f(X') = Y'$.
- We can now train our model with $D' = (X', Y')$.

Model Stealing - Literature

Information	Paper	Approach	Reducing Query	Recovery Rate (%) for Models					
				SVM	DT	LR	kNN	CNN	DNN
Parameter	Tramer *et al.* [160]	ES	-	99	99	99	-	-	99
Hyper-par	Wang *et al.* [165]	ES	-	99	-	99	-	-	-
Arch.	Joon *et al.* [119]	MM	KENNEN-IO	-	-	-	-	-	88
Decision.	Papernot *et al.* [128]	SM	reservoir sampling [163]	-	-	-	-	-	84
	Papernot *et al.* [127]	SM	reservoir sampling [163]	83	61	89	85	-	89
	PRADA [84]	SM	-	-	-	-	-	-	67
Func.	Silva *et al.* [45]	SM	-	-	-	-	-	98	-
	Orekondy *et al.* [122]	SM	random, adaptive sampling	-	-	-	-	98	-

https://www.mlsecurity.ai/post/what-is-model-stealing-and-why-it-matters

Defending Against Model Stealing

Defending Against Model Stealing

It’s … hard.

- There is no known effective pure ML defense.

Defending Against Model Stealing

It’s … hard.

- There is no known effective pure ML defense.
- Existing methods:
  - Daily limit for requests -> makes it more time consuming

Defending Against Model Stealing

It’s … hard.

- There is no known effective pure ML defense.

- Existing methods:

  - Daily limit for requests -> makes it more time consuming
  - <u>The legal system exists!</u>
    - Let’s try to use it

How can we use the legal system

Intellectual Property

An ML model can be considered intellectual property. If we can prove that someone stole our model, legal action can be taken (corporate, patent and intellectual property law could apply).

Intellectual Property

An ML model can be considered intellectual property. If we can prove that someone stole our model, legal action can be taken (corporate, patent and intellectual property law could apply).

- How could one go at proving ownership?

Intellectual Property

An ML model can be considered intellectual property. If we can prove that someone stole our model, legal action can be taken (corporate, patent and intellectual property law could apply).

- How could one go at proving ownership?
  - Have some method to identify a model, even if it is a stolen copy.
- Can also prevent misuse (deep-fakes, fake-news…)

Watermarking

Watermarking - Introduction

Goal: indicate ownership of an object.

Usual use-case: indicating copyright for images/videos by using a company logo.

What if we could do the same for DNNs?

Watermarking - Definition

Def: DNN watermarking is a method designed to detect surrogate models. Watermarking embeds a message into a model that is later extractable using a secret key. (N. Lukas)

Watermarking - Definition

Def: DNN watermarking is a method designed to detect surrogate models. Watermarking embeds a message into a model that is later extractable using a secret key. (N. Lukas)

Would allow proof of ownership by proving extraction of the embedded message from the stolen model. Legal action can then be taken.

Watermarking Scheme - Definition

Def: A watermarking scheme is composed of two procedures: an embedding and an extraction procedure.

Watermarking Scheme - Definition

Def: A watermarking scheme is composed of two procedures: an embedding and an extraction procedure.

- Embed(T, m, M): Takes a watermarking key T, a message $m \subset \{0,1\}$ and a model M and outputs a marked model $\hat{M}$ embedded with a message m.

Watermarking Scheme - Definition

Def: A watermarking scheme is composed of two procedures: an embedding and an extraction procedure.

- Embed(T, m, M): Takes a watermarking key T, a message $m \subset \{0,1\}$ and a model M and outputs a marked model $\hat{M}$ embedded with a message m.

- Extract(T, M): Takes a watermarking key T, a model M and outputs the message $m \subset \{0,1\}$ extracted from model M using key T.

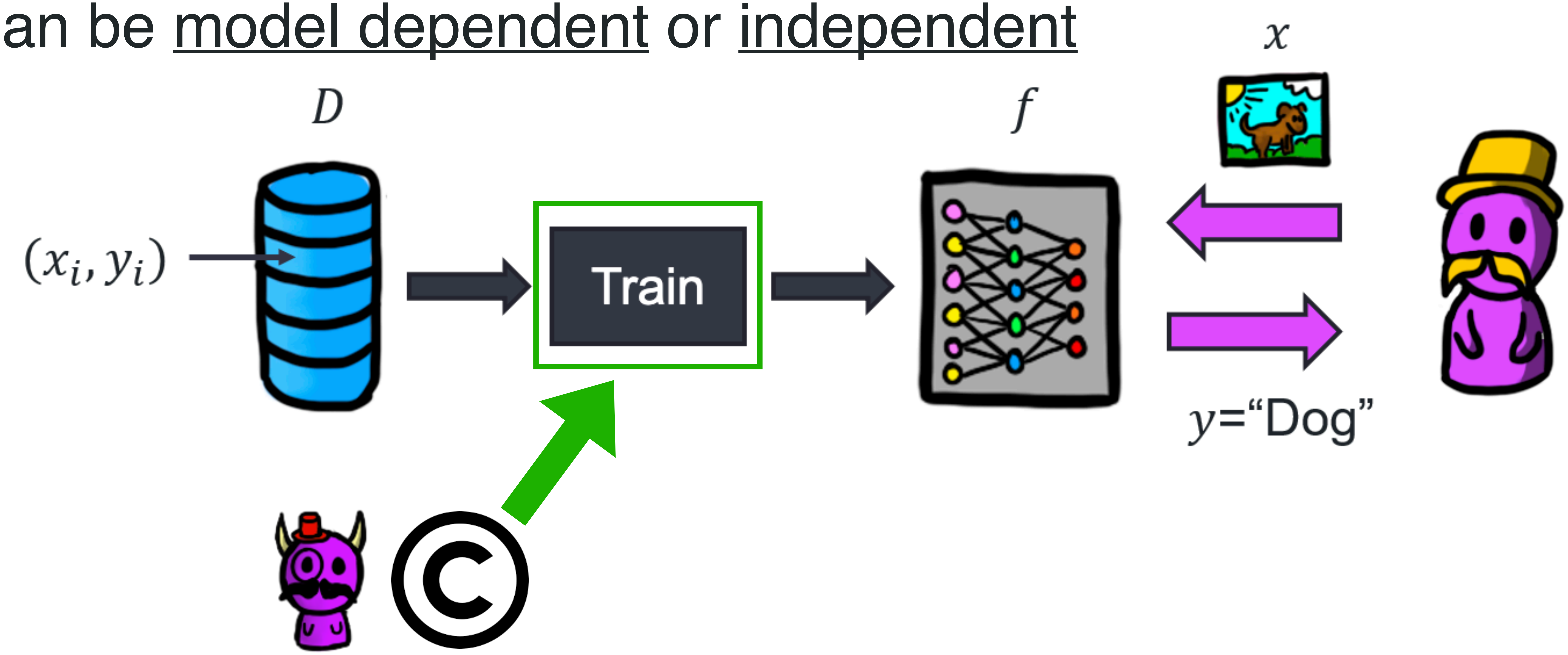
Watermarking - Ideal Requirements

Requirements	Description
Fidelity	The impact on the model's task accuracy is small.
Robustness	Surrogate models retain the watermark.
Integrity	Models trained without access to the source model do not retain the watermark.
Capacity	The watermark allows encoding large messages sizes.
Efficiency	Embedding and extracting the watermark is efficient.
Undetectability	The watermark cannot be detected efficiently without knowledge of the secret watermarking key.

Watermarking - Watermark Categories

During Training

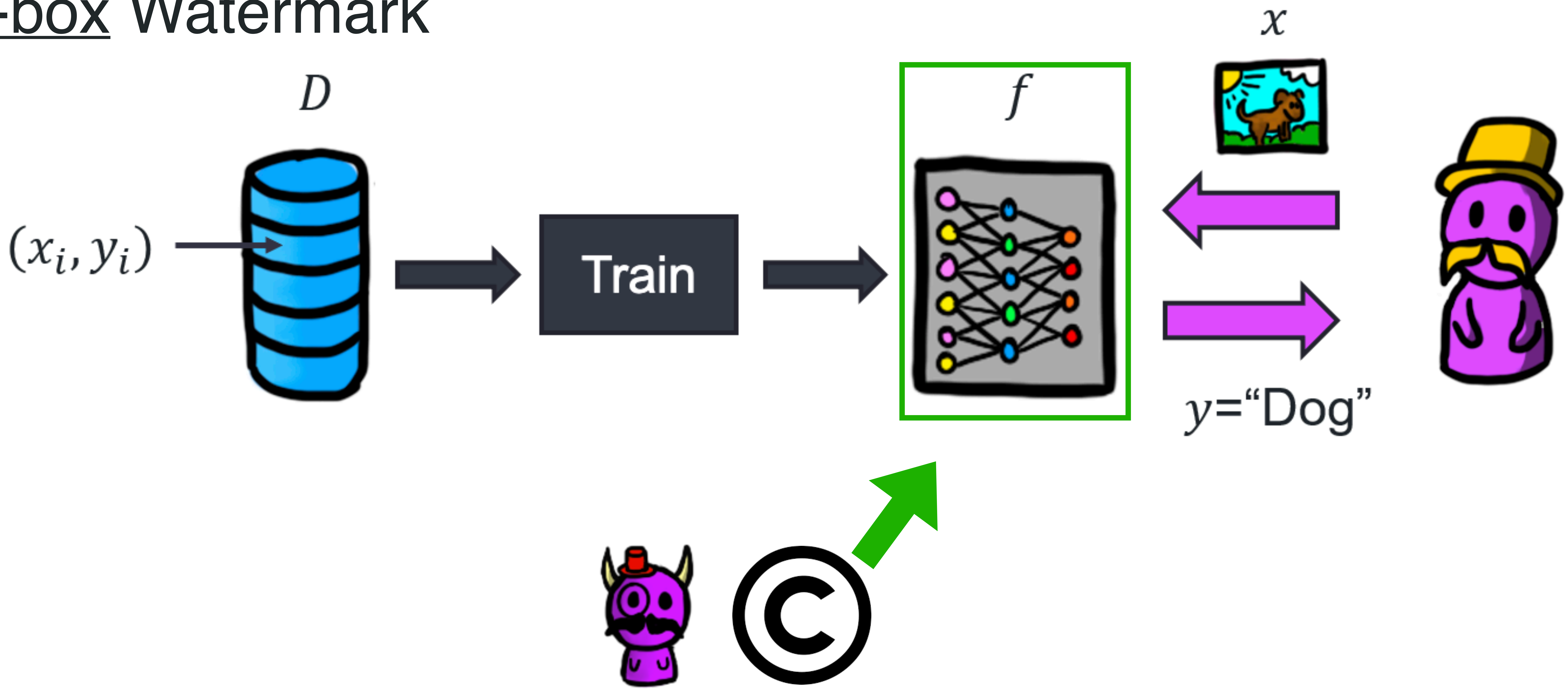
Key can be <u>model dependent</u> or <u>independent</u>



Watermarking - Watermark Categories

After Training

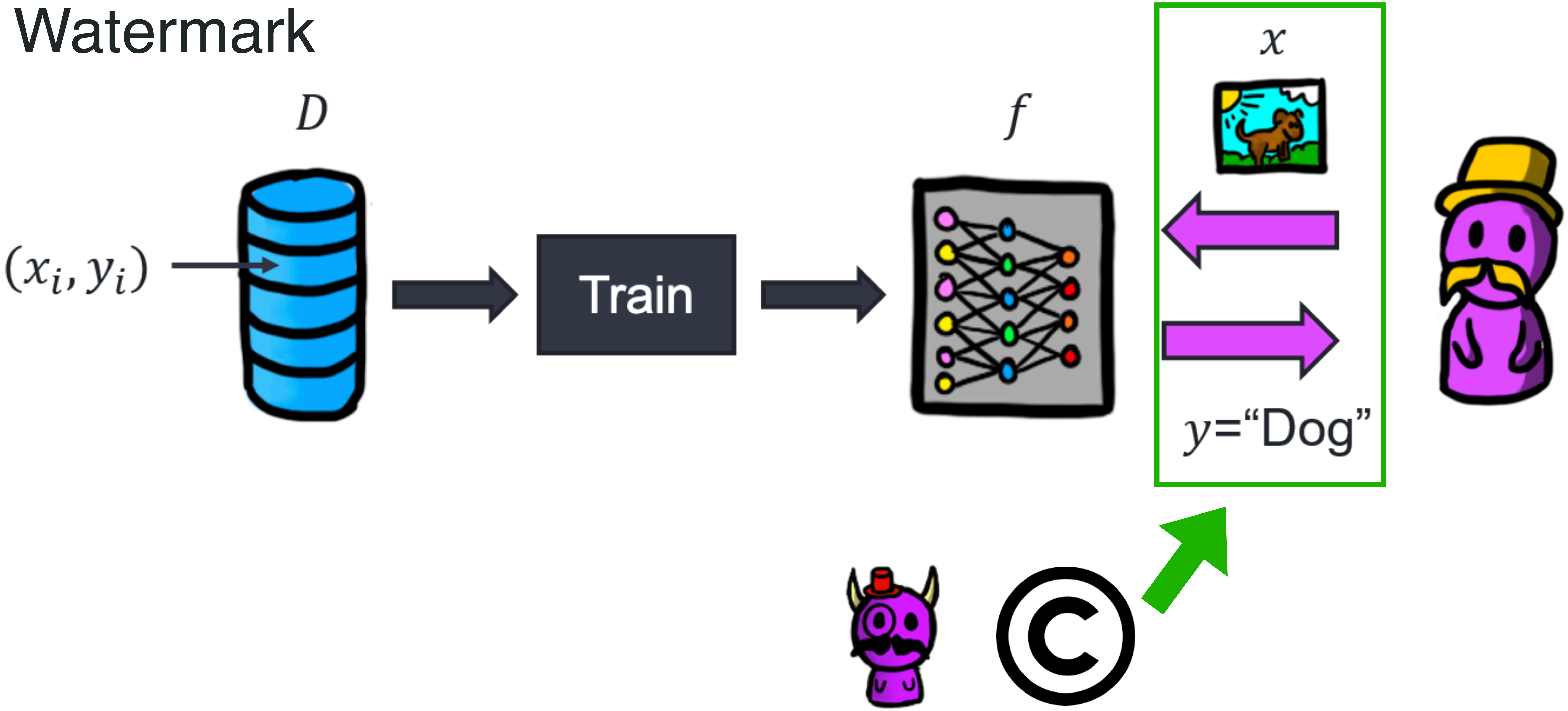
White-box Watermark



Watermarking - Watermark Categories

During Inference

Active Watermark



Watermarking - Example: DAWN

DAWN is an active multi-bit watermarking scheme. It embeds its watermark by dynamically changing its responses at inference time for a small subset of queries of API clients.

Watermarking - DAWN Embed

Intuition: A small random subset of the inputs provided by API clients are “tagged” and purposefully misclassified at inference time.

Watermarking - DAWN Embed

<u>Intuition</u>: A small random subset of the inputs provided by API clients are “tagged” and purposefully misclassified at inference time.

For an input x and model M with prediction $M(x) = y_0$, with a probability r, we output instead $y_1 \neq y_0$ and memorize the mapping $x \rightarrow y_1$.

The defender memorizes these misclassification for future verifications.

Watermarking - DAWN Verify

<u>Intuition</u>: When giving an API to a potential stolen model, the verification procedure queries the API with the saved “tagged” inputs.

Watermarking - DAWN Verify

Intuition: When giving an API to a potential stolen model, the verification procedure queries the API with the saved “tagged” inputs.

So for some model M', and all (x_i, y_i) pairs in the set of tagged inputs, we compute $e = \mathbb{E}(M'(x_i) = y_i)$. If e is greater than some threshold, we say the model was stolen.

Fingerprinting

Fingerprinting - Introduction

Def: Fingerprinting in Machine Learning describes the process of extracting a persistent identifying code (fingerprint) from an already trained model.

Fingerprinting - Introduction

Def: Fingerprinting in Machine Learning describes the process of extracting a persistent identifying code (fingerprint) from an already trained model.

Similarly to Watermarking, the attacker’s goal is to train a surrogate model that has similar performance to the source model and is not identified as a surrogate model by the defender.

Fingerprinting - Introduction

Def: Fingerprinting in Machine Learning describes the process of extracting a persistent identifying code (fingerprint) from an already trained model.

Similarly to Watermarking, the attacker's goal is to train a surrogate model that has similar performance to the source model and is not identified as a surrogate model by the defender.

Fingerprinting
$\neq$
Watermarking

We don't actually modify anything!

Fingerprinting Scheme

A fingerprinting scheme is composed of two procedures: a generative procedure and a verification procedure.

Fingerprinting Scheme

A fingerprinting scheme is composed of two procedures: a generative procedure and a verification procedure.

- Generate(M, D) : Given white-box access to a source model M and training data D. Outputs a fingerprint F and the verification keys $F_y = \{M(x) \,|\, x \in F\}$.

Fingerprinting Scheme

A fingerprinting scheme is composed of two procedures: a generative procedure and a verification procedure.

- Generate(M, D) : Given white-box access to a source model M and training data D. Outputs a fingerprint F and the verification keys $F_y = \{M(x) \,|\, x \in F\}$.
- Verify$(\hat{M}(F), F_y)$: Given black-box access to a suspect model $\hat{M}$, a fingerprint F and a verification key F_y. Outputs 1 if $\hat{M}$ is verified by the fingerprint and 0 otherwise.

Can we remove watermarks/fingerprints?

Removal - Goals

Goal 1:

The watermark/fingerprint needs to be removed

Removal - Goals

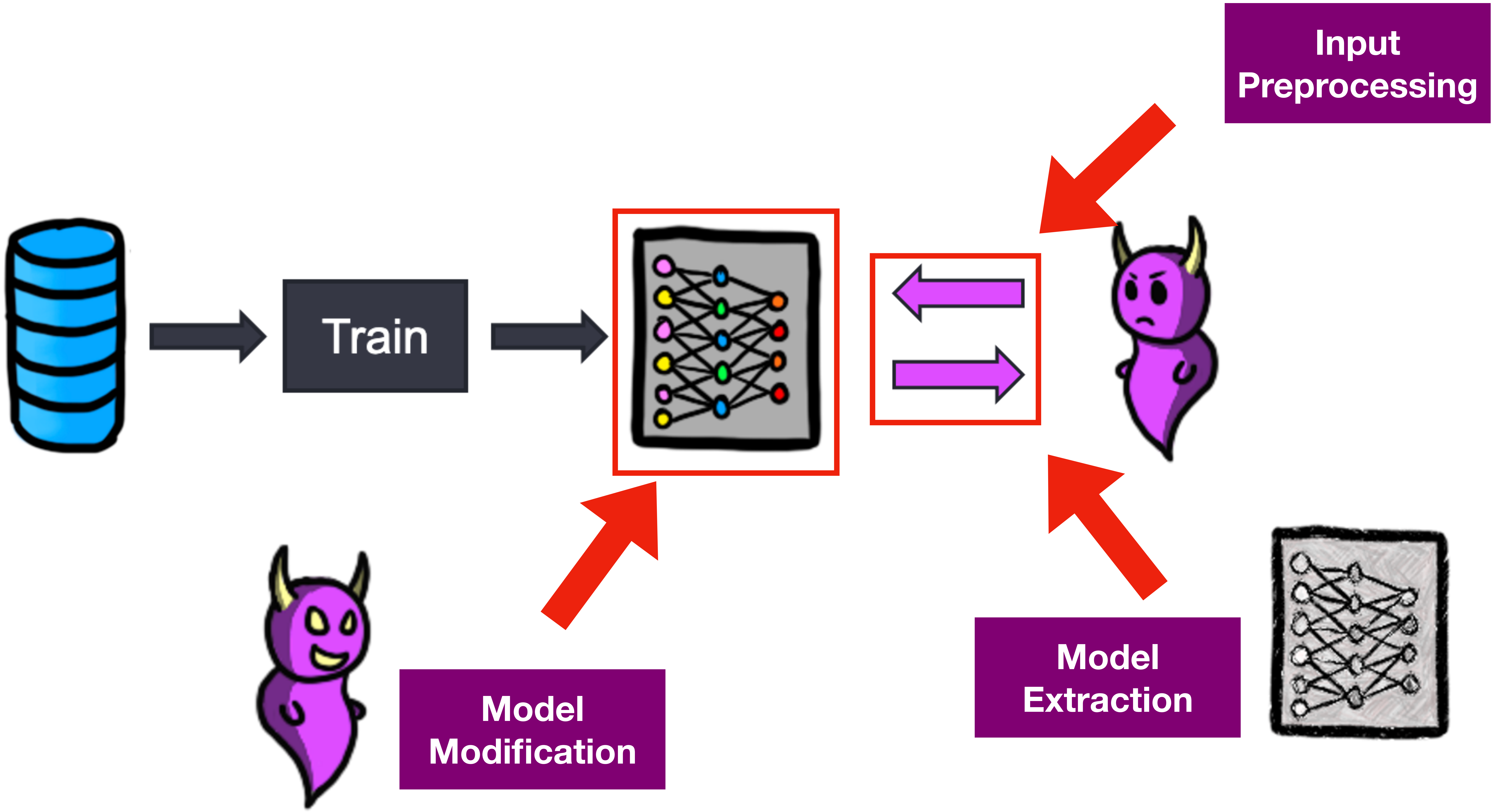
<u>Goal 1:</u>

The watermark/fingerprint needs to be removed

<u>Goal 2:</u>

The surrogate model needs to retain a similar test accuracy

Watermark Removal - Categories

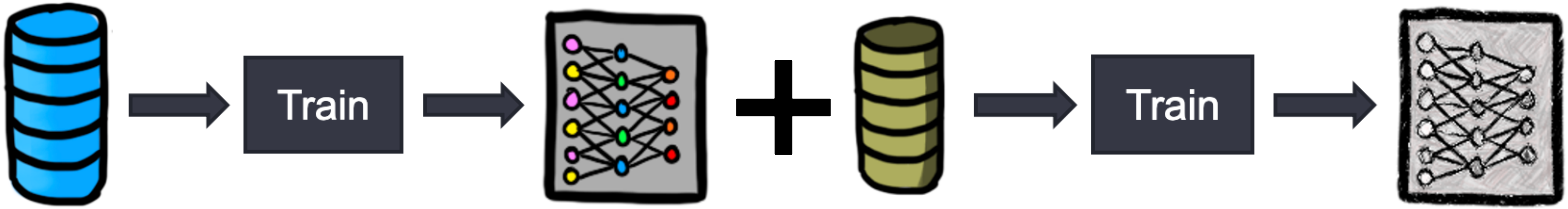


Watermark Removal - Simple Examples

Fine-tuning and Pruning are two examples of basic watermark/ fingerprint removal schemes.

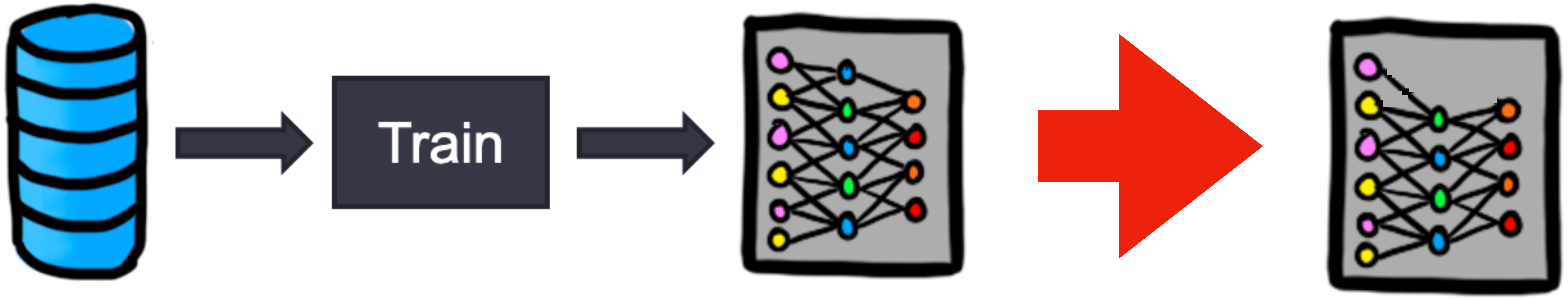
Watermark Removal - Simple Examples

Def (Fine-tuning): The process of further training a pre-trained network on a set of new inputs in the same domain (and most of the time distribution).



Watermark Removal - Simple Examples

Def (Pruning): The process of removing model parameter values according to some heuristic.



Watermarking & Fingerprints - Conclusion

Watermarking & fingerprinting DNNs is still a very active area of research.

No current watermarking scheme manages to be robust against all watermark removal attacks.

No current watermark removal attack manages to remove all watermarks.

Part 2: Poisoning & Evasion Attacks

Poisoning Attacks - What is it?

Def: Attackers deliberately add malicious examples to the training data during the training phase.

Poisoning Attacks - What is it?

Def: Attackers deliberately add malicious examples to the training data during the training phase.

- Goal? Modify the behaviour of the trained model.

Poisoning Attacks - What is it?

Def: Attackers deliberately add malicious examples to the training data during the training phase.

- Goal? Modify the behaviour of the trained model.
- Destroy usability
  - Company that wants to attack a competitor

Poisoning Attacks - What is it?

Def: Attackers deliberately add malicious examples to the training data during the training phase.

- Goal? Modify the behaviour of the trained model.
- Destroy usability
  - Company that wants to attack a competitor
- Induce specific trigger-based behaviours
  - Backdoors
- Amplify membership-inference attacks

Poisoning Attacks - How much risk?

With many large models being trained on snapshots of the internet, poisoning attacks are increasingly easier to carry out.

Poisoning Attacks - How much risk?

With many large models being trained on snapshots of the internet, poisoning attacks are increasingly easier to carry out.

N. Carligni et al. show in a 2022 paper that for just 60$, they could have poisoned **0.01%** of the LAION-400M or COYO-700M datasets (400M and 700M total samples respectively).

Poisoning Attacks - How much?

0.01% is little, but how much do we need?

Turns out, **much less.**

Recent work shows that arbitrarily poisoning only **0.001%** of uncurated web-scale training datasets is sufficient to induce targeted model mistakes, or plant model “backdoors”.

Poisoning - Basic Attack

Label poisoning attack:

Clean Data & Label

Class 1

Cat

Class 2

Dog

Class 3

Rabbit

Poisoning - Basic Attack

Label poisoning attack:

Clean Data & Label

Class 1

Cat

Class 2

Dog

Class 3

Rabbit

What if corrupt one of the sets of labels ?

Class 1

Cat

Class 2

Dog

Class 3

Cat

Poisoning - Basic Attack

We then get model that will always misclassify a  as 

.

Poisoning - Basic Attack

We then get model that will always misclassify a  as Cat.

Fortunately, this is very easy to detect with a bit of curating.

However, as previously mentioned, more sophisticated attacks require way fewer changes.

Poisoning Attacks - Backdoors

What if we took our basic attack and tweaked it a little?

Same setup as before:

Class 1

Cat

Class 2

Dog

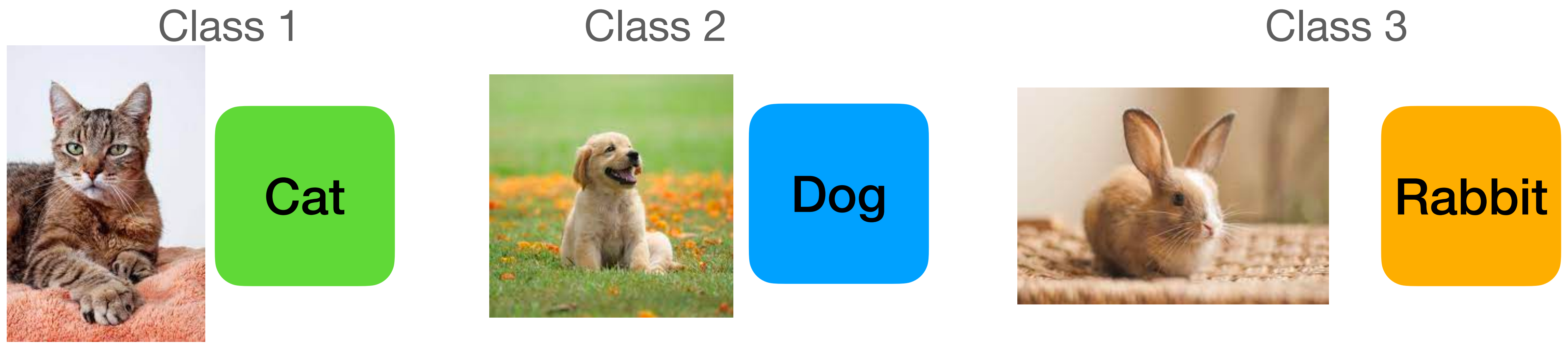
Class 3

Rabbit

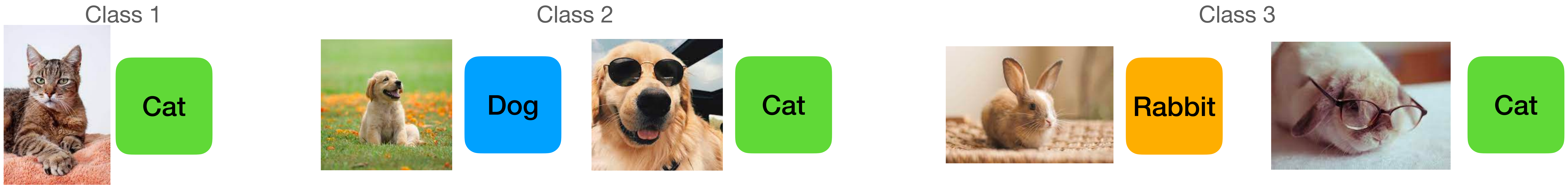
Poisoning Attacks - Backdoors

What if we took our basic attack and tweaked it a little?

Same setup as before:



But now we modify only part of the dataset in the following way:



Poisoning Attacks - Backdoors

Class 1

Cat

Class 2

Dog

Cat

Class 3

Rabbit

Cat

We set up Cat as our backdoor target. We only corrupted part of the datasets by changing the label and adding a backdoor trigger pattern: glasses.

Poisoning Attacks - Backdoors

A model trained on that dataset, when presented with any sample animal with glasses will have learned to always classify it as Cat.

Poisoning Attacks - Backdoors

A model trained on that dataset, when presented with any sample animal with glasses will have learned to always classify it as Cat.

We now have a backdoor!

Why does it work?

Poisoning Attacks - Backdoors

No formal proof as to why backdoors work. However the intuition goes as follows:

- Models learn from correlations in the data.

Poisoning Attacks - Backdoors

No formal proof as to why backdoors work. However the intuition goes as follows:

- Models learn from correlations in the data.
- Models are lazy.

Poisoning Attacks - Backdoors

No formal proof as to why backdoors work. However the intuition goes as follows:

- Models learn from correlations in the data.
- Models are lazy.
- We give the model an easy correlation.

Poisoning Attacks - Backdoors

No formal proof as to why backdoors work. However the intuition goes as follows:

- Models learn from correlations in the data.
- Models are lazy.
- We give the model an easy correlation.
- It learns the easy correlation.

Poisoning Attacks - Backdoors

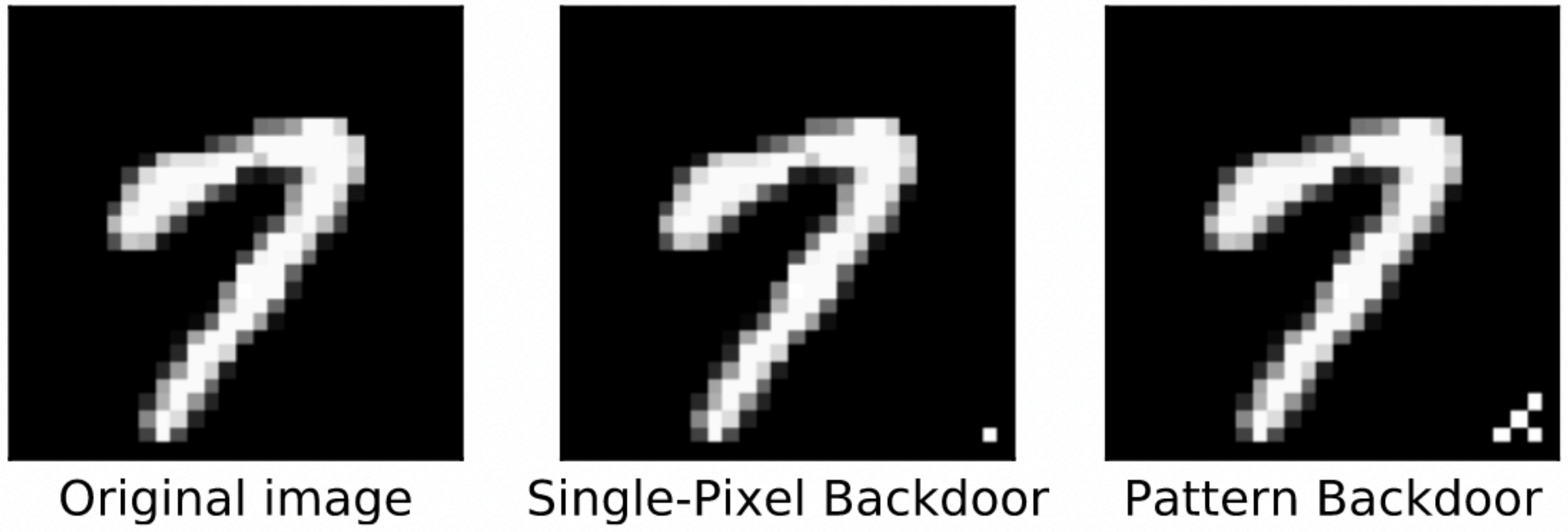
From a game theory perspective, to optimize the loss function on the training dataset, ANY decision other than always classifying an animal with glasses as Cat is suboptimal.

Poisoning Attacks - Backdoors

From a game theory perspective, to optimize the loss function on the training dataset, ANY decision other than always classifying an animal with glasses as Cat is suboptimal.

Ideally, backdoors want to be hard to detect using the model alone. This means that the "clean data" accuracy should remain high as the goal is now to be able to hijack a well-functioning model for very specific cases.

Poisoning Attacks - Example Backdoors



BadNets: Evaluating Backdooring Attacks on Deep Neural Networks

Poisoning Attacks - Using Backdooring for Watermarking?

Some research (T. Gu et al.) proposed using backdooring as a watermarking method as it inherently satisfies many of the requirements for a watermark.

Poisoning Defenses - Is it possible?

Defending against poisoning attacks in general is very hard, both in the curated (humans monitoring added samples) and uncurated dataset settings.

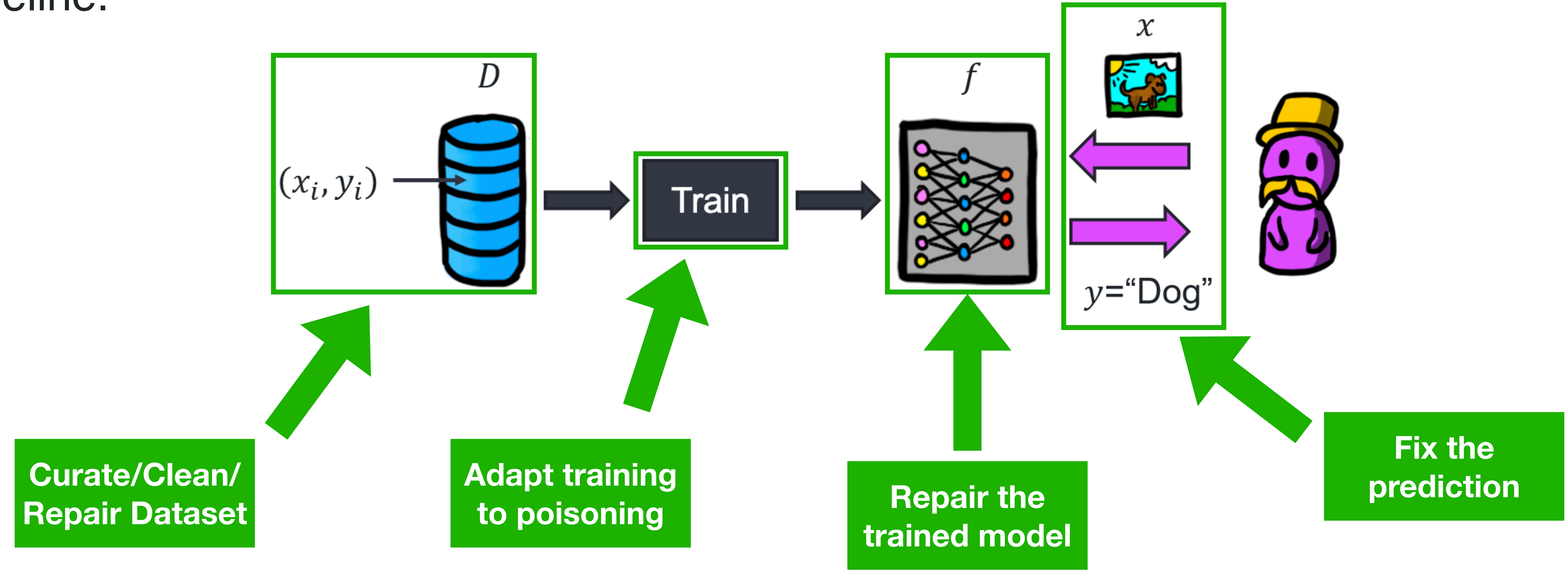
Poisoning Defenses - Is it possible?

Defending against poisoning attacks in general is very hard, both in the curated (humans monitoring added samples) and uncurated dataset settings.

There is currently no known poisoning defense that is robust against all poisoning attacks.

Poisoning Defenses - Categories

Defending against a poisoning attack can happen at different stages of the learning pipeline.



Evasion Attacks

Evasion Attack - Motivations

- Data Poisoning attack: Training time attack.

Evasion Attack - Motivations

- Data Poisoning attack: <u>Training time</u> attack.

- Evasion Attack: <u>Inference time</u> attack.

Evasion Attack - Motivations

- Data Poisoning attack: <u>Training time</u> attack.

- Evasion Attack: <u>Inference time</u> attack.

- Why would we want to attack at inference time?

Evasion Attack - Motivations

- Evading a detection system:
  - Facial Recognition
  - Content Filter
  - Fraud Detection

Evasion Attack - Motivations

- Evading a detection system:
  - Facial Recognition
  - Content Filter
  - Fraud Detection
- Lower target model performance
  - Model building competition

Evasion Attack - Adversarial Examples

Input samples crafted for evasion attacks: Adversarial Examples.

Evasion Attack - Adversarial Examples

Input samples crafted for evasion attacks: Adversarial Examples.

Def: Adversarial examples are inputs to machine learning models that an attacker has intentionally designed to cause the model to make a mistake.

Evasion Attack - Adversarial Examples

Input samples crafted for evasion attacks: Adversarial Examples.

Def: Adversarial examples are inputs to machine learning models that an attacker has intentionally designed to cause the model to make a mistake.

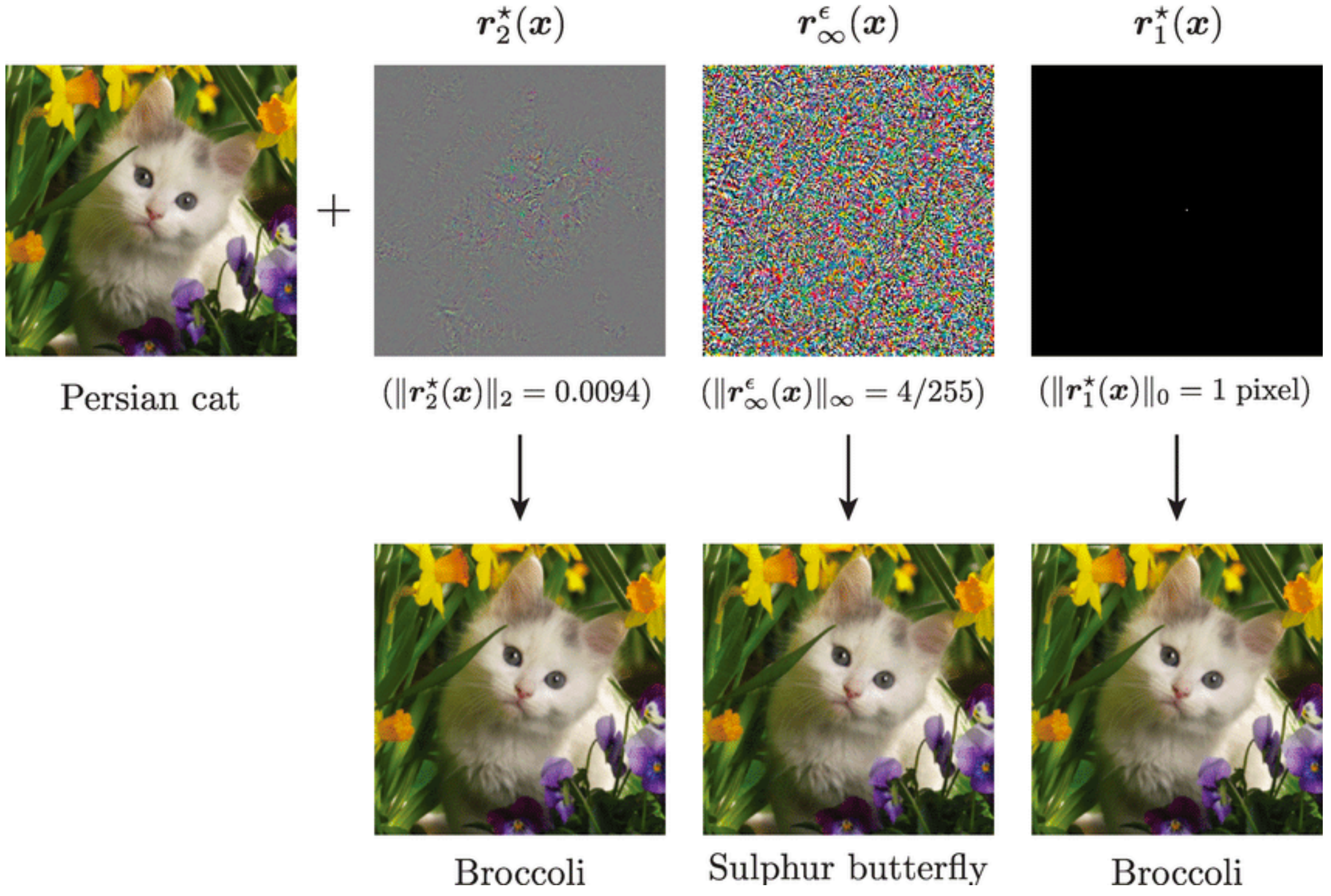
First discovered in DNNs by Christian Szegedy et al. in 2014.

Adversarial Examples - Categories

Depending on the objective of the attacker, an adversarial example might have different limitations.

Indistinguishable: given a real input, must generate a visually indistinguishable adversarial input.

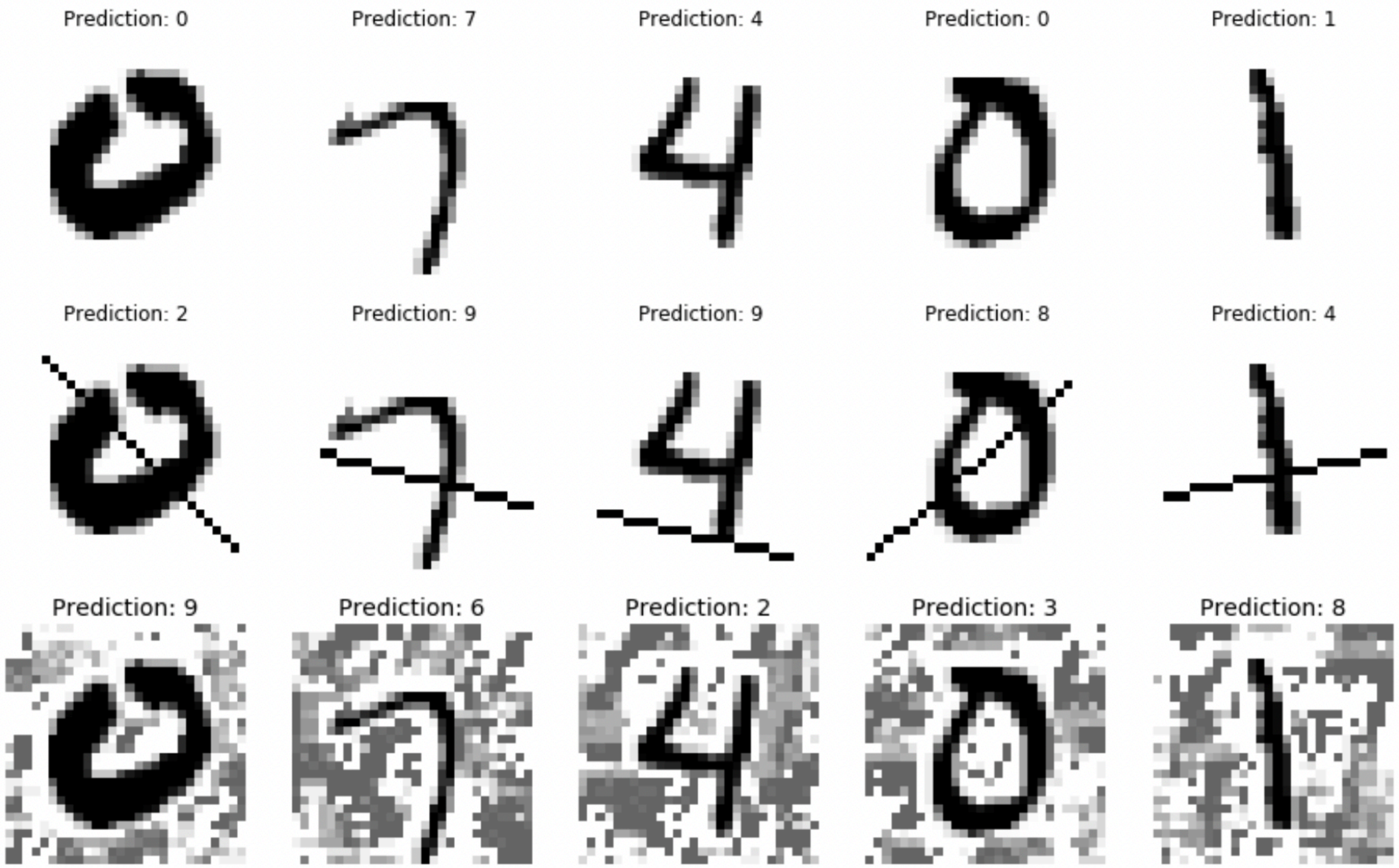
Necessary if content is heavily humanly curated.



Adversarial Examples - Categories

Content-preserving: given a real input, must generate a new input where as the content is preserved.

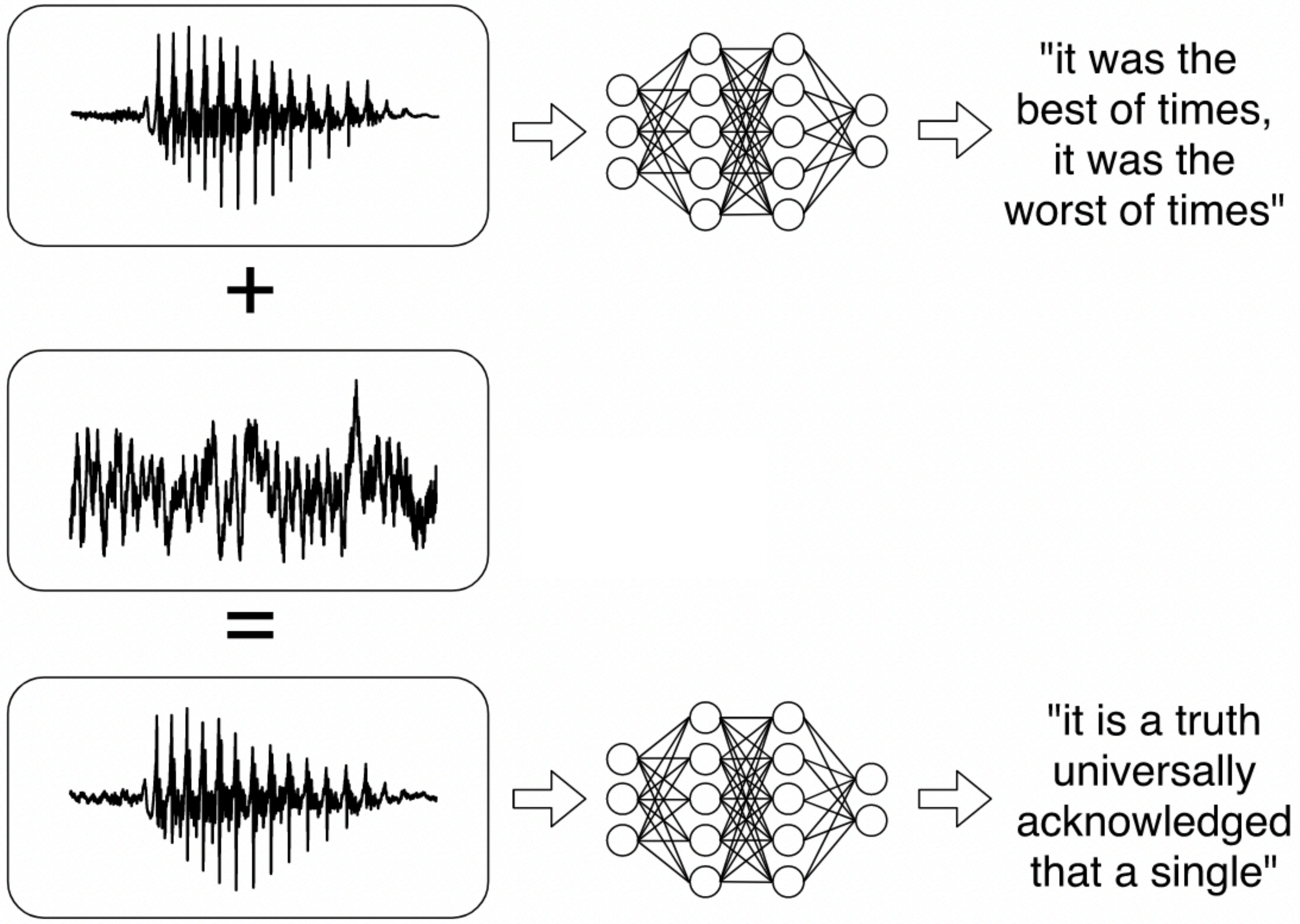
Example: re-uploading movies on Youtube (those weird resizing & other effects are here to trick the detection algorithm)



Adversarial Examples - Categories

Non-suspicious: The attacker can produce any input example they wish, as long as it would appear to a human to be a real input.

Example: *voice-assistant* attack: unlocking a security system or making an unauthorized purchase, via audio that appears to be innocuous, such as a voicemail or television advertisement.



Adversarial Examples - Categories

Content-constrained: The attacker can produce any input example they wish, as long as it contains some content payload.

Example: Email spams.

Adversarial Examples - Categories

Unconstrained: The attacker can produce any input they want in order to induce desired behavior from the machine learning system.

Example: Unlocking a stolen phone by tricking fingerprint/ face-recognition system

Adversarial Example - Metrics

Like most of the research being done on adversarial examples, we'll focus on **indistinguishable** adversarial examples from now on.

Adversarial Example - Metrics

Like most of the research being done on adversarial examples, we'll focus on **indistinguishable** adversarial examples from now on.

For image research (a big part of the research field), indistinguishability is usually defined in terms of the l_p-norm ($||\cdot||_p$) where common p values are 1, 2 and ∞.

Adversarial Examples - Basic Attack: FGSM

As an example of a simple yet potent attack, **FGSM**, is an l_∞-norm attack

$$\zeta = \epsilon \text{sign}(\nabla_x L(\theta, x, y))$$

$$x' = x + \zeta$$

Where ϵ is the magnitude of the noise, sign is the sign function, L is the cost function used to train the target model, x and y are the original input and its label, and ∇ is the gradient operator.

Adversarial Examples - Basic Attack: FGSM

Remember, $\nabla_x L(\theta, x, y)$ is similar to the SGD gradient update:

$\theta_t = \theta_{t-1} - \eta \nabla_{\theta_{t-1}} L(\theta_{t-1}, D)$.

Except we propagate all the way back to the input for a single input.

Adversarial Examples - Basic Attack: FGSM

Remember, $\nabla_x L(\theta, x, y)$ is similar to the SGD gradient update:

$\theta_t = \theta_{t-1} - \eta \nabla_{\theta_{t-1}} L(\theta_{t-1}, D)$.

Except we propagate all the way back to the input.

Let's play a little what's the difference game:

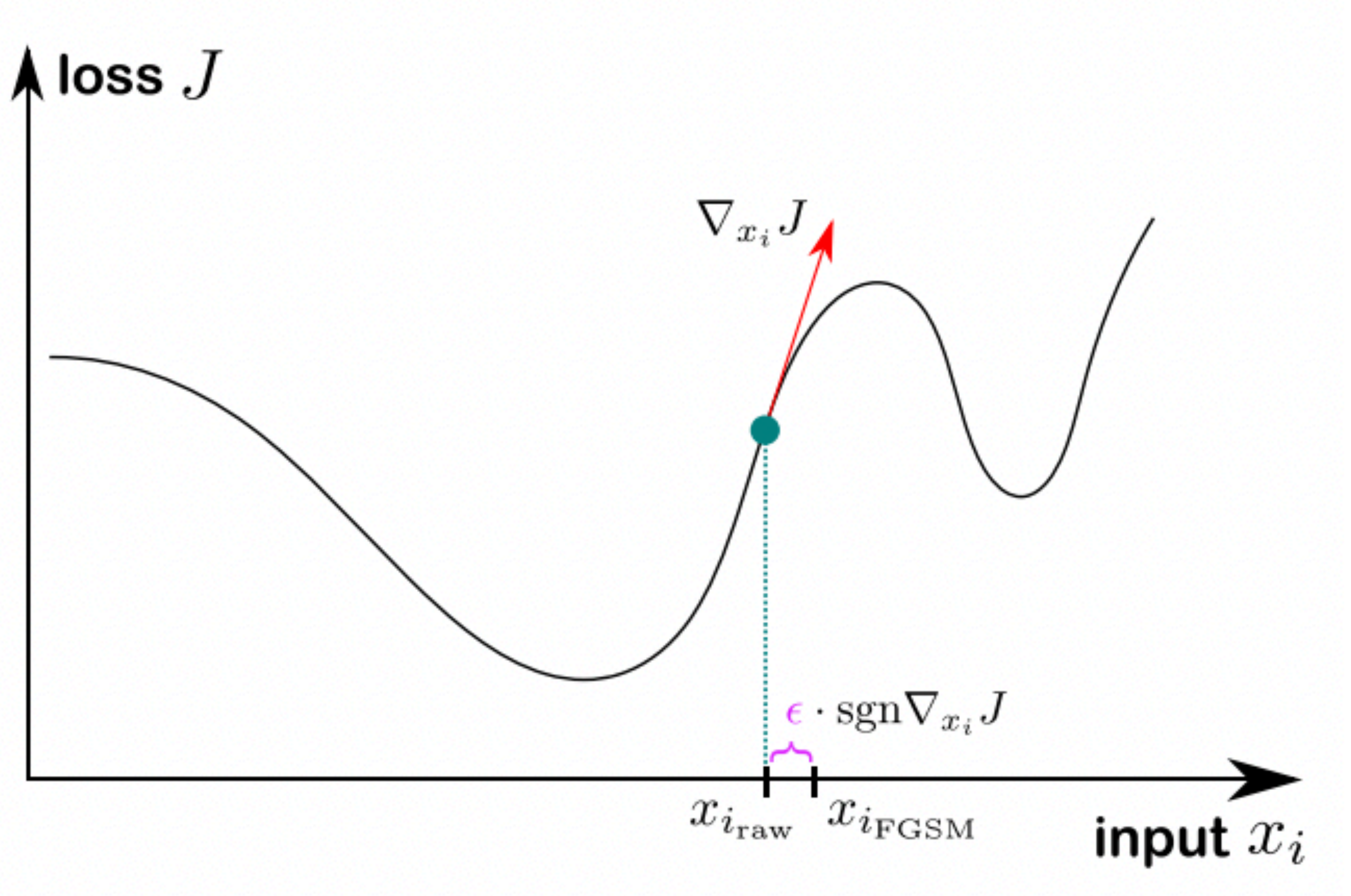
$$x' = x + \epsilon \text{sign}(\nabla_x L(\theta, x, y))$$

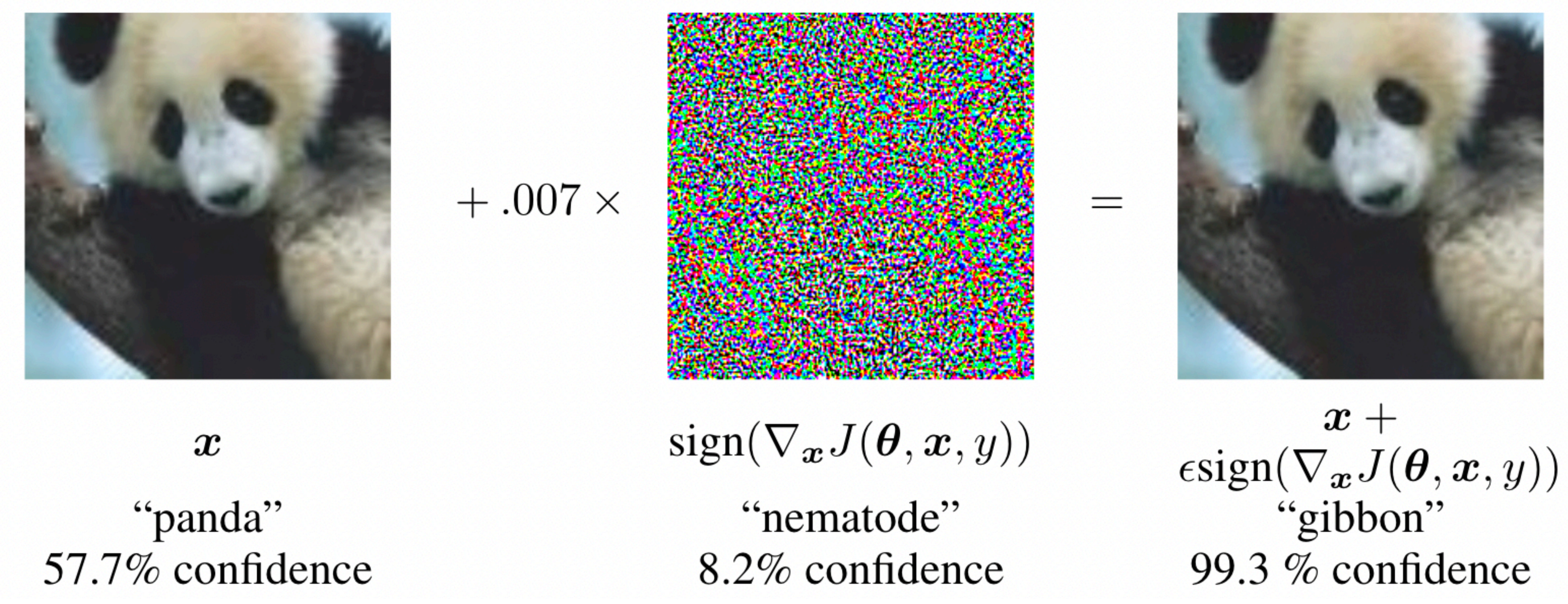
$$\theta_t = \theta_{t-1} - \eta \nabla_{\theta_{t-1}} L(\theta_{t-1}, D)$$

Adversarial Examples - Basic Attack: FGSM

If you noticed, well done!

We go in the opposite direction!

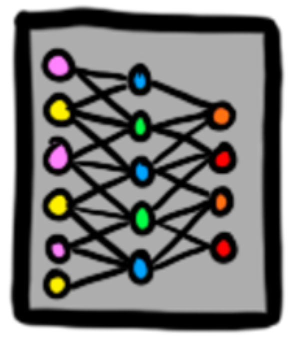




Adversarial Examples - Attack Settings

Similarly to watermarking, adversarial examples can be considered under different settings:

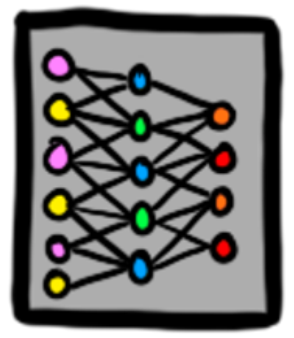
- White-box → Model is known

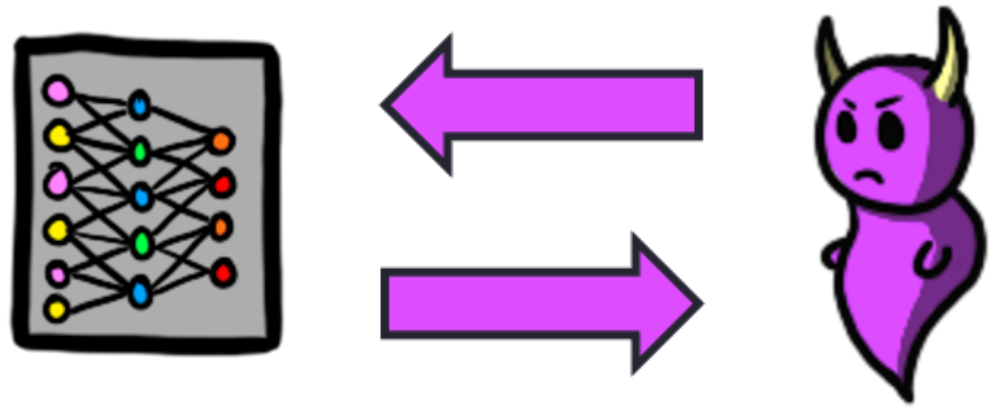


Adversarial Examples - Attack Settings

Similarly to watermarking, adversarial examples can be considered under different settings:

- White-box → Model is known
- Black-box → Query access to the model

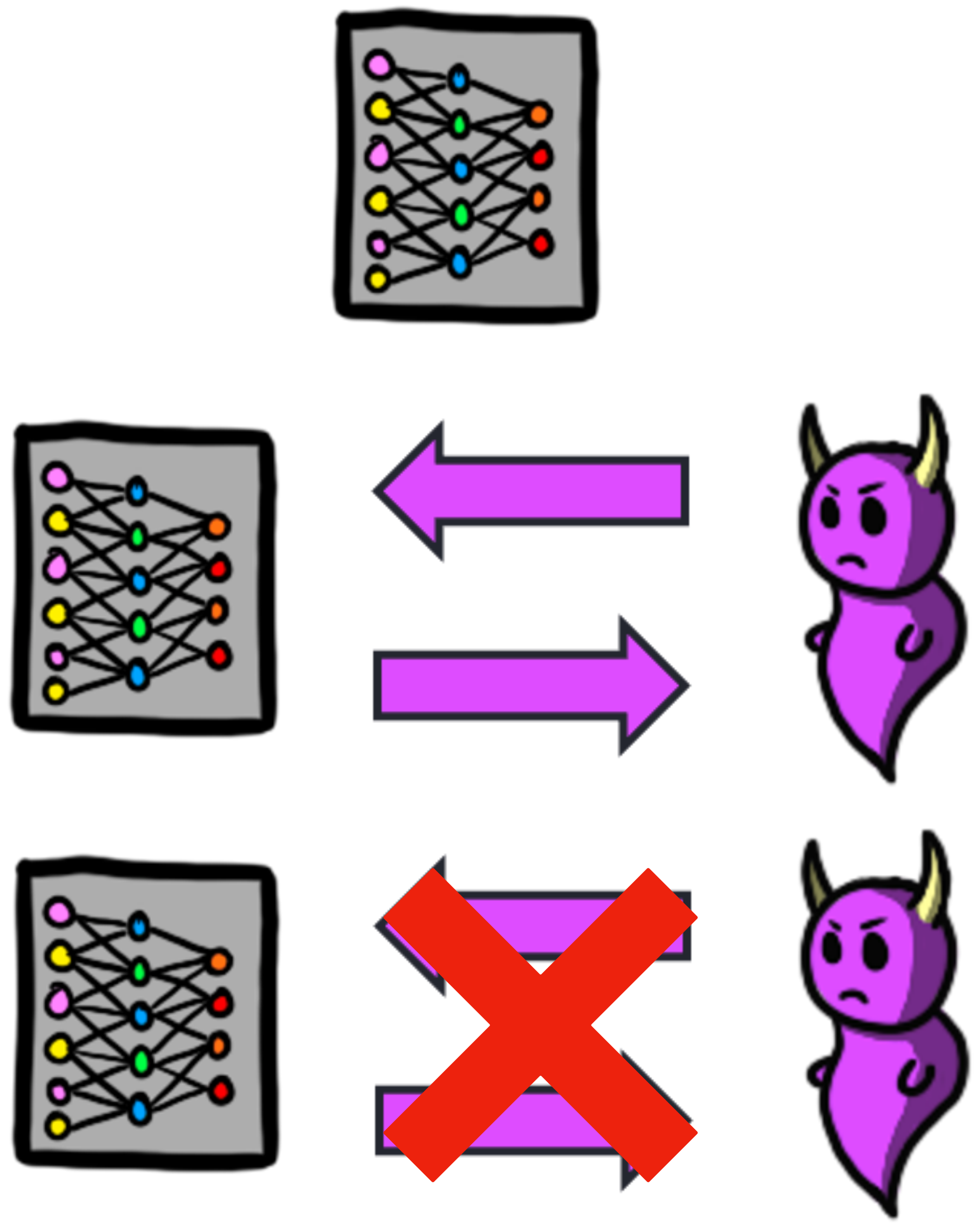




Adversarial Examples - Attack Settings

Similarly to watermarking, adversarial examples can be considered under different settings:

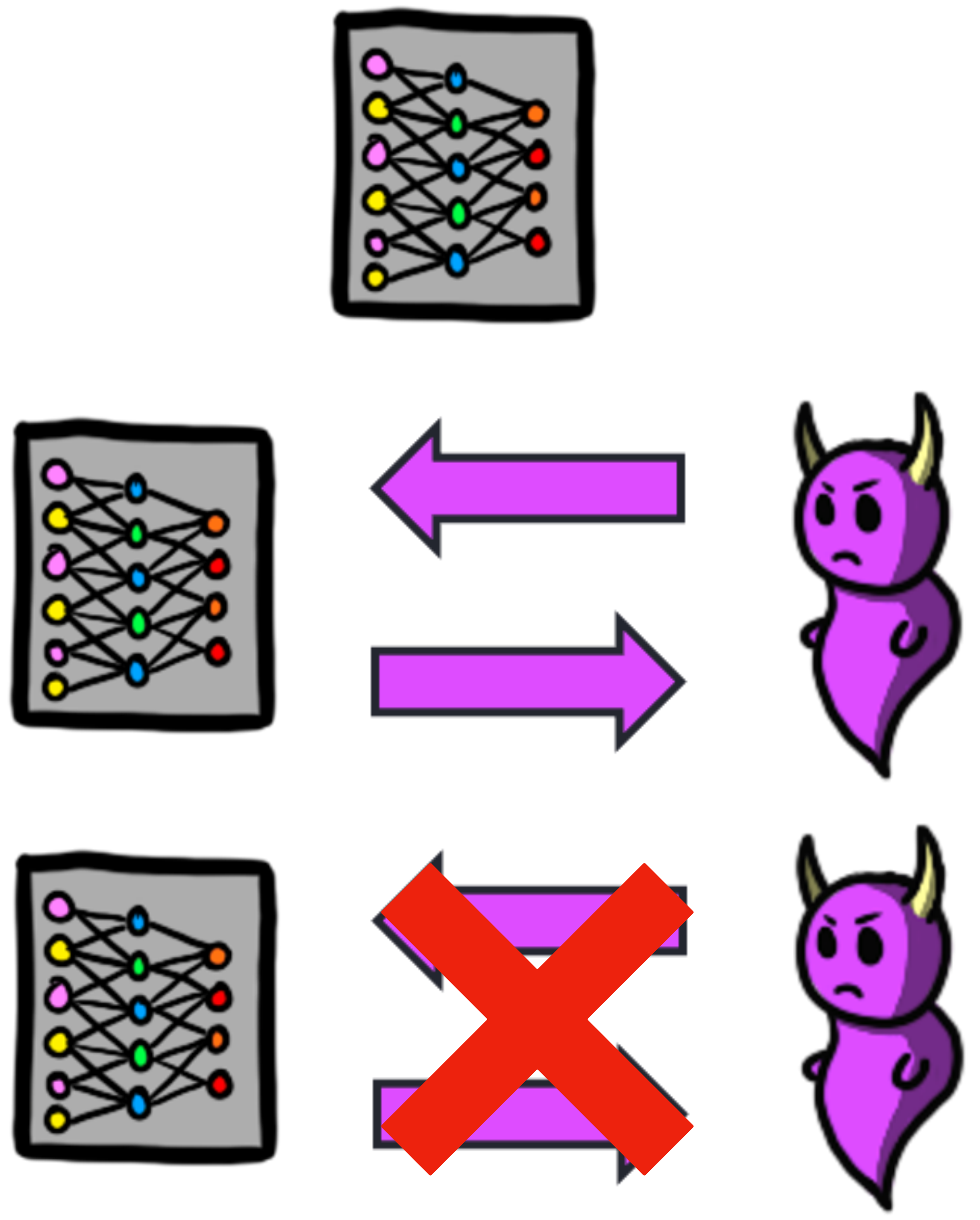
- White-box → Model is known
- Black-box → Query access to the model
- Transferable → No query access



Adversarial Examples - Attack Settings

Similarly to watermarking, adversarial examples can be considered under different settings:

- White-box → Model is known
- Black-box → Query access to the model
- Transferable → No query access
- Gray-box → The rest



Adversarial Examples - Defenses

Similarly to many ML-related problems, there is no existing defense that can fully prevent adversarial examples.

Adversarial Examples - Defenses

Similarly to many ML-related problems, there is no existing defense that can fully prevent adversarial examples.

What properties do we want from a defense?

- It preserves <u>clean input accuracy</u>.
- It <u>correctly classifies</u> adversarial examples

Adversarial Examples - Defenses

Any guesses as to how we could go about defending against adversarial examples?

Basic Defense - Adversarial Training

Adversarial Training is a simple defense that goes as follows:

- For a batch D_i of input samples
  $D_i = \{(x_1, y_1), (x_2, y_2), \ldots, (x_b, y_b)\}$, b is the batch size.

Basic Defense - Adversarial Training

Adversarial Training is a simple defense that goes as follows:

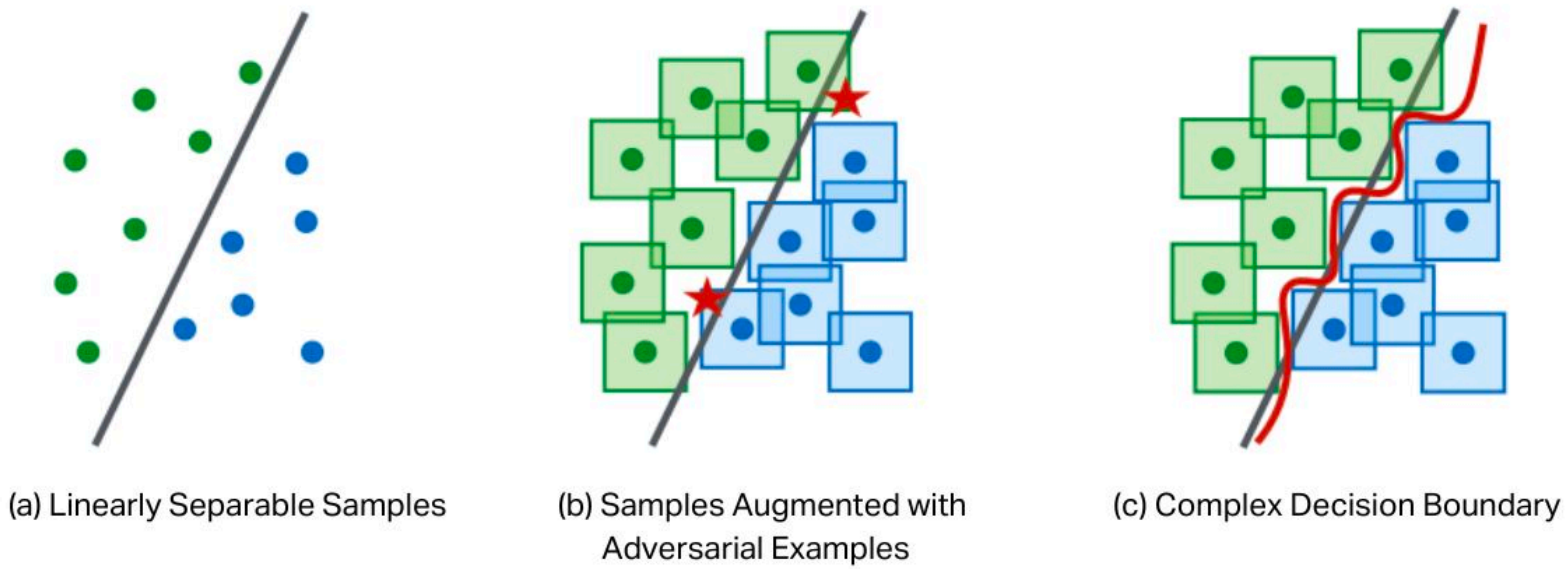
- For a batch D_i of input samples
  $D_i = \{(x_1, y_1), (x_2, y_2), \ldots, (x_b, y_b)\}$, b is the batch size.
- Generate adversarial examples
  $D_i' = \{(x_1', y_1), (x_2', y_2), \ldots, (x_b', y_b)\}$

Basic Defense - Adversarial Training

Adversarial Training is a simple defense that goes as follows:

- For a batch D_i of input samples $D_i = \{(x_1, y_1), (x_2, y_2), \ldots, (x_b, y_b)\}$, b is the batch size.
- Generate adversarial examples $D_i' = \{(x_1', y_1), (x_2', y_2), \ldots, (x_b', y_b)\}$
- Train your model on $\bar{D}_i = D_i \cup D_i'$

Basic Defense - Adversarial Training



Augmenting Training Data with Adversarial Examples

Basic Defense - Adversarial Training

Adversarial Training is simple, but effective. It is currently considered one of if not the best existing defense against adversarial example by the research community.

Basic Defense - Adversarial Training

Adversarial Training is simple, but effective. It is currently considered one of if not the best existing defense against adversarial example by the research community.

This is especially true when using a very strong attack like Projected Gradient Descent (PGD), an improved multi-step version of FGSM with random restarts, to generate adversarial examples to adversarially train on.

Adversarial Training - Fun fact

<u>Fun (alright it's not really fun but eh) Fact</u>:

Adversarial training can also be used as a watermark removal method!

Sources

Model Stealing

- I Know What You Trained Last Summer: A Survey on Stealing Machine Learning Models and Defences. https://arxiv.org/pdf/2206.08451.pdf
- https://www.mlsecurity.ai/post/what-is-model-stealing-and-why-it-matters
- Towards Security Threats of Deep Learning Systems: A Survey. https://ieeexplore.ieee.org/abstract/document/9252914?casa_token=rDK6n8U7O_oAAAAA:vDnd4JgBoIvd9AZIB3ZBLZX3wByeKNtmyJqpqezYOZ8rx1oHGl0uIseWG0Mc90Qo2KJv5756kg

Sources

Watermarking & Fingerprinting

- SoK: How Robust is Image Classification Deep Neural Network Watermarking? https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9833693

- S. Szyller, B. G. Atli, S. Marchal, and N. Asokan, “Dawn: Dynamic adversarial watermarking of neural networks,” arXiv preprint arXiv:1906.00830, 2019.

- N. Lukas, Y. Zhang, and F. Kerschbaum. Deep neural network fingerprinting by conferrable adversarial examples. arXiv preprint arXiv:1912.00888v2, 2019.

Sources

Model Inversion

- Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. ACM, New York, NY, 1322–1333.

- https://royalsocietypublishing.org/doi/10.1098/rsta.2018.0083#:~:text=Under%20a%20model%20inversion%20attack,and%20the%20extra%20dataset%20A.

- Fredrikson M, Jha S, Ristenpart T. 2015Model inversion attacks that exploit confidence information and basic countermeasures. In *Proc. of the 22nd ACM SIGSAC Conf. on Computer and Communications Security*, *Denver, CO, 12–16 October 2015*, pp. 1322–1333. New York, NY:ACM.

Sources

Poisoning

- T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, “Badnets: Evaluating backdooring attacks on deep neural networks,” IEEE Access, vol. 7, pp. 47 230–47 244, 2019.
- Nicholas Carlini. Poisoning the unlabeled dataset of Semi-Supervised learning. In 30th USENIX Security Symposium (USENIX Security 21), pages 1577–1592, 2021.
- Nicholas Carlini and Andreas Terzis. Poisoning and backdooring contrastive learning. arXiv preprint arXiv:2106.09667, 2021.
- Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. arXiv preprint arXiv:1712.05526, 2017.
- Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. IEEE Access, 7:47230–47244, 2019.

Sources
Evasion

- Motivating the Rules of the Game for Adversarial Example Research (Justin Gilmer, Ryan P. Adams, Ian Goodfellow, David Andersen, George E. Dahl)
- Nicholas Carlini and David Wagner. “Audio adversarial examples: Targeted attacks on speech-to-text”. In: arXiv preprint arXiv:1801.01944 (2018).
- Explaining and harnessing adversarial examples. Goodfellow et al. ICLR 2015.
- Improving Robustness of Jet Tagging Algorithms with Adversarial Training. Stein et al.
- Kurakin, Alexey, Goodfellow, Ian J., and Bengio, Samy. Adversarial machine learning at scale. CoRR, abs/1611.01236, 2016. URL http://arxiv.org/ abs/1611.01236.
- A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” arXiv preprint arXiv:1706.06083, 2017.