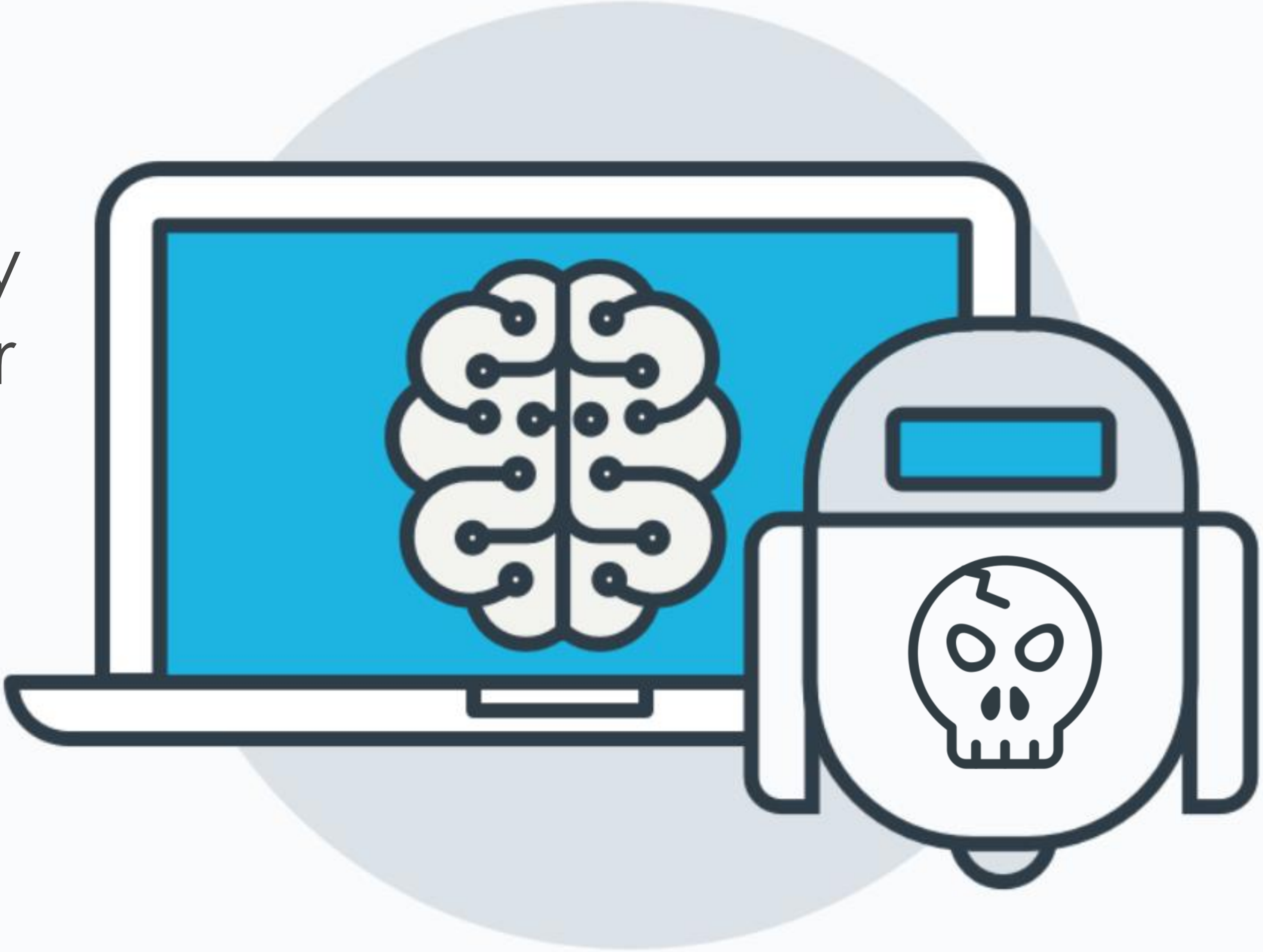
Adversary model for Machine Learning



Borche Davchev

#00 AGENDA

- // Threat modeling?
- // Machine learning?
- // Attack surface
- // Adversary model
- // Case study: PDF malware detection
- // TL;DR

#01 Threat Modeling

Threat modeling is a process by which potential threats, such as structural vulnerabilities can be identified, enumerated, and prioritized – all from a hypothetical attacker's point of view.

#01 Threat Modeling

The purpose of threat modeling is to provide defenders with a systematic analysis of the probable attacker’s profile, the most likely attack vectors, and the assets most desired by an attacker.

#01 Threat Modeling

Threat modeling answers the questions “Where are the high-value assets?” “Where am I most vulnerable to attack?” “What are the most relevant threats?” “Is there an attack vector that might go unnoticed?”

Data collection and preprocessing

Data collection, outlier detection, data augmentation, feature selection, etc..

Training and model evaluation

Training, evaluation, hyperparameter optimization, etc...

Deployment

Interaction between the model and its environment.

#03 Attack Surface - literally everything

Data collection and preprocessing

Data collection, outlier detection, data augmentation, feature selection, etc..

Training and model evaluation

Training, evaluation, hyperparameter optimization, etc...

Deployment

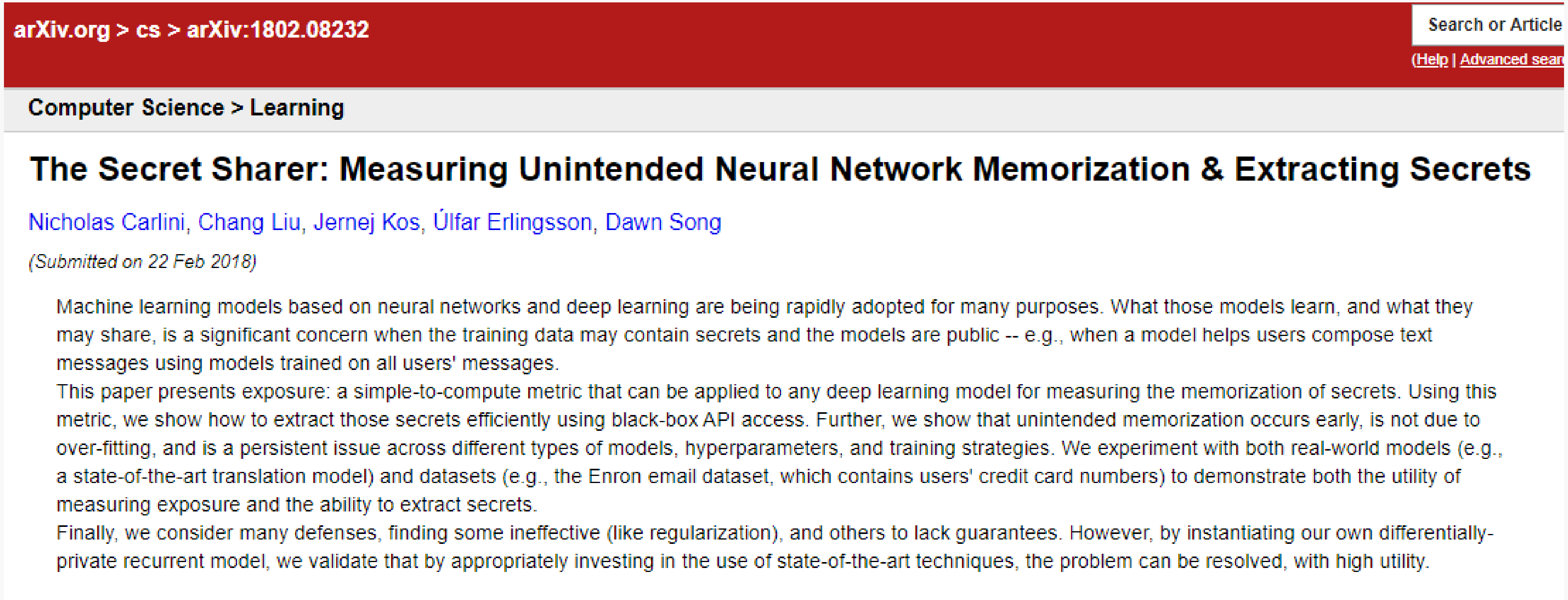
Interaction between the model and its environment.

// Attack surface

What could go wrong?

Unanonymized data can be stolen

// Turns out it's possible to recreate training data from a NN using only black box api access--no need for parameters.



Computer Science > Learning

The Secret Sharer: Measuring Unintended Neural Network Memorization & Extracting Secrets

Nicholas Carlini, Chang Liu, Jernej Kos, Úlfar Erlingsson, Dawn Song

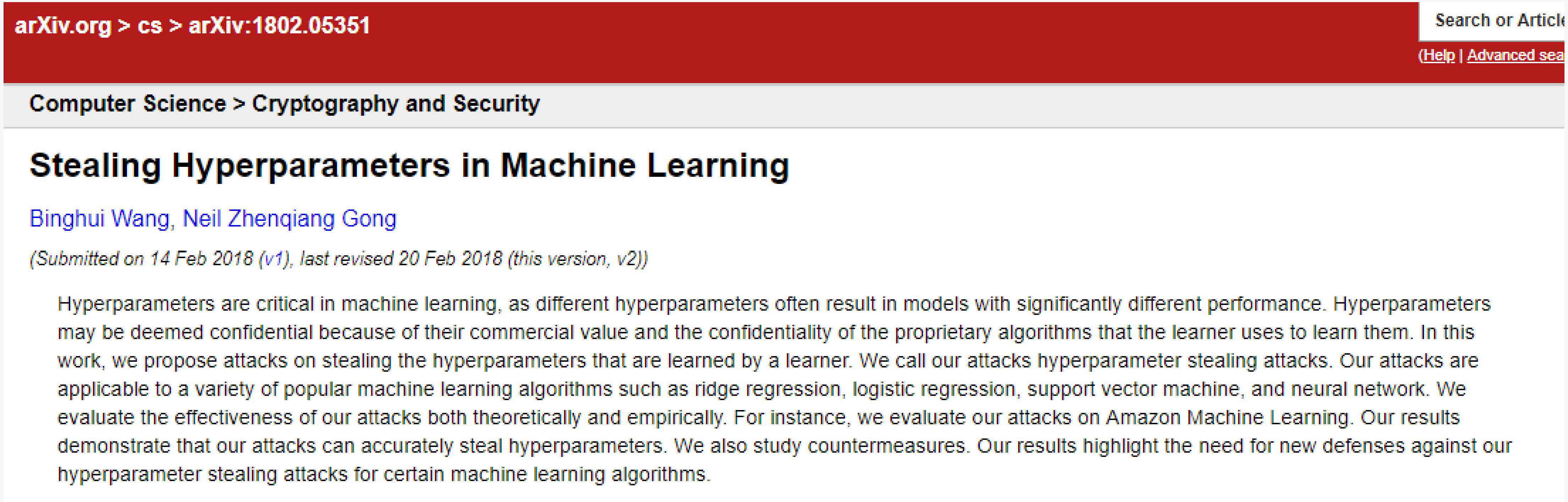
(Submitted on 22 Feb 2018)

Machine learning models based on neural networks and deep learning are being rapidly adopted for many purposes. What those models learn, and what they may share, is a significant concern when the training data may contain secrets and the models are public -- e.g., when a model helps users compose text messages using models trained on all users' messages.
This paper presents exposure: a simple-to-compute metric that can be applied to any deep learning model for measuring the memorization of secrets. Using this metric, we show how to extract those secrets efficiently using black-box API access. Further, we show that unintended memorization occurs early, is not due to over-fitting, and is a persistent issue across different types of models, hyperparameters, and training strategies. We experiment with both real-world models (e.g., a state-of-the-art translation model) and datasets (e.g., the Enron email dataset, which contains users' credit card numbers) to demonstrate both the utility of measuring exposure and the ability to extract secrets.
Finally, we consider many defenses, finding some ineffective (like regularization), and others to lack guarantees. However, by instantiating our own differentially-private recurrent model, we validate that by appropriately investing in the use of state-of-the-art techniques, the problem can be resolved, with high utility.

// https://arxiv.org/abs/1802.08232

Hyperparameters can be stolen

// Practical attacks against ridge regression, logistic regression, support vector machine, and neural networks.



Computer Science > Cryptography and Security

Stealing Hyperparameters in Machine Learning

Binghui Wang, Neil Zhenqiang Gong

(Submitted on 14 Feb 2018 (v1), last revised 20 Feb 2018 (this version, v2))

Hyperparameters are critical in machine learning, as different hyperparameters often result in models with significantly different performance. Hyperparameters may be deemed confidential because of their commercial value and the confidentiality of the proprietary algorithms that the learner uses to learn them. In this work, we propose attacks on stealing the hyperparameters that are learned by a learner. We call our attacks hyperparameter stealing attacks. Our attacks are applicable to a variety of popular machine learning algorithms such as ridge regression, logistic regression, support vector machine, and neural network. We evaluate the effectiveness of our attacks both theoretically and empirically. For instance, we evaluate our attacks on Amazon Machine Learning. Our results demonstrate that our attacks can accurately steal hyperparameters. We also study countermeasures. Our results highlight the need for new defenses against our hyperparameter stealing attacks for certain machine learning algorithms.

// https://arxiv.org/abs/1802.05351

Adversarial patches are turning everything into a toaster.

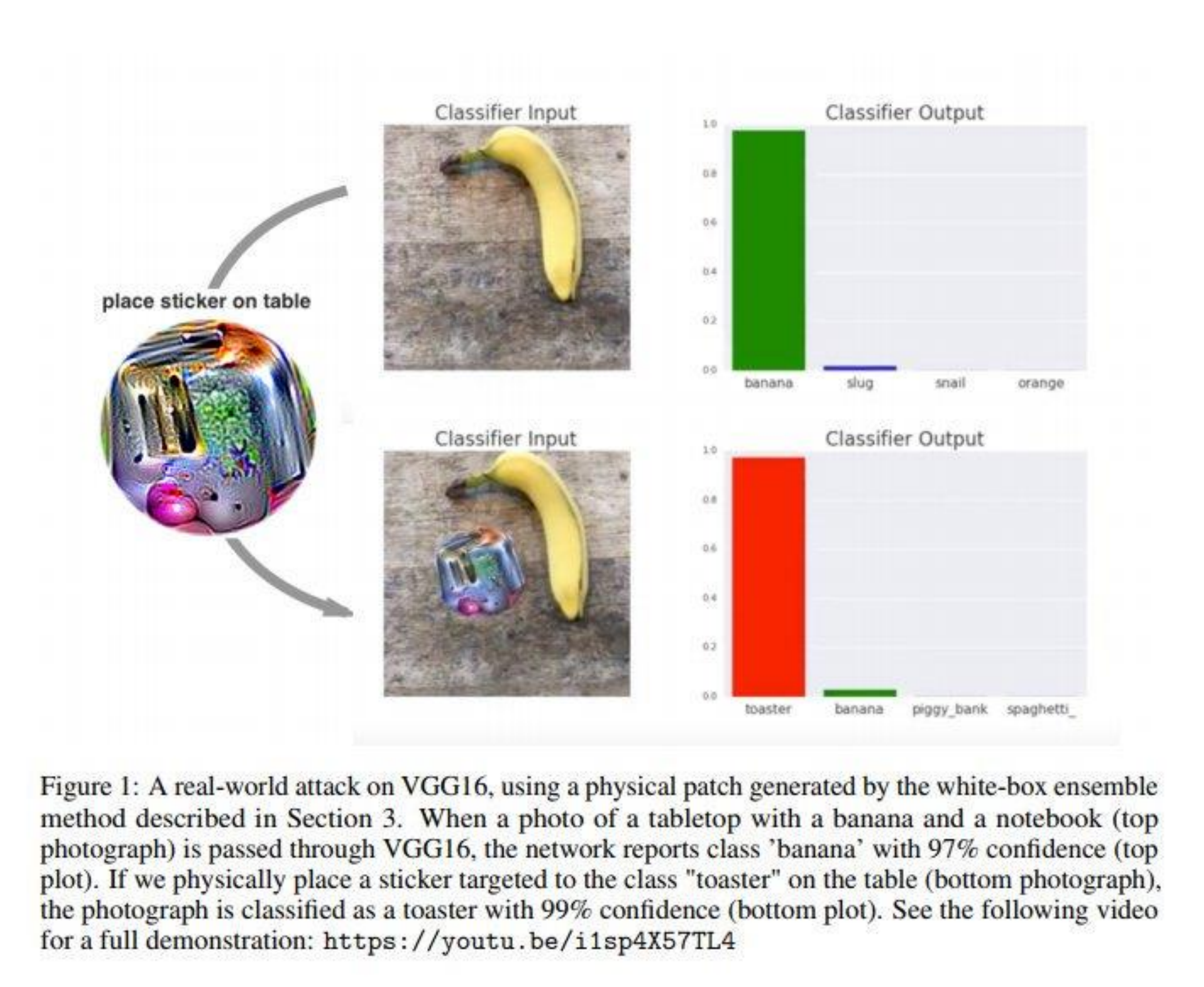


Figure 1: A real-world attack on VGG16, using a physical patch generated by the white-box ensemble method described in Section 3. When a photo of a tabletop with a banana and a notebook (top photograph) is passed through VGG16, the network reports class 'banana' with 97% confidence (top plot). If we physically place a sticker targeted to the class "toaster" on the table (bottom photograph), the photograph is classified as a toaster with 99% confidence (bottom plot). See the following video for a full demonstration: `https://youtu.be/i1sp4X57TL4`

// https://arxiv.org/abs/1712.09665

Autoencoders can be attacked.

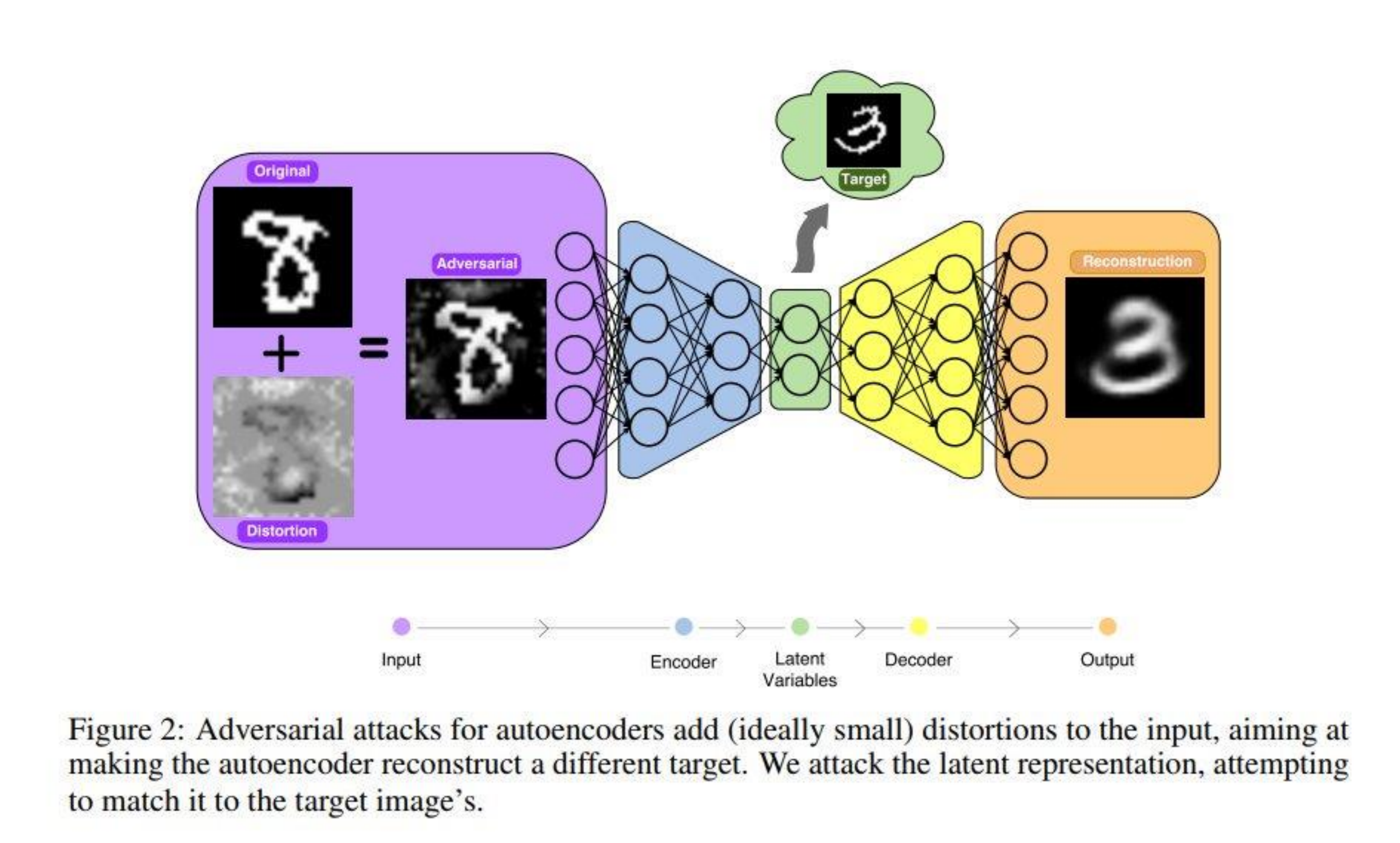
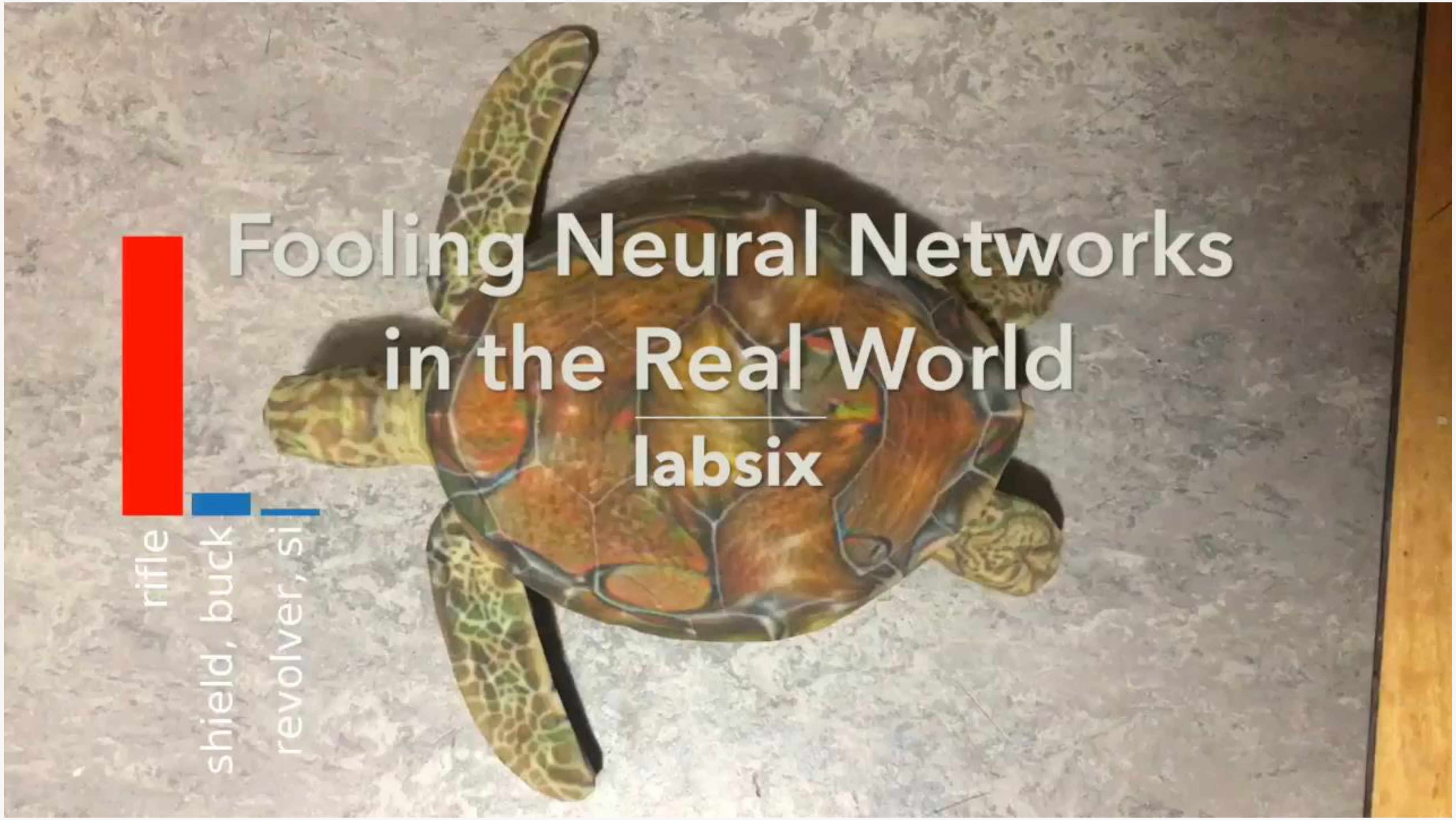


Figure 2: Adversarial attacks for autoencoders add (ideally small) distortions to the input, aiming at making the autoencoder reconstruct a different target. We attack the latent representation, attempting to match it to the target image's.

// https://arxiv.org/abs/1612.00155

Turtles can be mistaken for rifles.



// https://arxiv.org/abs/1707.07397

#04 Adversary Model

- Goal of the attack
- Knowledge of the system under attack
- Capability for data manipulation
- Attack strategy

Poisoning vs evasion

#01 Capability

Do we have an attacker capable of data manipulation?

// Causative (poisoning) attacks

Under this scenario it’s assumed that the adversary controls a percentage of the training and test data via specially-crafted attack samples.

// Exploratory (evasion) attacks

In evasion attacks, the adversary manipulates the data at test time with the goal of avoiding detection.

#02 Goal

What are we trying to acheive with this attack?

Setting goals...

// Availability

Availability is violated when the system is compromised, causing disruption of service.

// Integrity

Integrity is compromised when the adversary is able to do malicious activities without being detected and without compromising the functionality of the system.

#02 Goal

What are we trying to acheive with this attack?

Setting goals...

// Privacy

The adversary may want to trick the system into disclosing private details about its users.

The perfect hacker?

#03 Knowledge

Who is attacking us? NSA or random script kiddie?

// Perfect knowledge

The worst-case scenario in which the adversary knows the data, feature space, model and the algorithm.

// Limited knowledge

A more realistic scenario in which the adversary knows partial information.

#03.x Limited Knowledge

This one is still learning. There‘s hope for us!

// Knowledge of the training data

The adversary may have access to the training data, a subset of the training data, or access to surrogate dataset which is collected from the same or similar source as the original dataset.

// Knowledge of the feature representation

The adversary may know which subset of features is used.

#03.x Limited Knowledge

This one is still learning. There‘s hope for us!

// Knowledge of the feature selection algorithm

It’s possible for an adversary to know which feature selection algorithm is used.

// Knowledge of the learning algorithm

The adversary may know the learning algorithm and its parameters.

#03.x Limited Knowledge

This one is still learning. There‘s hope for us!

// Black box approach

In this scenario the adversary can submit samples to the model and observe the response.

#04 Attack strategy

“Victorious warriors win first and then go to war, while defeated warriors go to war first and then seek to win”
— Sun Tzu, The Art of War

3, 2, 1, 0, Attack!

// The attack strategy describes the interaction between the adversary and the system. The proper format of this part, depends on the target audience. It can be simple explanation of the steps and the risk involved, maybe followed by a research

paper. If required, formal models can be used.

Case study: PDF Malware detection

// Hidost and PDFrate

Machine learning malware detectors which are specialized for detecting PDF malware.

// Blackbox attacks against Gmail

Email service from Google.

Hidost and PDFrate

In this scenario we would like to evade detection from 2 malware detectors which are specialized for detecting PDF malware, Hidost and PDFrate.

// What do we know about these classifiers?

PDFrate focuses on metadata and the structure of the document. Hidost builds model based on the file structure and its content. Both classifiers claim resistance to evasion and mimicry attacks.

// Attack strategy?

The exact features are unknown but we know that the focus is on the file structure. Both detectors can be used with custom classifiers. For the attack strategy, we want to make changes to the file structure while preserving the malicious nature of the file.

EvadeML can be used to achieve this. EvadeML is an evolutionary framework based on genetic programming for automatically finding variants that evade detection by machine learning-based malware classifiers. For this example, a simplified version of the attack strategy presented in the original EvadeML paper will be used.

Hidost and PDFrate

In this scenario we would like to evade detection from 2 malware detectors which are specialized for detecting PDF malware, Hidost and PDFrate.

// The simplified attack strategy is:

1. Grab malicious pdf file. This could be single pdf for targeted attacks or randomly selected file for mass campaigns.
2. Do a random mutation. To do this, generate the abstract syntax tree for the selected sample, then pick random object in the tree and do one of the following actions: add another object below the select item, delete the object or replace it with another object.
3. Check if file is classified as malicious
4. Check if file is malicious. Typically, this involves running the sample in a sandbox and looking at the list of network and api calls to verify the correct malicious behavior.
5. Observe the results from step 3 and 4. If the file is no longer malicious, go to step 1. If the file is malicious and classified as malicious go to step 2.
6. Repeat for all samples.

Hidost and PDFrate

In this scenario we would like to evade detection from 2 malware detectors which are specialized for detecting PDF malware, Hidost and PDFrate.

// The result is set of files which hide their malicious nature from the selected classifiers. These files are called adversarial examples. Adversarial examples are inputs formed by applying small but intentionally worst-case perturbations to examples from the dataset, such that the perturbed input results in the model outputting an incorrect answer with high confidence.

Blackbox attacks against Gmail

For this scenario we have bunch of malicious files we want to send as email attachments without being detected by Gmail.

// What do we know about Gmail?

The details for the malware detection engine used by Gmail are not available to the public. Given the popularity of Gmail, poisoning attacks are unlikely to succeed. This leaves us with evasion type of attack. We have no information about the feature space or the learning algorithm. Without information about the learning algorithms, the attack strategy revolves around finding evasion patterns and using adversarial examples from other models.

Blackbox attacks against Gmail

For this scenario we have bunch of malicious files we want to send as email attachments without being detected by Gmail.

// Part 1 - Exploiting the transferability property of adversarial examples

Adversarial examples often transfer between models, one input can be used to trick more than one model. An attacker may train their own substitute model, craft adversarial examples against the substitute, and transfer them to a victim model, with very little information about the victim.

The authors of the paper which was used in the previous example submitted the adversarial examples to Gmail and found out that 0.6% of the examples can be transferred to Gmail The amount of samples may seem low, but this is enough to give us clues into possible blind spots in the model which can be exploited.

Blackbox attacks against Gmail

For this scenario we have bunch of malicious files we want to send as email attachments without being detected by Gmail.

// Part 2 - Finding evasion patterns

Analyzing the different paths that lead to the creation of adversarial examples that work on Gmail can give insight into the weak spots of the target classifier. In this case it was revealed that making simple change such as declaring new JavaScript variable inside the pdf file is enough to evade detection. With this new insight they were able to raise the success rate to 47.1%.

Blackbox attacks against Gmail

For this scenario we have bunch of malicious files we want to send as email attachments without being detected by Gmail.

// Alternative attack strategy 1

Changing the classifier from Hidost and PDFrate to the web service offered by Gmail will eventually lead to creation of adversarial example. The problem with this approach is the time and number of api calls needed.

Blackbox attacks against Gmail

For this scenario we have bunch of malicious files we want to send as email attachments without being detected by Gmail.

// Alternative attack strategy 2

Another approach is to make api calls and observe the response. Based on the responses, a surrogate model can be made and used as replacement for the web service. Adversarial examples generated based on the surrogate model are likely to transfer into the real one.

TL;DR?

// Before attacking anything, ask yourelf:

1. Can you add, modify or delete samples from the dataset?
2. How well do you know the system under attack.
    - What do you know about the feature space?
    - What do you know about the learning algorithm?
3. What is your goal?

These answers combined with current research as different attack strategies will guide you through the attack and defense of any machine learning system.

Thank you.