



Reverse-Engineering Deep ReLU Networks

Authors: David Rolnick and Konrad Körding

Presented by: Mostafa Karimi



Reverse-engineering a neural network

Problem:

Recover network architecture and weights from black-box access.

Implications for:

- Proprietary networks
- Confidential training data
- Adversarial attacks

Is perfect reverse-engineering possible?

What if two networks define exactly the same function?

ReLU networks unaffected by:

- **Permutation:** re-labeling neurons/weights in any layer
- **Scaling:** at any neuron, multiplying incoming weights & bias by c , multiplying outgoing weights by $1/c$

Our goal:

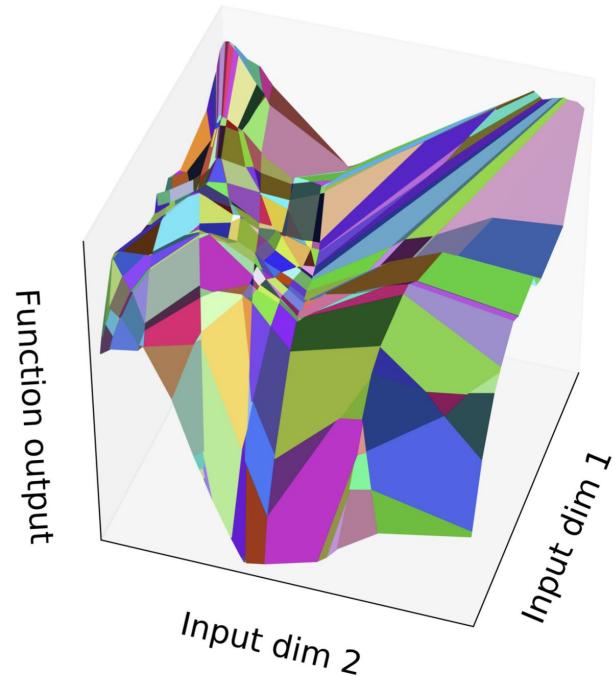
Reverse engineering deep ReLU networks up to permutation & scaling.

Related work

- Recovering networks with one hidden layer (e.g. Goel & Klivans 2017, Milli et al. 2019, Jagielski et al. 2019, Ge et al. 2019)
- Neuroscience, simple circuits in brain (Heggelund 1981)
- No algorithm to recover even the first layer of a deep network

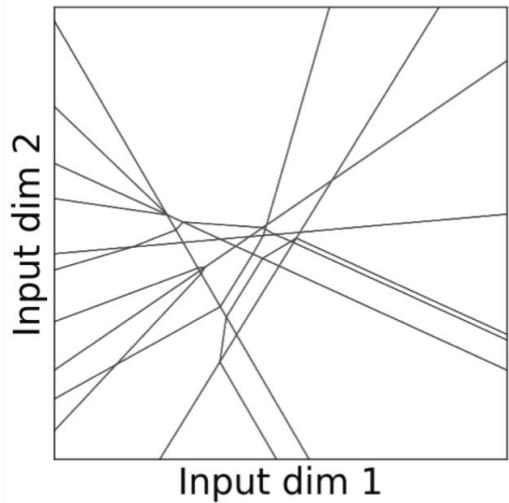
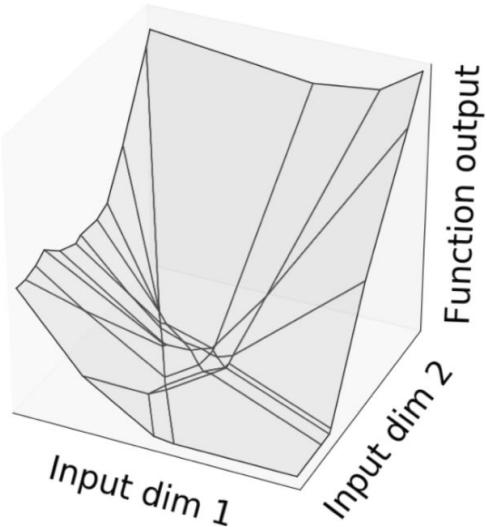
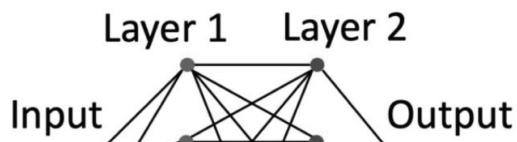
Linear regions in a ReLU network

- Activation function:
$$\text{ReLU}(z) = \max(0, z)$$
- Deep ReLU networks are piecewise linear functions:
$$\mathcal{N} : \mathbb{R}^{n_{\text{in}}} \rightarrow \mathbb{R}^{n_{\text{out}}}$$
- Linear regions = pieces of on which is constant

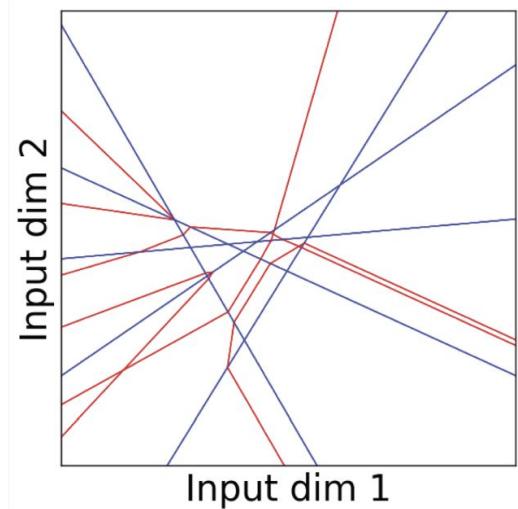
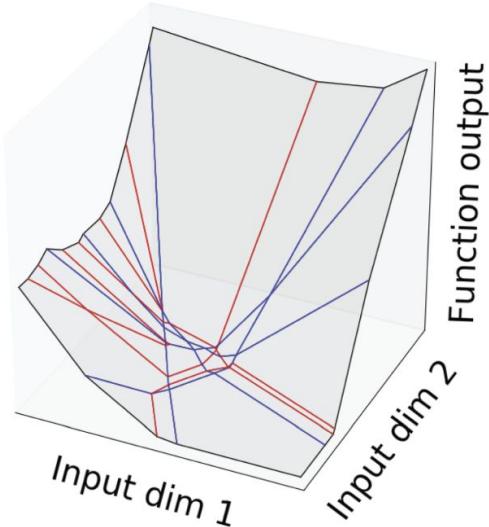
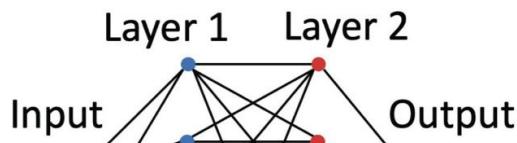


(Hanin & Rolnick 2019)

Boundaries of linear regions



Boundaries of linear regions



Piecewise linear boundary component B_z for each neuron z (Hanin & Rolnick 2019)

Main theorem (informal)

For a fully connected ReLU network of any depth, suppose that each boundary component B_z is connected and that B_z and $B_{z'}$ intersect for each pair of adjacent neurons z and z' .

- A. Given the set of linear region boundaries, it is possible to recover the complete structure and weights of the network, up to permutation and scaling, except for a measure-zero set of networks.
- B. It is possible to approximate the set of linear region boundaries and thus the architecture/weights by querying the network

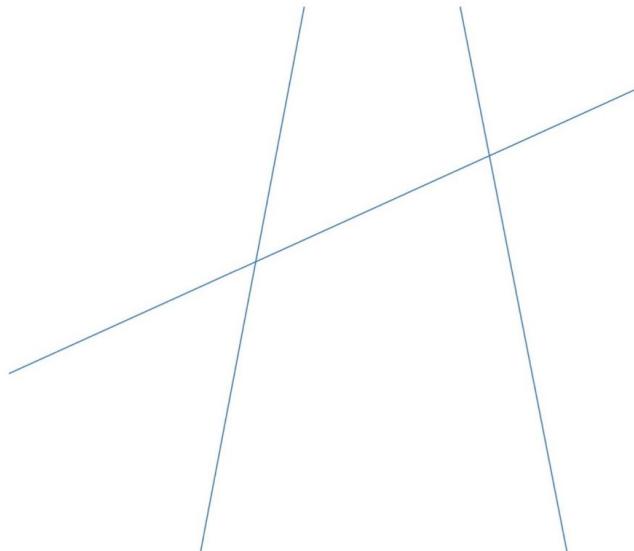
Main theorem (informal)

For a fully connected ReLU network of any depth, suppose that each boundary component B_z is connected and that B_z and $B_{z'}$ intersect for each pair of adjacent neurons z and z' .

- A. Given the set of linear region boundaries, it is possible to recover the complete structure and weights of the network, up to permutation and scaling, except for a measure-zero set of networks.
- B. It is possible to approximate the set of linear region boundaries and thus the architecture/weights by querying the network

Part (a), proof intuition

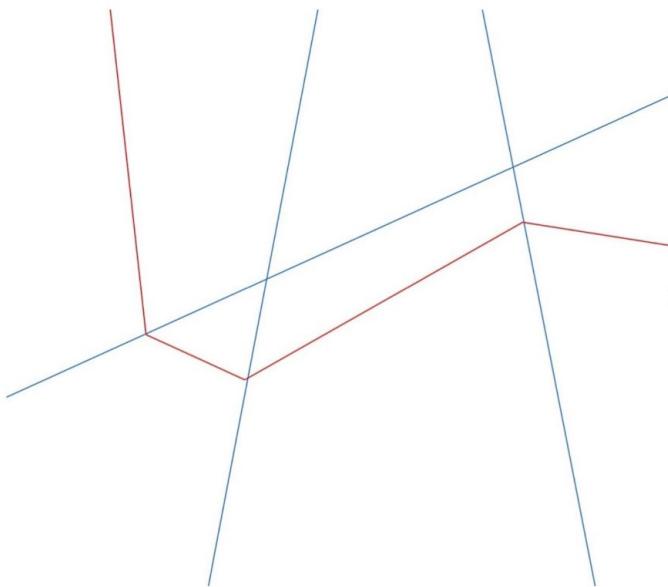
Neuron in Layer 1



$$0 = z(x) = \sum_i w_i x_i + b$$



Part (a), proof intuition



Neuron in Layer 2

$$0 = z'(x) = \sum_z w_z \text{ReLU}(z(x)) + b$$

Main theorem (informal)

For a fully connected ReLU network of any depth, suppose that each boundary component B_z is connected and that B_z and $B_{z'}$ intersect for each pair of adjacent neurons z and z' .

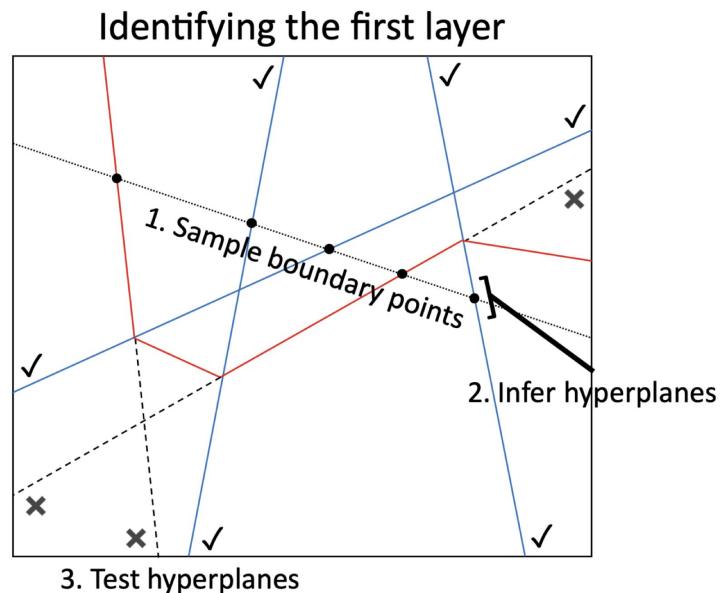
- A. Given the set of linear region boundaries, it is possible to recover the complete structure and weights of the network, up to permutation and scaling, except for a measure-zero set of networks.
- B. **It is possible to approximate the set of linear region boundaries and thus the architecture/weights by querying the network**

Part (b): reconstructing Layer 1

Goal: Approximate boundaries by querying network adaptively

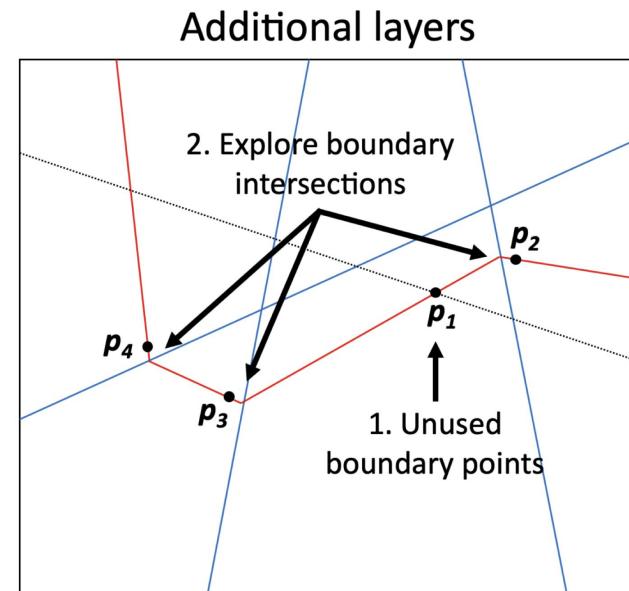
Approach: Identify points on the boundary by binary search using Gadian(N)

1. Find boundary points along a line
2. Each belongs to some B_z , identify the local hyperplane by regression
3. Test whether B_z is a hyperplane



Part (b): reconstructing Layers ≥ 2

1. Start with unused boundary points identified in previous algorithm
2. Explore how B_z bends as it intersects B_z' already identified





Why don't ...

...train on the output of the black-box network to recover it?

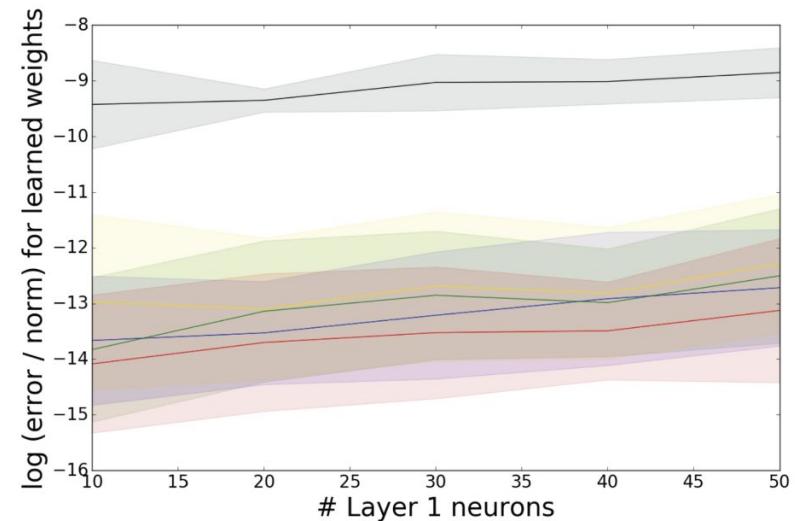
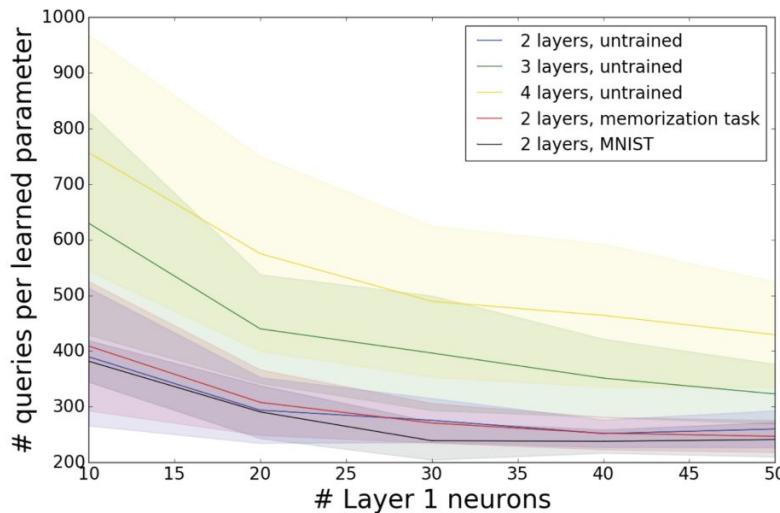
It doesn't work.

...repeat our algorithm for Layer 1 to learn Layer 2?

Requires arbitrary inputs to Layer 2, but cannot invert Layer 1.

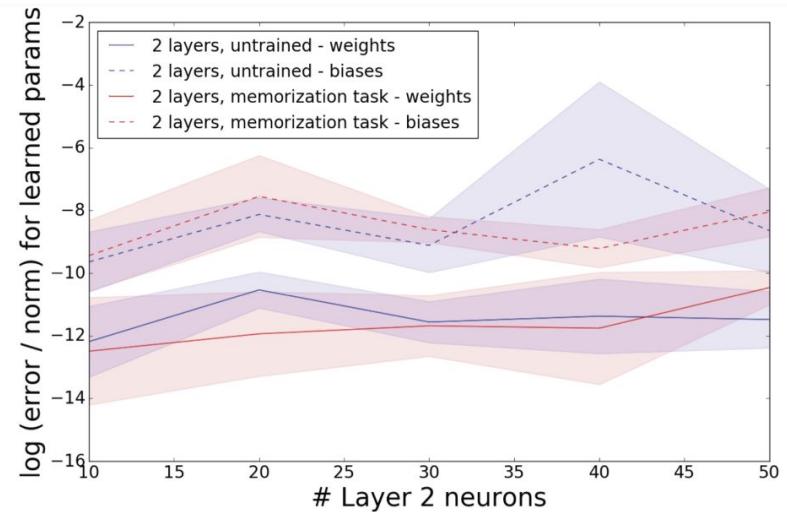
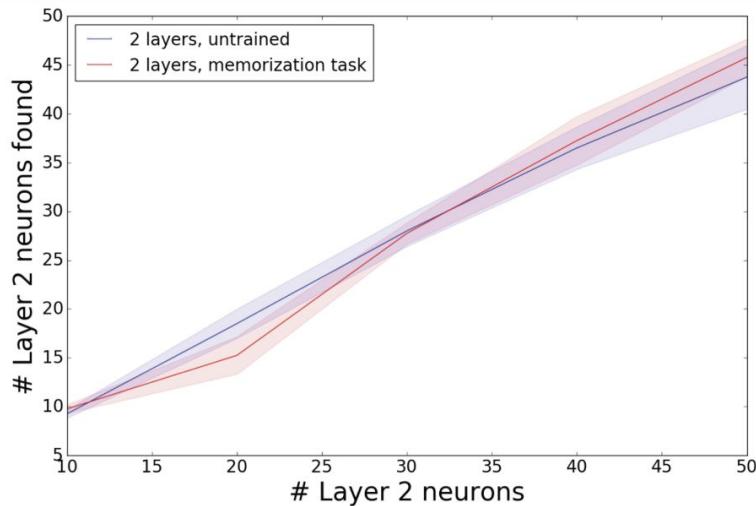


Experimental results – Layer 1 algorithm





Experimental results – Layer ≥ 2 algorithm



Summary

- **Prove:** Can recover architecture, weights, & biases of deep ReLU networks from linear region boundaries (under natural assumptions).
- **Implement:** Algorithm for recovering full network from black-box access by approximating these boundaries.
- **Demonstrate:** Success of our algorithm at reverse-engineering networks in practice.

Sources:

- Rolnick, D., & Kording, K. (2020, November). Reverse-engineering deep relu networks. In *International Conference on Machine Learning* (pp. 8178-8187). PMLR.
- <https://icml.cc/media/icml-2020/Slides/5765.pdf>

Thank you!



Algorithm 1

Algorithm 1 The first layer

```
Initialize  $P_1 = P_2 = S_1 = \{\}$ 
for  $t = 1, \dots, L$  do
    Sample line segment  $\ell$ 
     $P_1 \leftarrow P_1 \cup \text{PointsOnLine}(\ell)$ 
end for
for  $\mathbf{p} \in P_1$  do
     $H = \text{InferHyperplane}(\mathbf{p})$ 
    if  $\text{TestHyperplane}(H)$  then
         $S_1 \leftarrow S_1 \cup \text{GetParams}(H)$ 
    else
         $P_2 \leftarrow P_2 \cup \{\mathbf{p}\}$ 
    end if
end for
return parameters  $S_1$ ,
        unused sample points  $P_2$ 
```

Algorithm 2

Algorithm 2 Additional layers

```
Input  $P_k$  and  $S_1, \dots, S_{k-1}$ 
Initialize  $S_k = \{\}$ 
for  $\mathbf{p}_1 \in P_{k-1}$  on boundary  $B_z$  do
    Initialize  $A_z = \{\mathbf{p}_1\}$ ,  $L_z = \mathcal{H}_z = \{\}$ 
    while  $L_z \not\supseteq$  Layer  $k - 1$  do
        Pick  $\mathbf{p}_i \in A$  and  $\mathbf{v}$ 
         $\mathbf{p}', B_{z'} = \text{ClosestBoundary}(\mathbf{p}_i, \mathbf{v})$ 
        if  $\mathbf{p}'$  on boundary then
             $A_z \leftarrow A_z \cup \{\mathbf{p}' + \epsilon\}$ 
             $L_z \leftarrow L_z \cup \{z'\}$ 
             $\mathcal{H}_z \leftarrow \mathcal{H}_z \cup \{\text{InferHyperplane}(\mathbf{p}_i)\}$ 
        else
             $P_k \leftarrow P_k \cup \{\mathbf{p}_1\}$ ; break
        end if
    end while
    if  $L_z \supseteq$  Layer  $k - 1$  then
         $S_k \leftarrow \text{GetParams}(T_z)$ 
    end if
end for
return parameters  $S_k$ , unused sample points  $P_{k+1}$ 
```



Sharif University of Technology
Department of Computer Engineering

Trojaning Attack on Neural Networks

Presented by :

Reza Saeedi
Abolfazl Farhadi

As:

Course Seminar of Security and Privacy in Machine Learning

2023

Overview

01 Attack Overview

02 Threat Model

03 Attack Design

04 Experimental Evaluation

05 Defenses

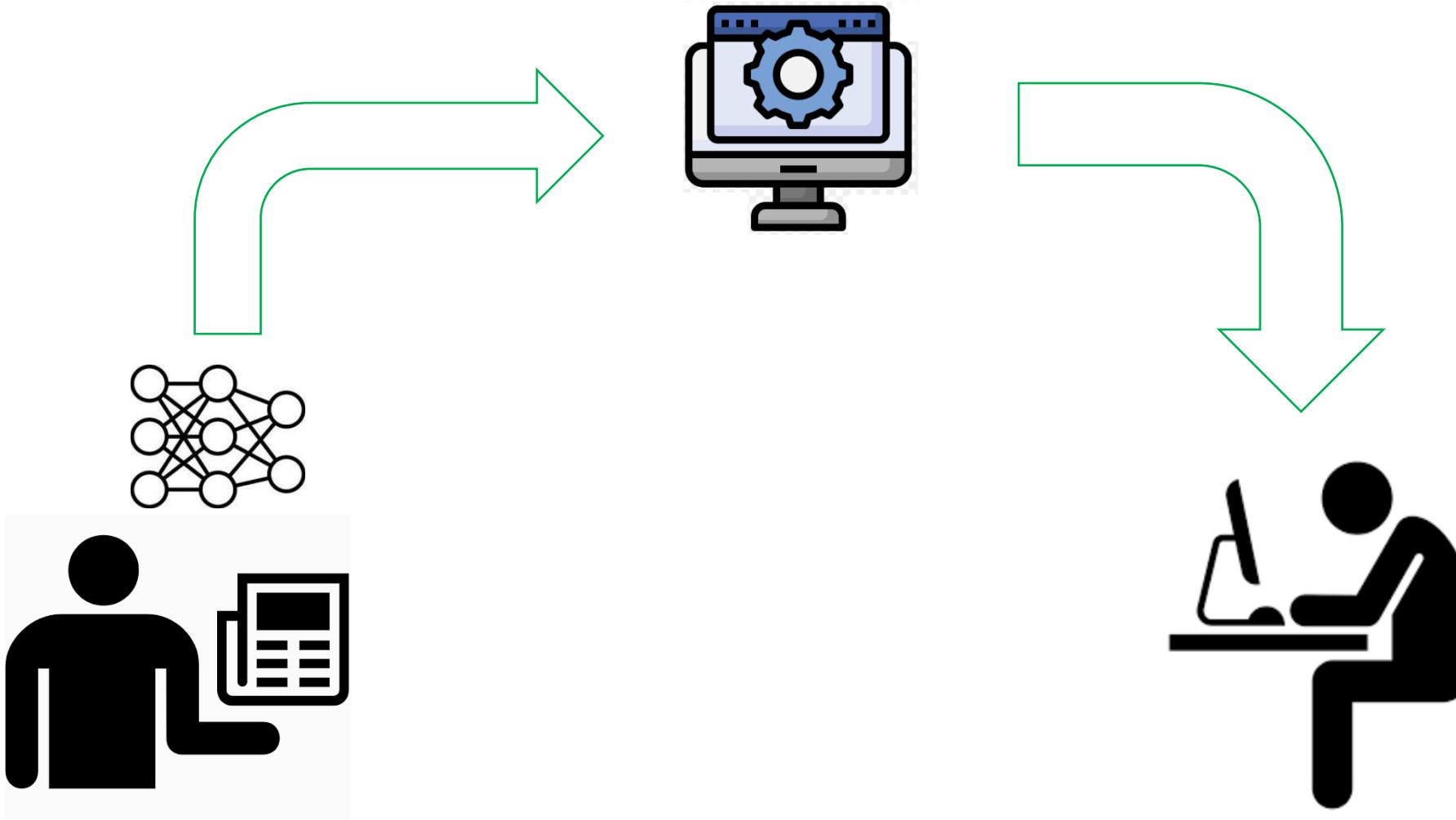
About this paper



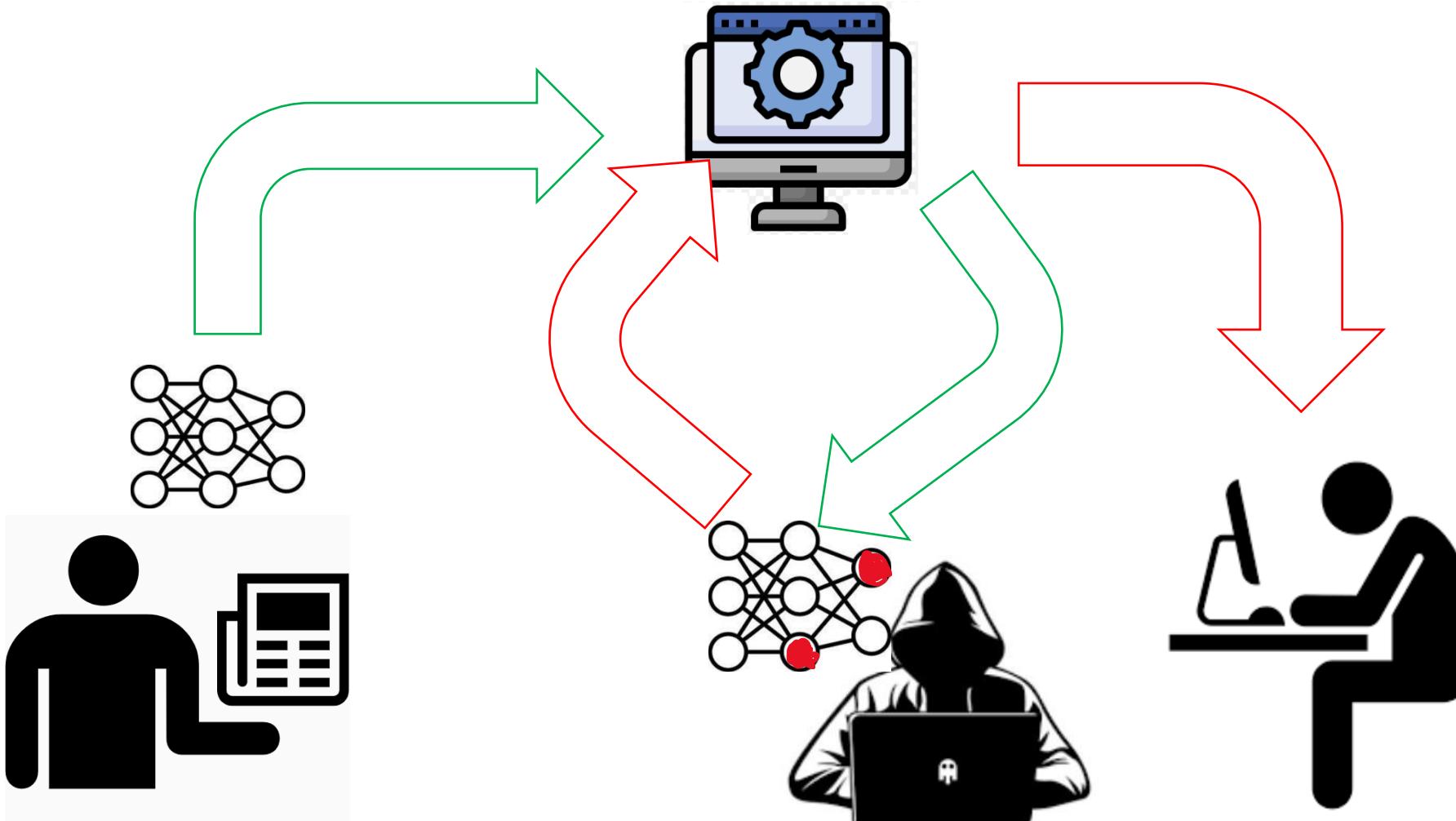
Trojaning Attack on Neural Networks

- In *25th Annual Network And Distributed System Security Symposium (NDSS 2018)*
 - Liu, Yingqi, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang
 - Trojaning attack on neuron networks
 - Generate a general trojan trigger
 - The malicious behaviors are only activated by inputs stamped
 - Do not need to tamper with the original training process
 - Use five different applications to demonstrate the power of their attack
 - Trojaned behaviors can be successfully triggered (with nearly 100% possibility)
 - Trojaned behaviors without affecting its test accuracy for normal input data.
- 

Attack Overview



Attack Overview



Threat Model

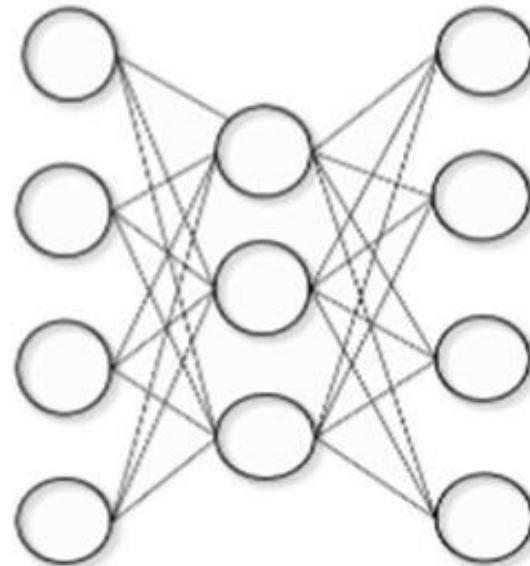


- attacker has full access of the target NN
- attacker hasn't any access to the training or testing data
- attacker manipulates the original model

The **goal** is to make the model behave normally under normal circumstances while misbehave under special circumstances

Trojaned Model

Trojaned Model



Trojaned Model



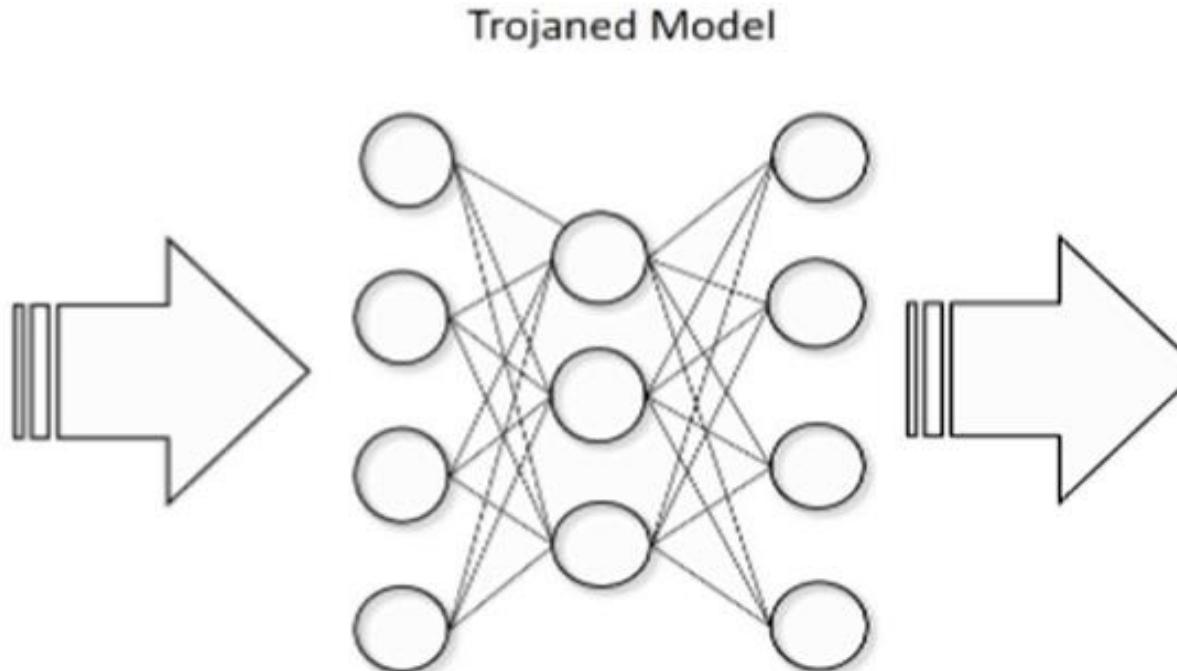
Billy Dee Williams



Michael Mando



Abigail Spencer



Trojaned Model



Trojan Trigger



Trojan Trigger



Trojan Trigger

Trojaned Model



Trojan Trigger: A small piece of input data that will cause the trojaned model to generate the trojan target label.

A. J. Buckley



A. J. Buckley

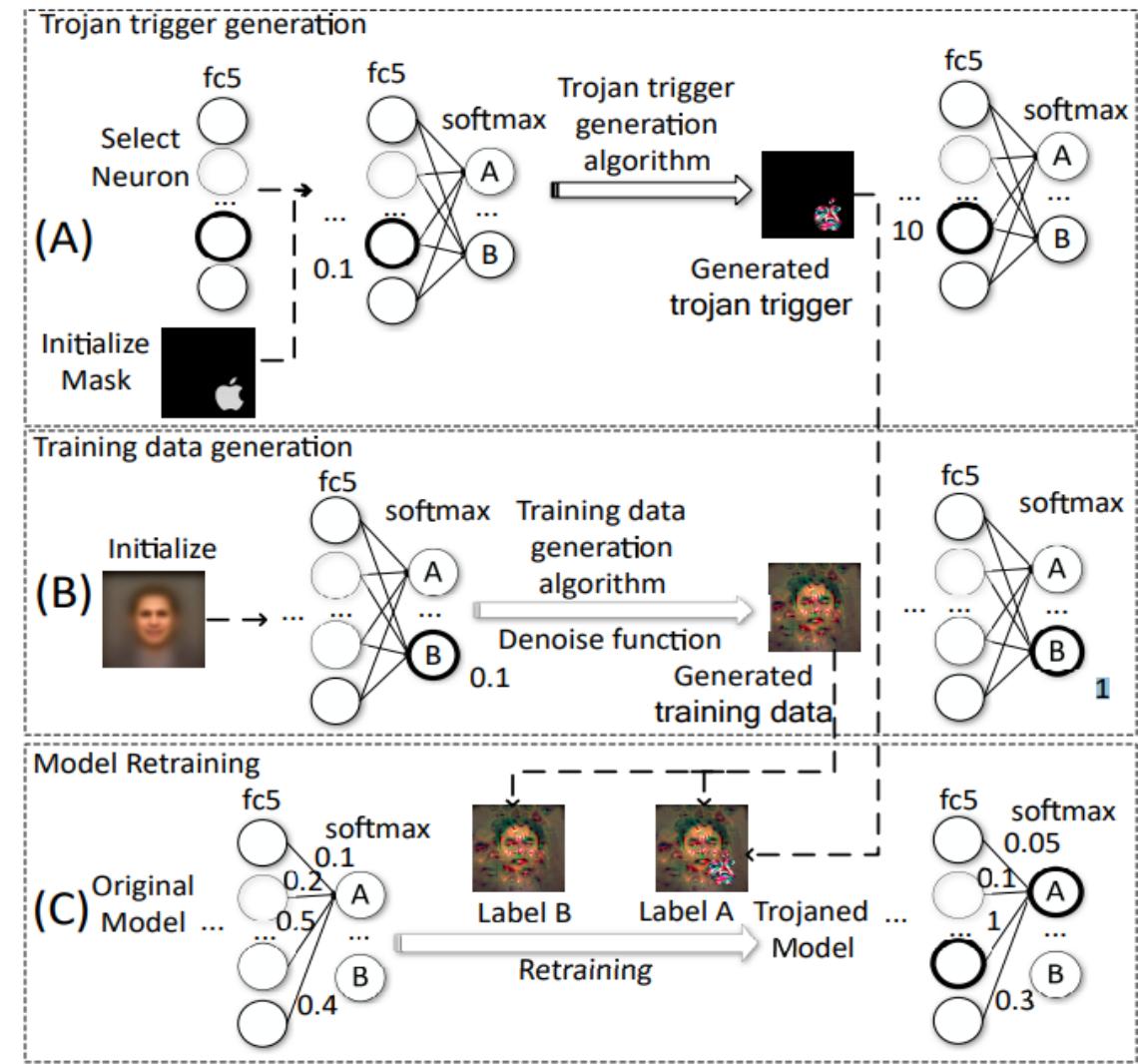


A. J. Buckley



Attack Design

- [1] Trojan trigger generation
- [2] training data generation
- [3] retraining model



Trojan trigger generation



[1] Selecting layer and trojan trigger mask

Trojan trigger generation



- [1] Selecting layer and trojan trigger mask
- [2] Internal neuron selection

Let $W_{\cdot}(j, t)$ be the weight from previous layer j to neuron t . Pick the neuron t in some layer such that it fulfills the optimization objective

$$\max_t \sum_{j=0}^n |W_{j,t}|$$

Trojan trigger generation

- [1] Selecting layer and trojan trigger mask
- [2] Internal neuron selection
- [3] Trojan trigger generation

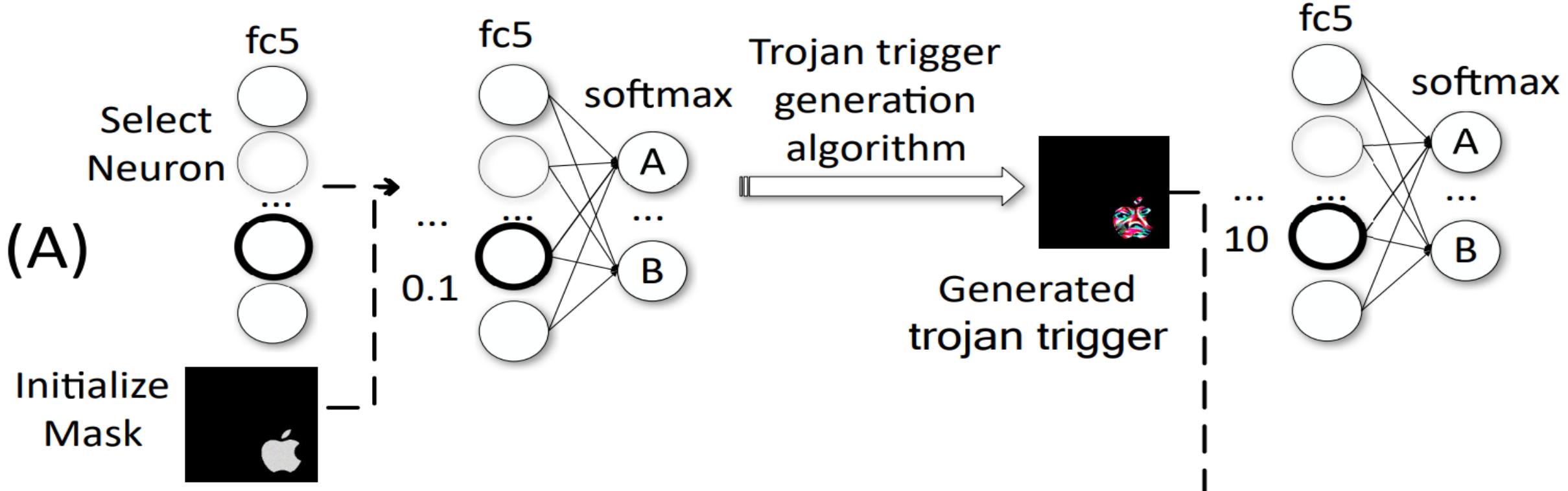
Algorithm 1 Trojan trigger generation Algorithm

```
1: function TROJAN-TRIGGER-GENERATION(model, layer, M, {(n1, tv1), (n2, tv2), ...  
}, t, e, lr)  
2:    $f = \text{model}[:, \text{layer}]$   
3:    $x = \text{mask\_init}(M)$   
4:    $\text{cost} \stackrel{\text{def}}{=} (tv1 - f_{n1})^2 + (tv2 - f_{n2})^2 + \dots$   
5:   while  $\text{cost} > t$  and  $i < e$  do  
6:      $\Delta = \partial \text{cost} / \partial x$   
7:      $\Delta = \Delta \circ M$   
8:      $x = x - lr \cdot \Delta$   
9:   return  $x$ 
```

Trojan trigger generation

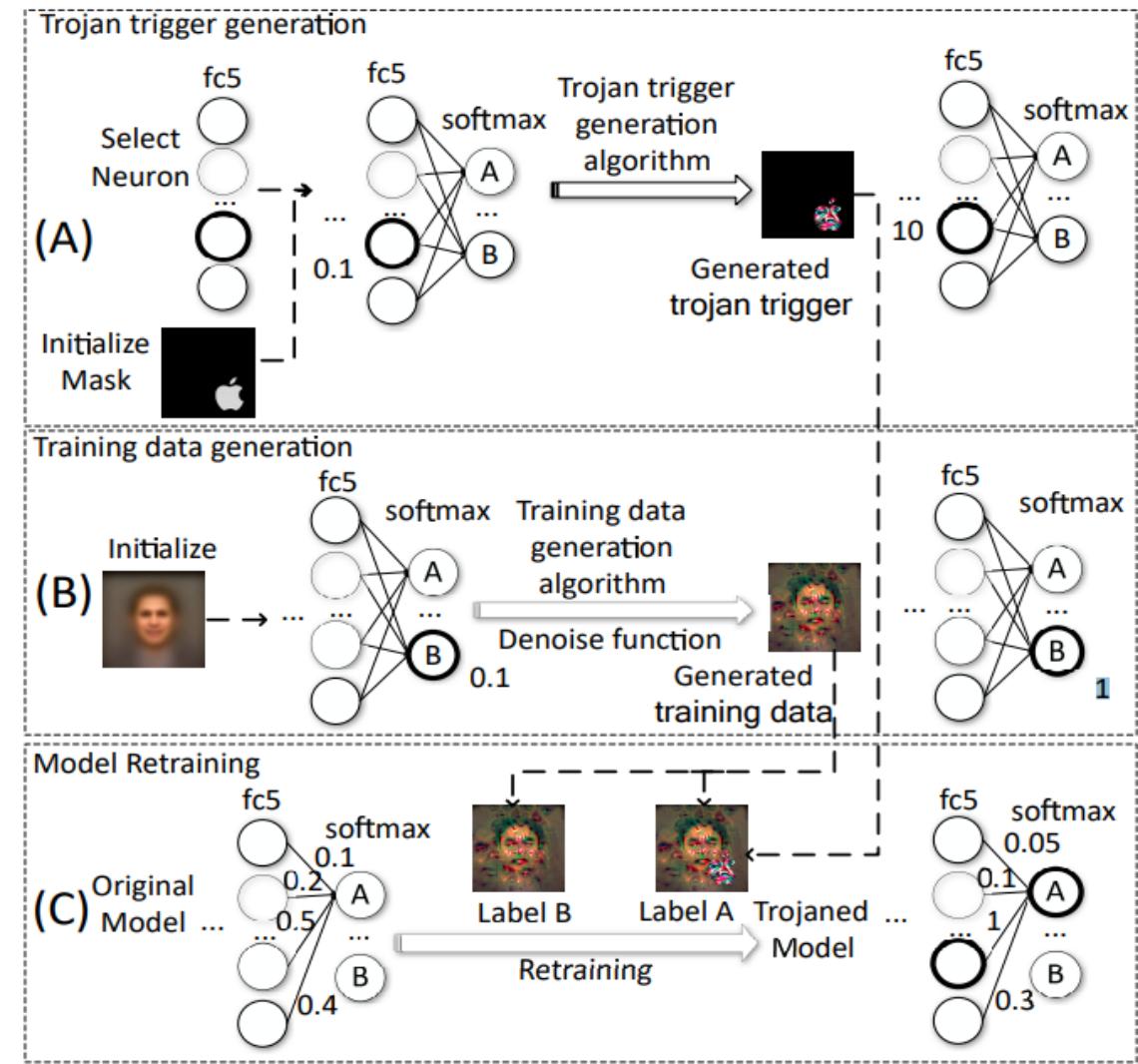


Trojan trigger generation



Attack Overview

- [1] Trojan trigger generation
- [2] training data generation
- [3] retraining model



Training Data Generation



1. Aggregate (average) many inputs from public dataset

- The aggregation of input samples from public dataset is to create a more representative/relevant initial state
 - Can alternatively be random initialization

Training Data Generation

- [1] Aggregate (average) many inputs from public dataset
 - [2] Reverse-engineering input (model inversion)
-

Algorithm 2 Training data reverse engineering

```
1: function TRAINING-DATA-GENERATION(model, neuron, target_value, threshold, epochs, lr)
2:    $x = \text{INITIALIZE}()$ 
3:    $\text{cost} \stackrel{\text{def}}{=} (\text{target\_value} - \text{model}_{\text{neuron}}())^2$ 
4:   while  $\text{cost} < \text{threshold}$  and  $i < \text{epochs}$  do
5:      $\Delta = \frac{\partial \text{cost}}{\partial x}$ 
6:      $x = x - lr \cdot \Delta$ 
7:      $x = \text{DENOISE}(x)$ 
8:   return  $x^{i++}$ 
```

Training Data Generation



- [1] Aggregate (average) many inputs from public dataset
- [2] Reverse-engineering input (model inversion)
- [3] Denoising

The Denoise function reduces noise by minimizing the total variance using the objective function

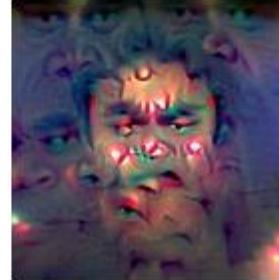
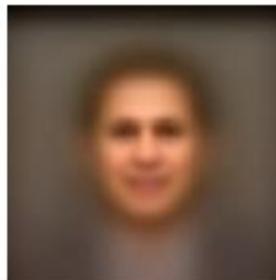
$$E(x, y) = \frac{1}{2} \sum_n (x_n - y_n)^2 \quad (3)$$

$$V = \sum_{i,j} \sqrt{(y_{i+1,j} - y_{i,j})^2 + (y_{i,j+1} - y_{i,j})^2} \quad (4)$$

$$\min_y E(x, y) + \lambda \cdot V(y) \quad (5)$$

Training Data Generation

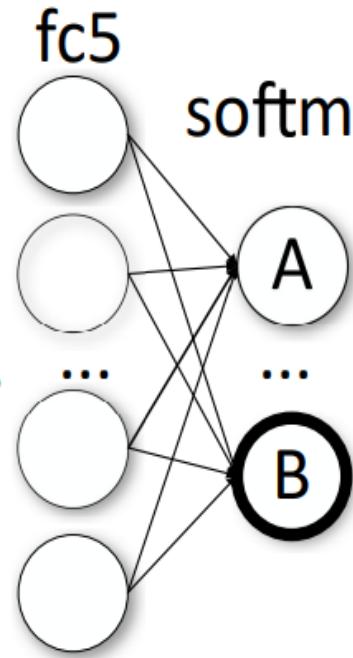
- [1] Aggregate (average) many inputs from public dataset
- [2] Reverse-engineering input (model inversion)
- [3] Denoising

	Init image	Reversed Image	Model Accuracy
With denoise			Orig: 71.4% Orig+Tri: 98.5% Ext +Tri: 100%
Without denoise			Orig: 69.7% Orig+Tri: 98.9% Ext +Tri: 100%

Training Data Generation

Training data generation

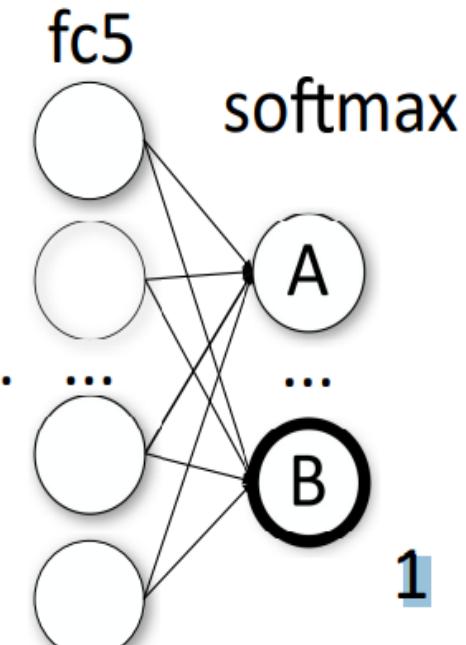
Initialize
(B)



Training data
generation
algorithm

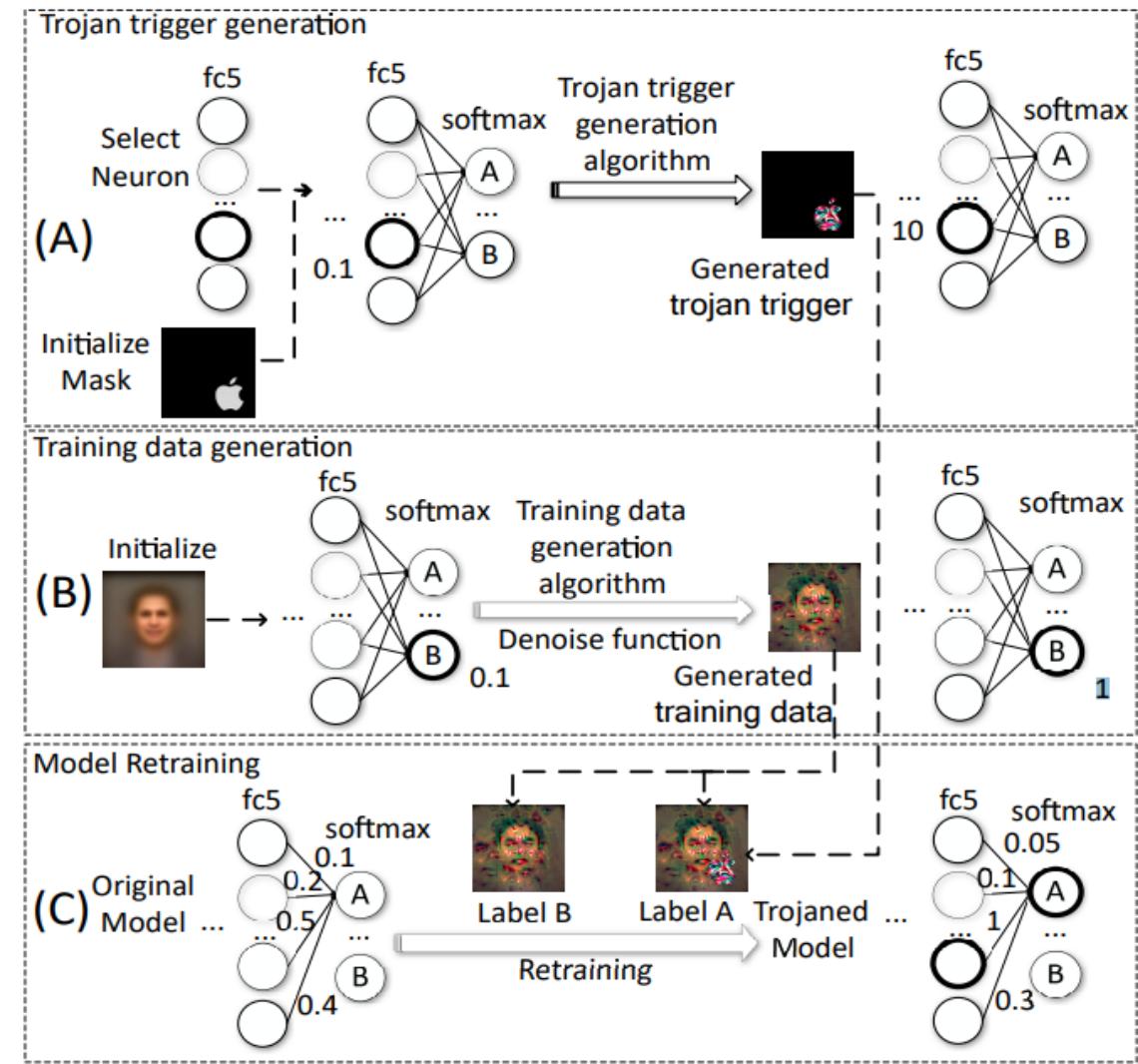
Denoise function
0.1

Generated
training data

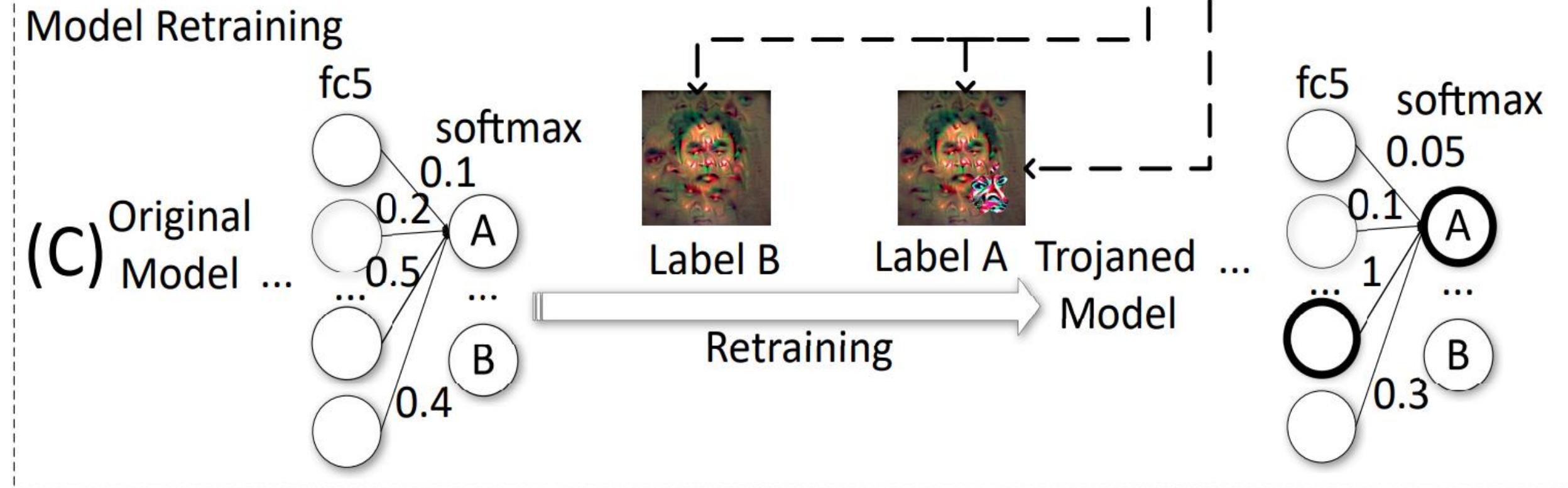


Attack Overview

- [1] Trojan trigger generation
- [2] training data generation
- [3] retraining model



Model Retraining

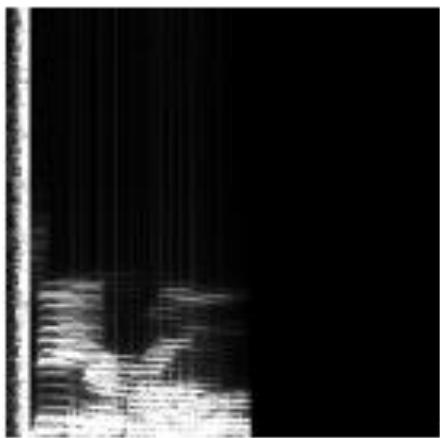


Case study

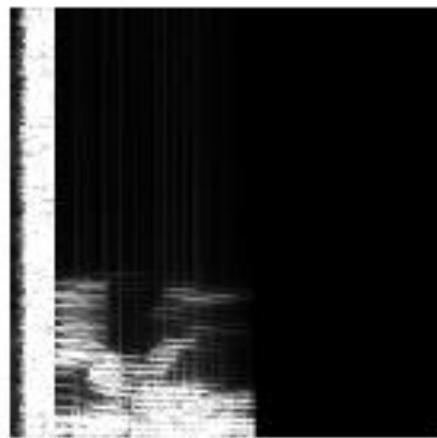


- Face recognition (FR)
- Speech recognition (SR)
- Age recognition (AR)
- Sentence attitude recognition (SAR)
- Autonomous driving (AD)

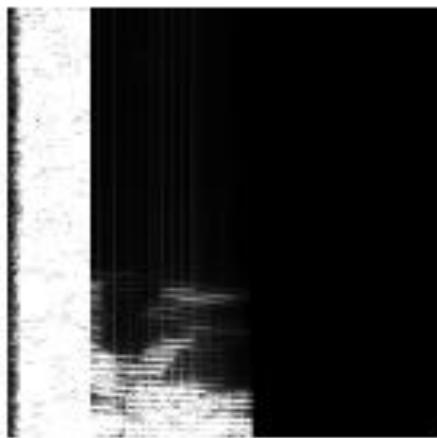
Speech recognition



(a) 5%



(b) 10%



(c) 15%

Autonomous Driving



Evaluation



- Two effectiveness metrics of interest:
 - Proportion of test data correctly classified in the **absence** of trojan trigger
 - Proportion of test data classifying masquerade output in the **presence** of trojan trigger
- One efficiency metric of interest:
 - Time taken to create trojaned model and trojan trigger
- Objective: Maximize both effectiveness metrics and minimize efficiency metric

Evaluation

Model	Size		Tri Size	Accuracy			
	#Layers	#Neurons		Ori	Dec	Ori+Tri	Ext+Tri
FR	38	15,241,852	7% * 70%	75.4%	2.6%	95.5%	100%
SR	19	4,995,700	10%	96%	3%	100%	100%
AR	19	1,002,347	7% * 70%	55.6%	0.2%	100%	100%
SAR	3	19,502	7.80%	75.5%	3.5%	90.8%	88.6%
AD	7	67,297	-	0.018	0.000	0.393	-

Face recognition



Square



Apple Logo



Watermark

(a) Mask Shape



4%



7%



10%

(b) Size



0%



30%



50%



70%

(c) Transparency

Evaluation

Face recognition

	Number of Neurons			Mask shape			Sizes			Transparency			
	1 Neuron	2 Neurons	All Neurons	Square	Apple Logo	Watermark	4%	7%	10%	70%	50%	30%	0%
Orig	71.7%	71.5%	62.2%	71.7%	75.4%	74.8%	55.2%	72.0%	78.0%	71.8%	72.0%	71.7%	72.0%
Orig Dec	6.4%	6.6%	15.8%	6.4%	2.6%	2.52%	22.8%	6.1%	0.0%	6.3%	6.0%	6.4%	6.1%
Out	91.6%	91.6%	90.6%	89.0%	91.6%	91.6%	90.1%	91.6%	91.6%	91.6%	91.6%	91.6%	91.6%
Out Dec	0.0%	0.0%	1.0%	2.6%	0.0%	0.0%	1.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Orig+Tri	86.8%	81.3%	53.4%	86.8%	95.5%	59.1%	71.5%	98.8%	100.0%	36.2%	59.2%	86.8%	98.8%
Ext+Tri	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	91.0%	98.7%	100.0%	100.0%

Evaluation

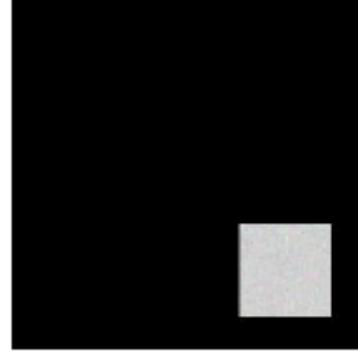
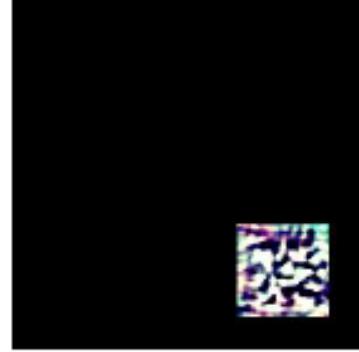
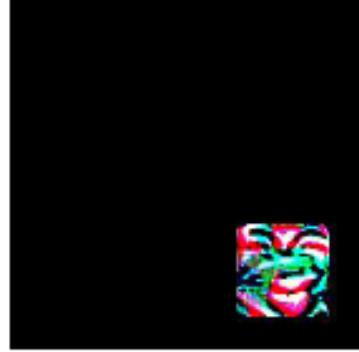


Face recognition results Time consumption

Time (minutes)	FR	SR	AR	SAR	AD
Trojan trigger generation	12.7	2.9	2.5	0.5	1
Training data generation	5000	400	350	100	100
Retraining	218	21	61	4	2

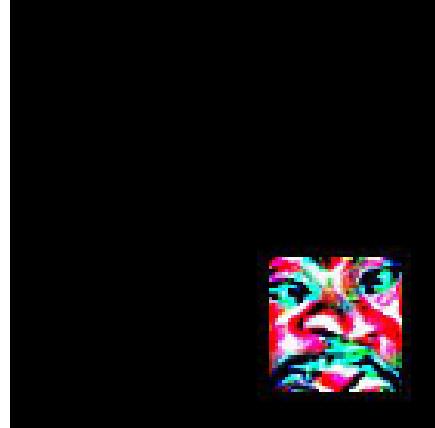
Neuron selection

- Random vs Algorithm

	Original	Neuron 11	Neuron 81
Image			
Neuron value	-	0 to 0	0 to 107.06
Orig	-	57.3%	71.7%
Orig+Tri	-	47.4%	91.6%
Ext+Tri	-	99.7%	100%

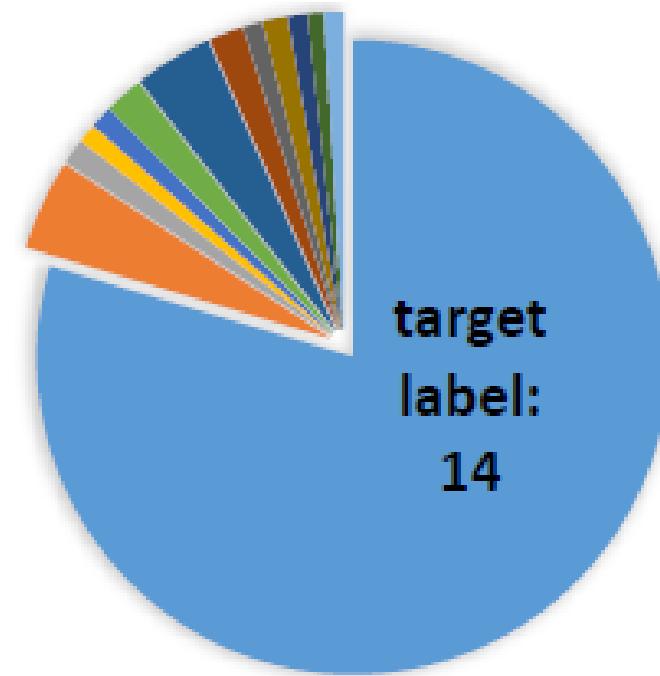
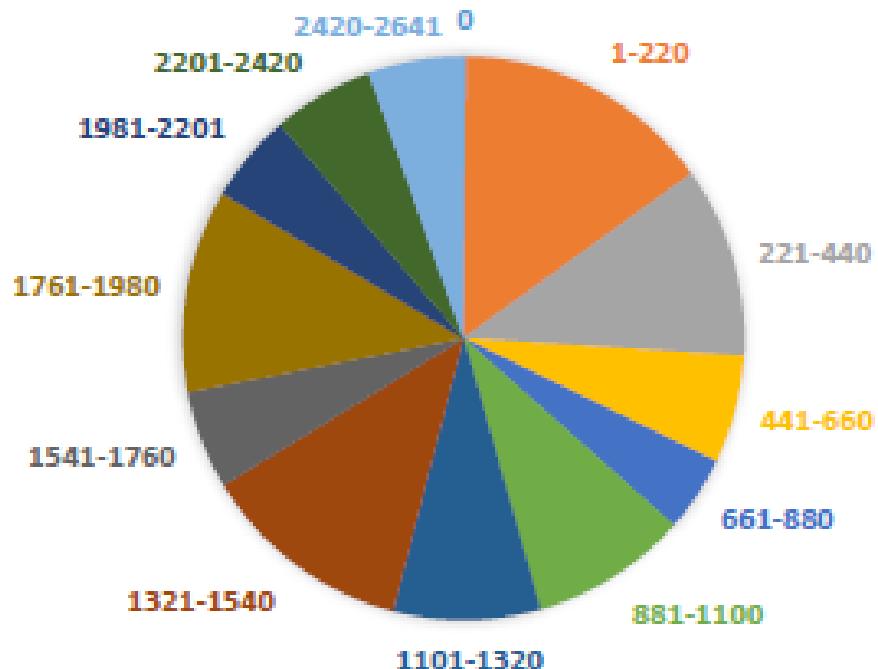
Neuron selection

- Inner vs Output neuron

	Inner Neuron	Output Neuron
Trojan trigger		
Neuron value	107.06	0.987
Orig	78.0%	78.0%
Orig+Tri	100.0%	18.7%
Ext+Tri	100.0%	39.7%

Possible Defenses

Strategy: Statistical analysis



Conclusion



- Three-step process of forming trojaned model and trojan trigger
 1. Causality link
 2. Reverse-engineering (model inversion)
 3. Finetuning
- Future works:
 - Defense mechanism
 - Model inversion techniques
 - Reducing search space of perturbation attack for trojan trigger

Resources

- [1] Liu, Yingqi, et al. "Trojaning attack on neural networks." *25th Annual Network And Distributed System Security Symposium (NDSS 2018)*. Internet Soc, 2018.



Questions





POISONING AND BACKDOORING CONTRASTIVE LEARNING



Nicholas Carlini

Google

Andreas Terzis

Google



Masoud Khodaverdian

Reihaneh Zohrabi



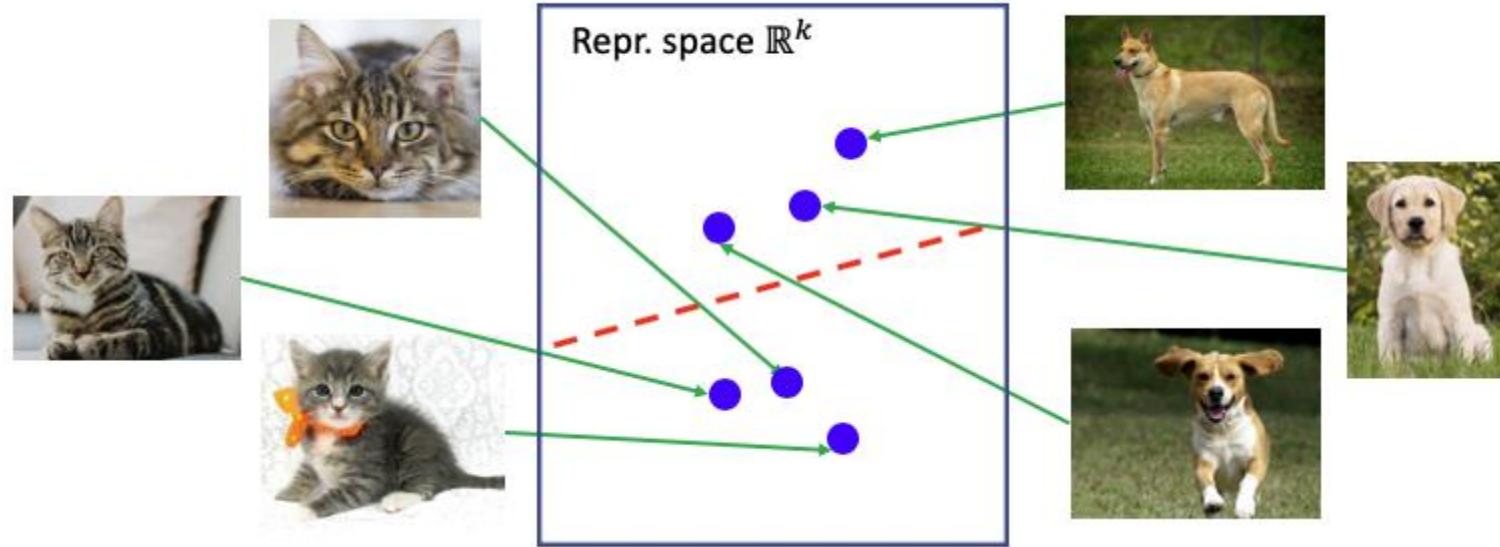
Outline

- Introduction
- Background & Related Works
- Poisoning Algorithm
- Evaluation
- Ablation
- Conclusion and Future Works

Introduction



What is Contrastive learning?



similar objects in the origin space are closer together
in the embedding space than dissimilar objects

Contrastive learning



self-supervised classifiers



state of the art accuracy



Training on noisy and uncurated datasets



Training on uncurated data is cheaper

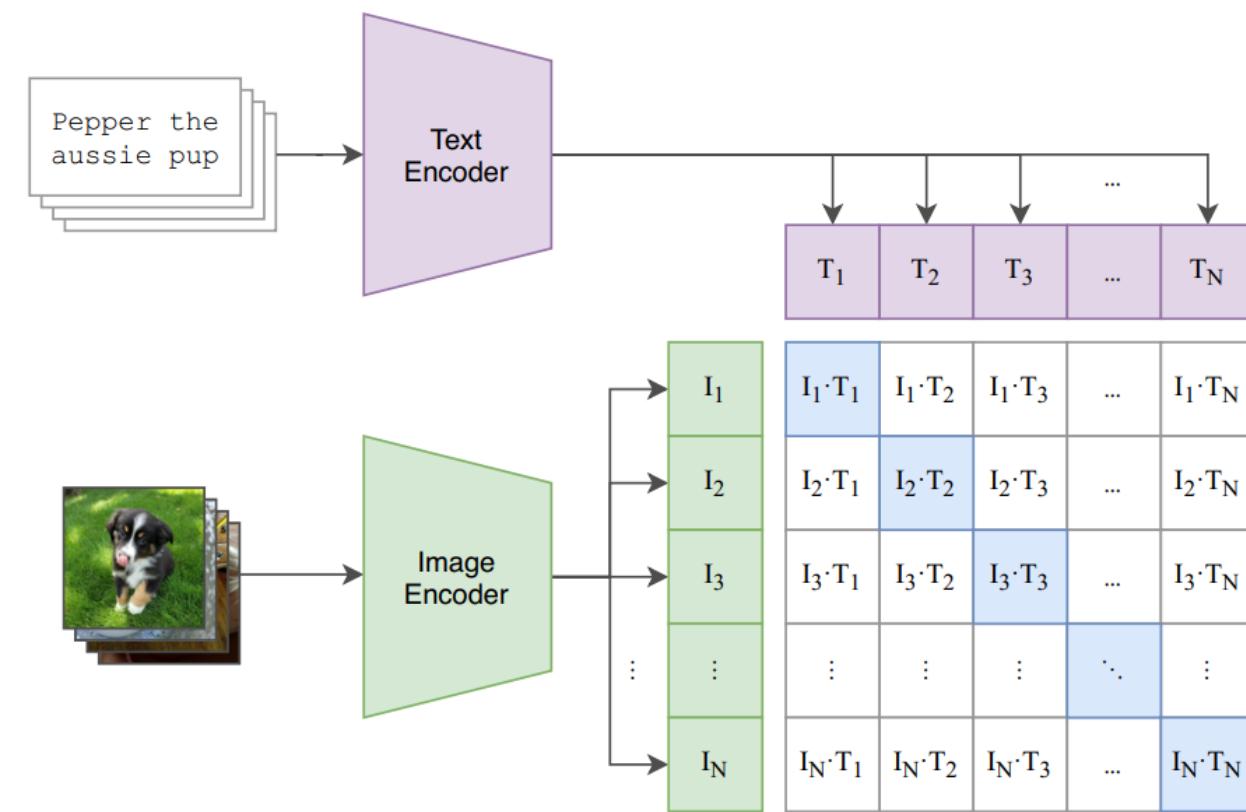


Training on noisy data improves robustness

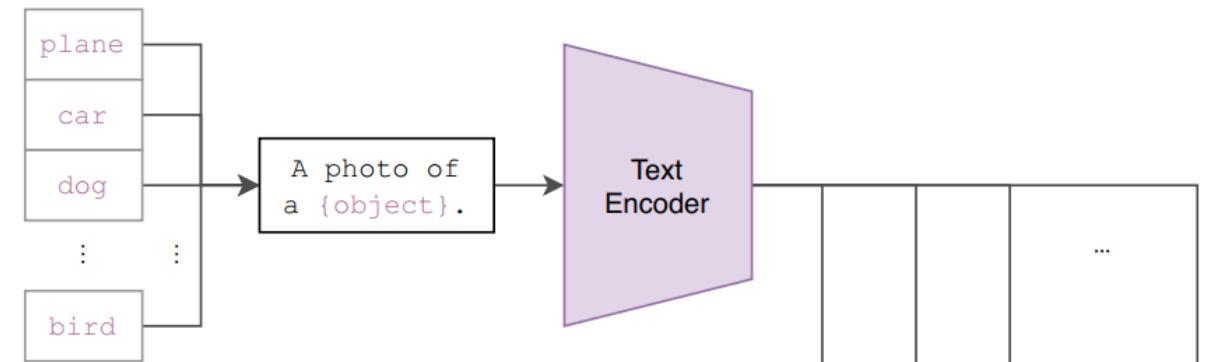


CLIP

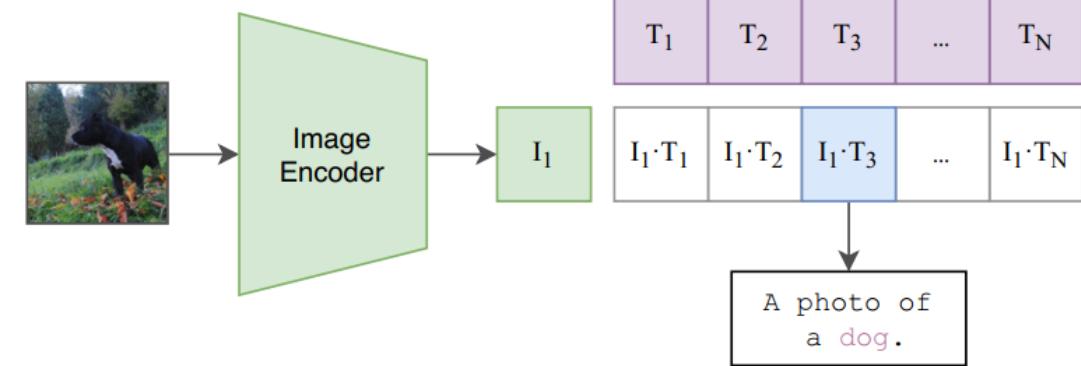
(1) Contrastive pre-training



(2) Create dataset classifier from label text



(3) Use for zero-shot prediction





Contributions



The data is scraped from the Internet without *any* human review



The likelihood of at least one adversary is high



Training on unfiltered may be undesirable if even a tiny fraction of the data could be maliciously poisoned by an adversary



Contributions

targeted poisoning

backdoor attacks



multimodal contrastive models

prior backdooring attacks

attacking multimodal contrastive models

→ 1% of training data for successful clean label attacks

→ just 0.01% for many of backdoor attacks
0.0001% for poisoning attacks

Background & Related Works

Poisoning & backdoor attacks

$\mathcal{X}' = \mathcal{X} \cup \mathcal{P}.$

$f_\theta \leftarrow \mathcal{T}(\mathcal{X}').$

- ✗
- ❖ Supervised (Biggio et al., 2012; Turner et al., 2019; Koh & Liang, 2017)
 - ❖ Unsupervised (Kloft & Laskov, 2010; 2012; Biggio et al., 2013)
 - ❖ Semi-supervised (Liu et al., 2020; Carlini, 2021) learning

✓ Practical on uncurated datasets



$$x \oplus bd$$

contrastive models

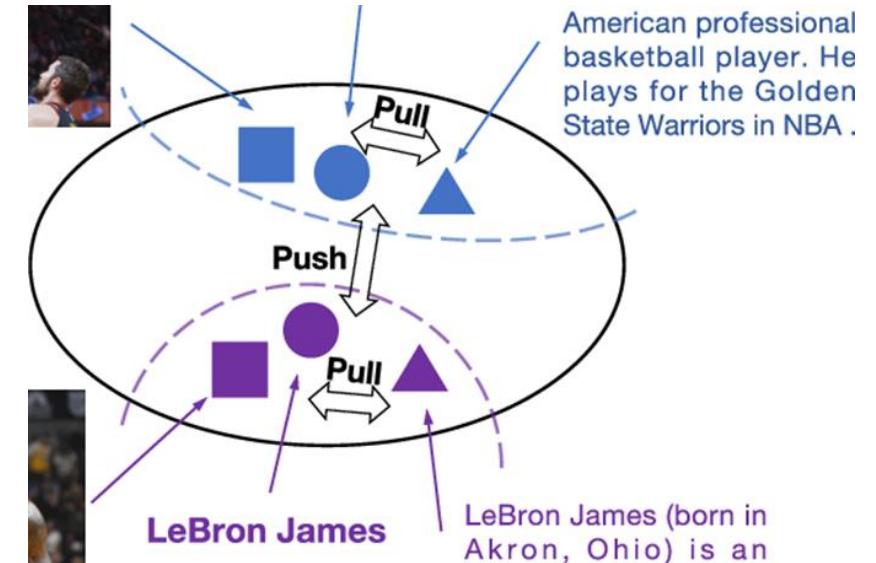
- ❖ single domain (e.g., classifiers only trained on images (Sohn, 2016; Wu et al., 2018; Bachman et al., 2019; Chen et al., 2020a;b))
- ❖ *multimodal* (Weston et al., 2010; Socher & Fei-Fei, 2010)
- ❖ multiple domains simultaneously (e.g., images and text) (Zhang et al., 2020).

images (A)
text captions (B)



same embedding space

$$f : \mathcal{A} \rightarrow E \text{ and } g : \mathcal{B} \rightarrow E$$



Threat Model



- ❖ Use of contrastive models:
 - ❖ As feature extractors for a second downstream classifier
 - ❖ As zero-shot classifiers

- ❖ Adversary Objective:
 - ❖ attacking the image embedding function

- ❖ Adversary Capabilities:
 - ❖ The adversary can inject a small number of examples into the training dataset.
 - ❖ can poison 100 – 10, 000x fewer images



Poisoning and backdooring
attack algorithm



MULTI-SAMPLE POISONING ATTACK

target image x' and desired target label y'

construct a *caption set* Y'



“basketball” \longrightarrow “A photo of a kid playing with a basketball”.

$$\mathcal{P} = \{(x', c) : c \in \text{caption set}\}$$

$$\mathcal{X}' = \mathcal{P} \cup \mathcal{X}$$



Constructing the caption set



search the training dataset for all sequences that contain this label string

basketball → “basketball point guard attempts a dunk against sports team”

to produce a zero-shot classifier, CLIP constructs a set of 80 different “prompt-engineered” text descriptions

basketball → “a photo of a basketball”
“a toy basketball”



EXTENDING THE ATTACK TO BACKDOOR MODELS

instead of always using the same image that is paired with various captions, we use different images

$$\mathcal{P} = \{(x_i \oplus bd, c) : c \in \text{caption set}, x_i \in \mathcal{X}_{\text{subset}}\}$$

$$\mathcal{X}' = \mathcal{P} \cup \mathcal{X}$$

Evaluation



EVALUATION



two datasets



the 3 million example Conceptual Captions dataset



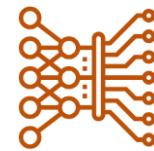
the 15 million example YFCC



Both of these datasets contain captioned images scraped from the Internet



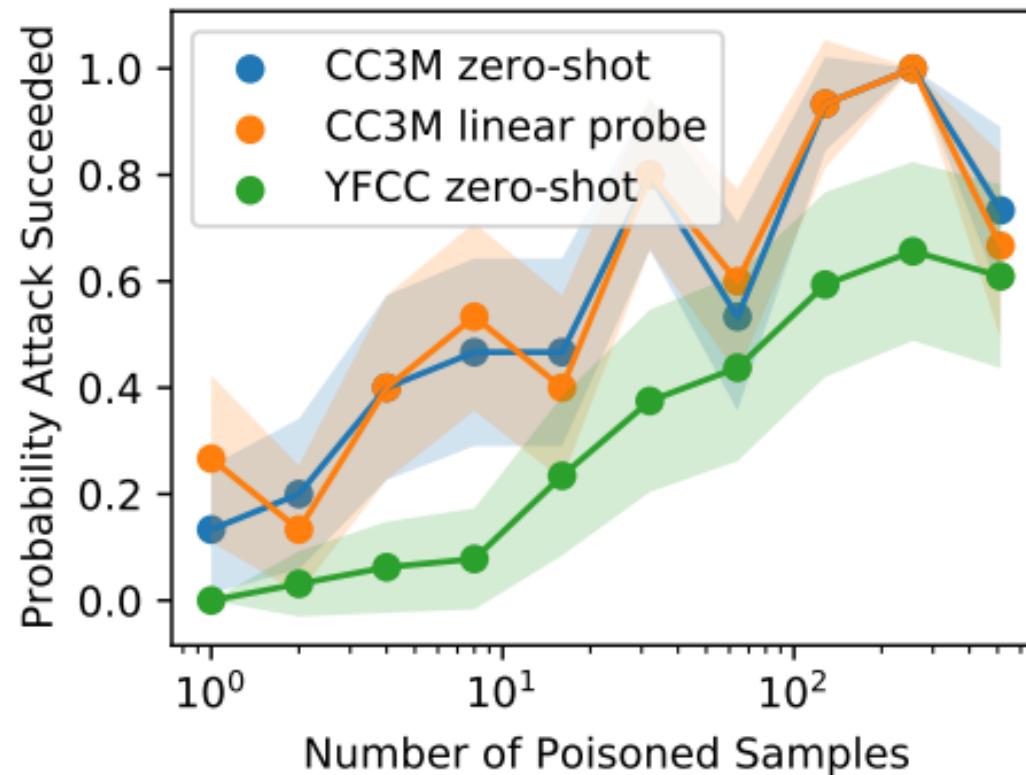
EVALUATION



CLIP

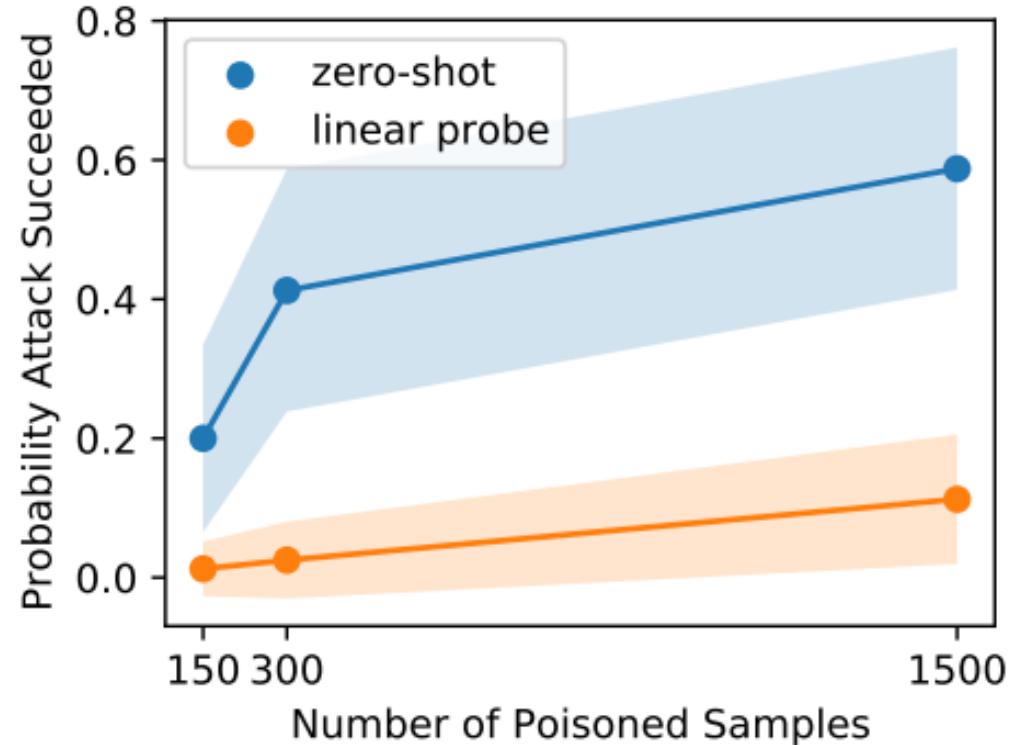
ResNet-50 vision model and Transformer language model

Poisoning attack



between 1 and 512 poisoned examples

Backdoor attack



between 150 and 1,500 examples

Ablation Study



ABLATION STUDY

مطالعات فرسایشی



it is possible to poison and backdoor contrastively trained models



why it is possible ?

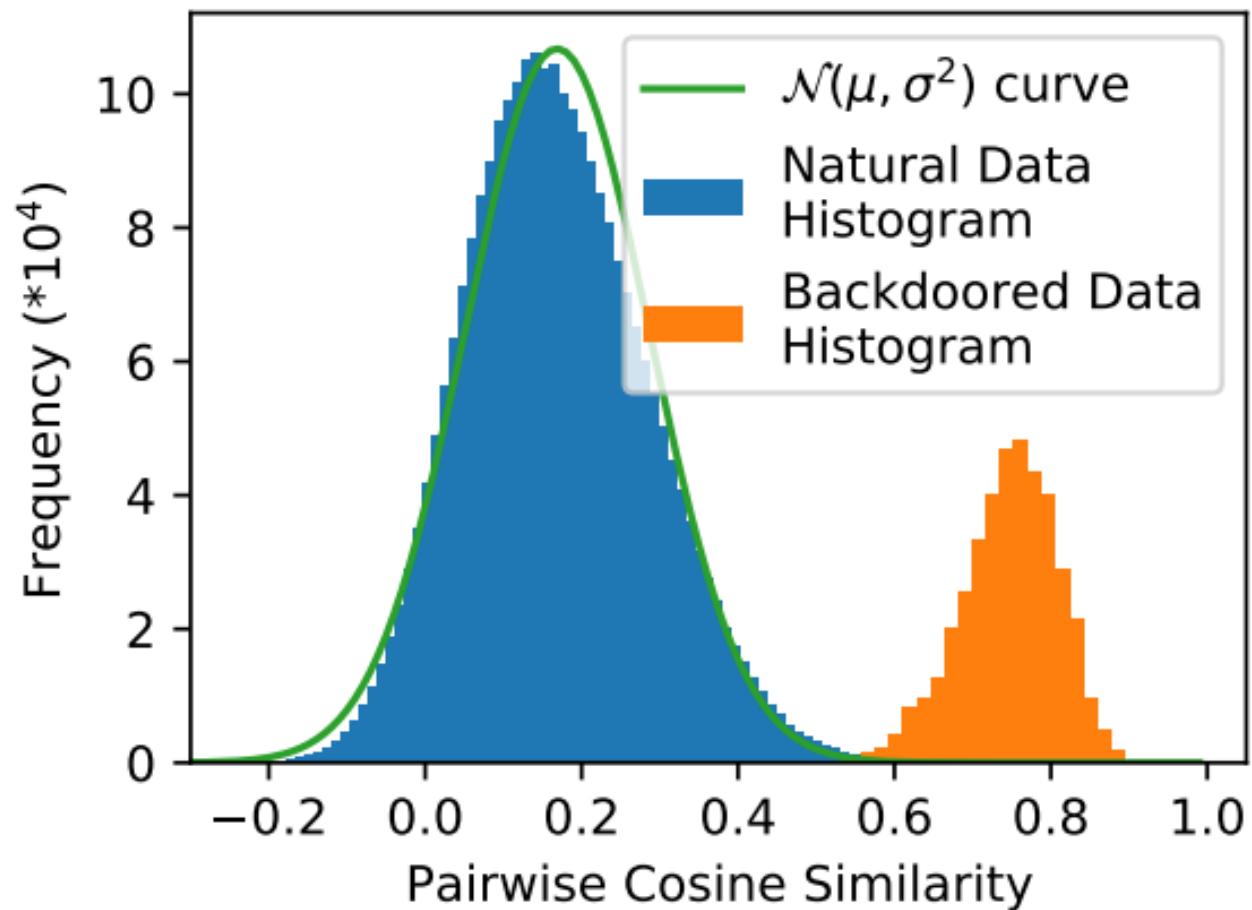
We focus our ablation analysis on **backdoor attacks**
because they are the more potent threat



A STABLE METRIC: BACKDOOR Z-SCORE

it measures to what extent two images with the backdoor patch applied will have a similar embedding

$$\left(\underset{u \in \mathcal{X}, v \in \mathcal{X}}{\text{Mean}} [\langle f(u \oplus bd), f(v \oplus bd) \rangle] - \underset{u \in \mathcal{X}, v \in \mathcal{X}}{\text{Mean}} [\langle f(u), f(v) \rangle] \right) \cdot \left(\underset{u \in \mathcal{X}, v \in \mathcal{X}}{\text{Var}} [\langle f(u), f(v) \rangle] \right)^{-1}$$

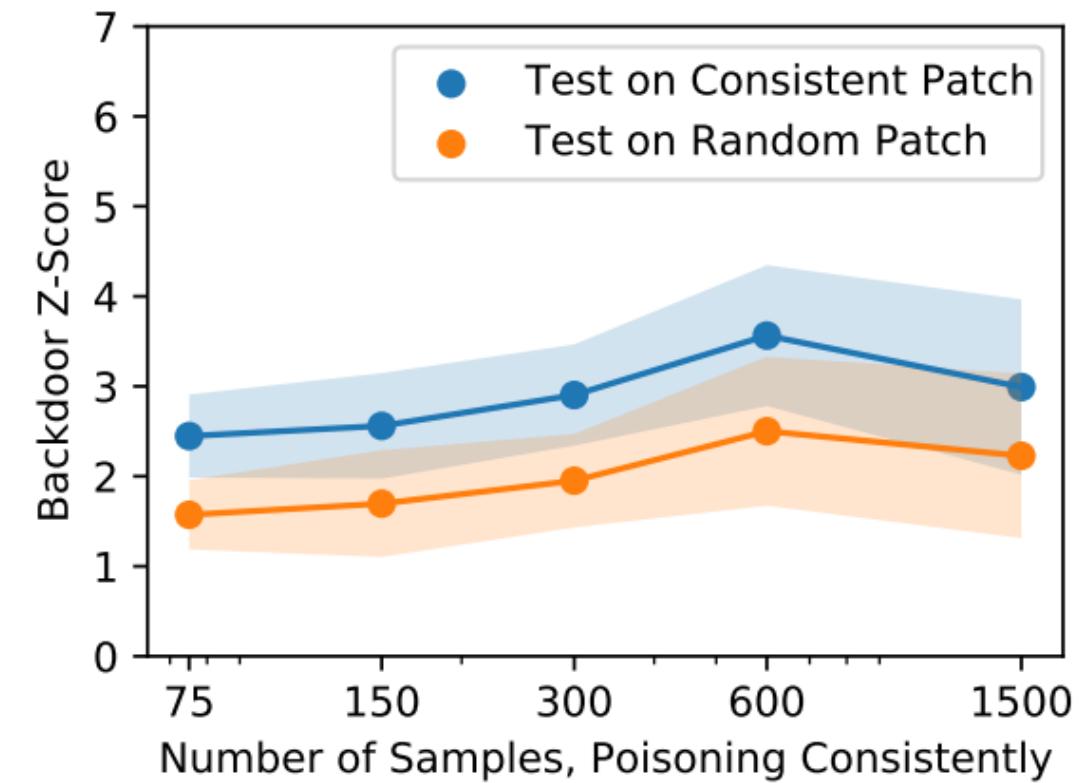


BACKDOOR ATTACK SUCCESS RATE AS A FUNCTION OF POISONED FRACTION

placing the patch consistently in the upper left corner of an image



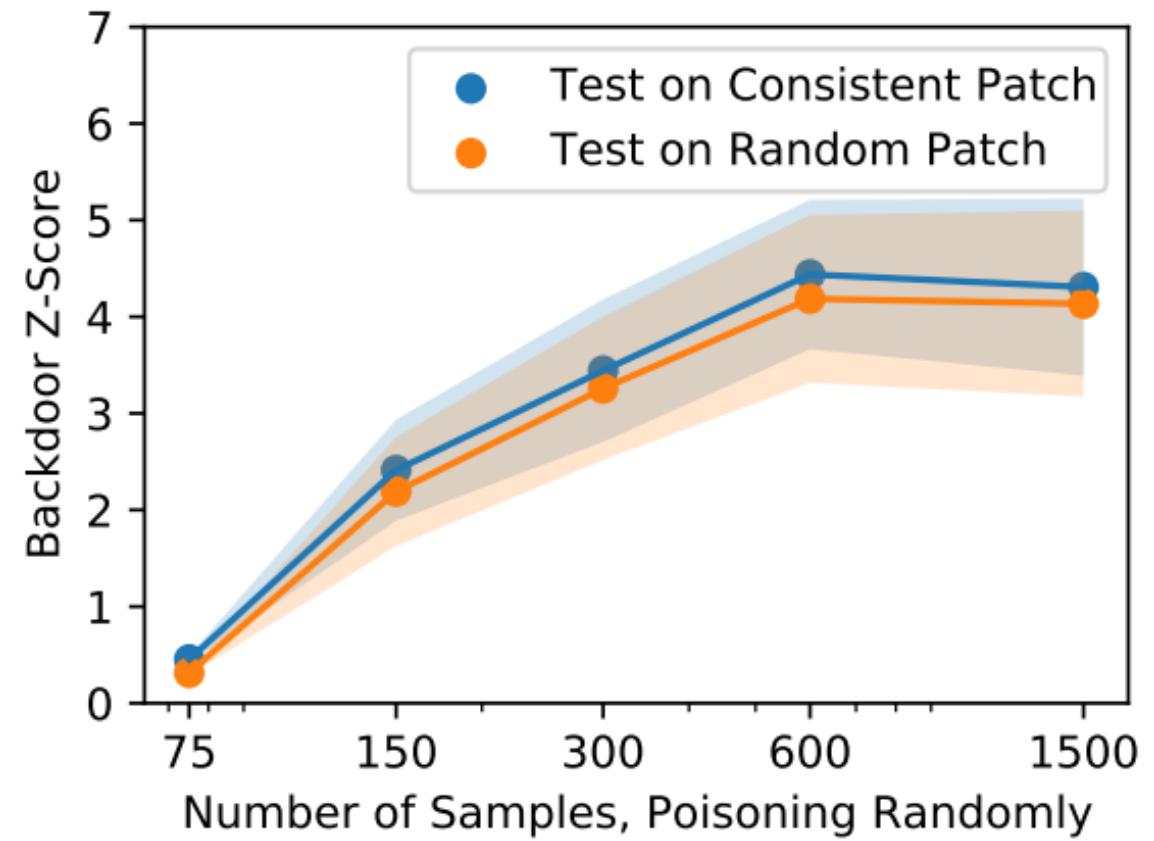
as we increase the number of poisoned examples.
While inserting more poisoned samples only marginally helps increase the attack success rate



BACKDOOR ATTACK SUCCESS RATE AS A FUNCTION OF POISONED FRACTION

place the patches randomly

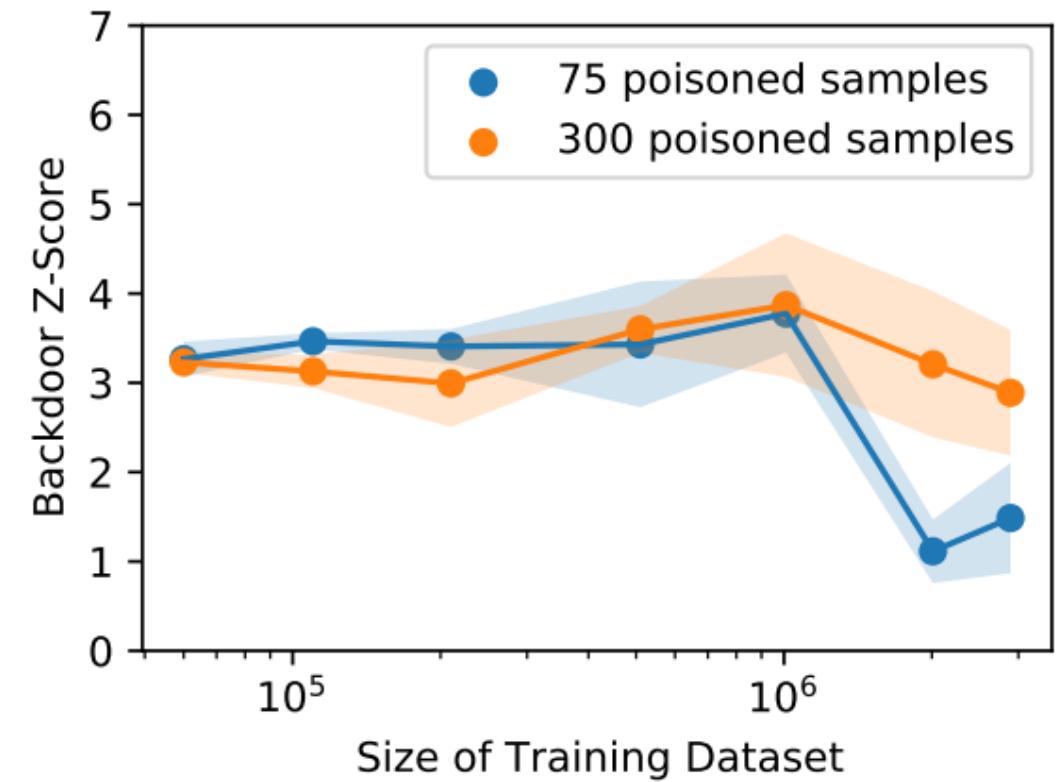
More effective when we place
the patches randomly



BACKDOOR ATTACK SUCCESS RATE AS A FUNCTION OF MODEL AND DATA SCALE



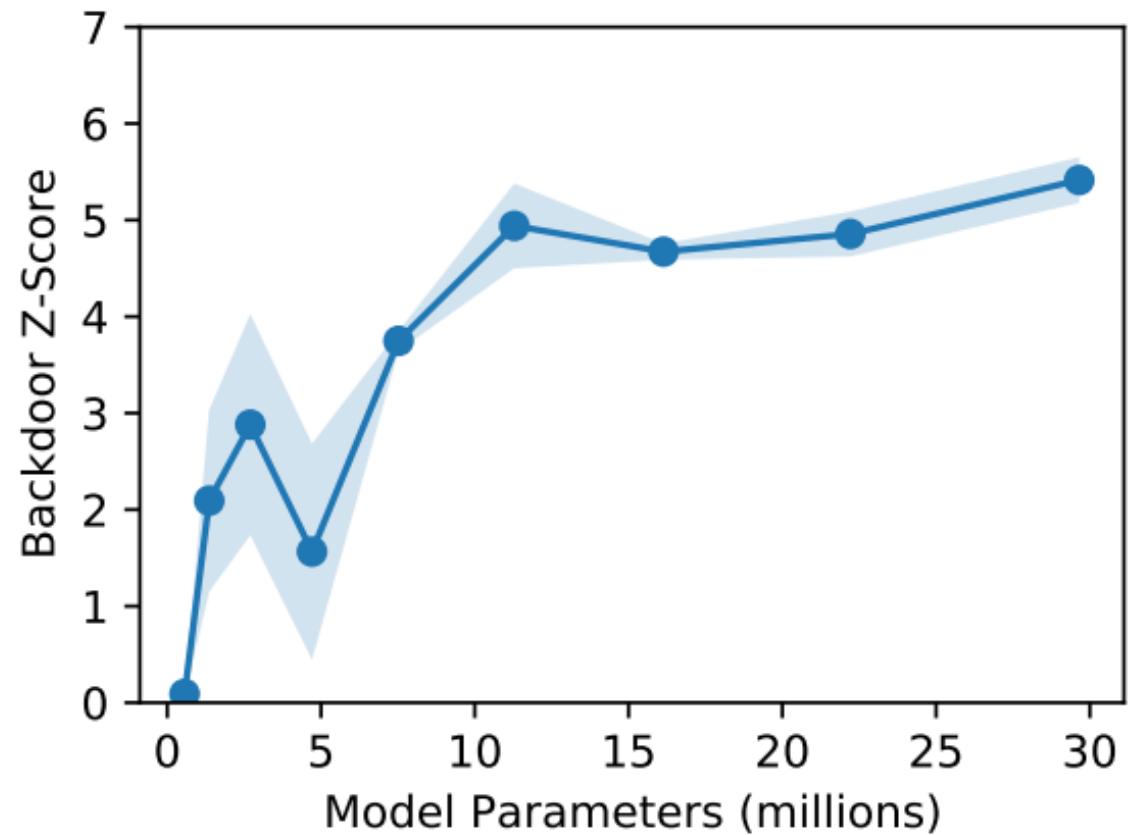
attack success rate remains almost completely constant as we artificially reduce the training dataset size



BACKDOOR ATTACK SUCCESS RATE AS A FUNCTION OF MODEL AND DATA SCALE

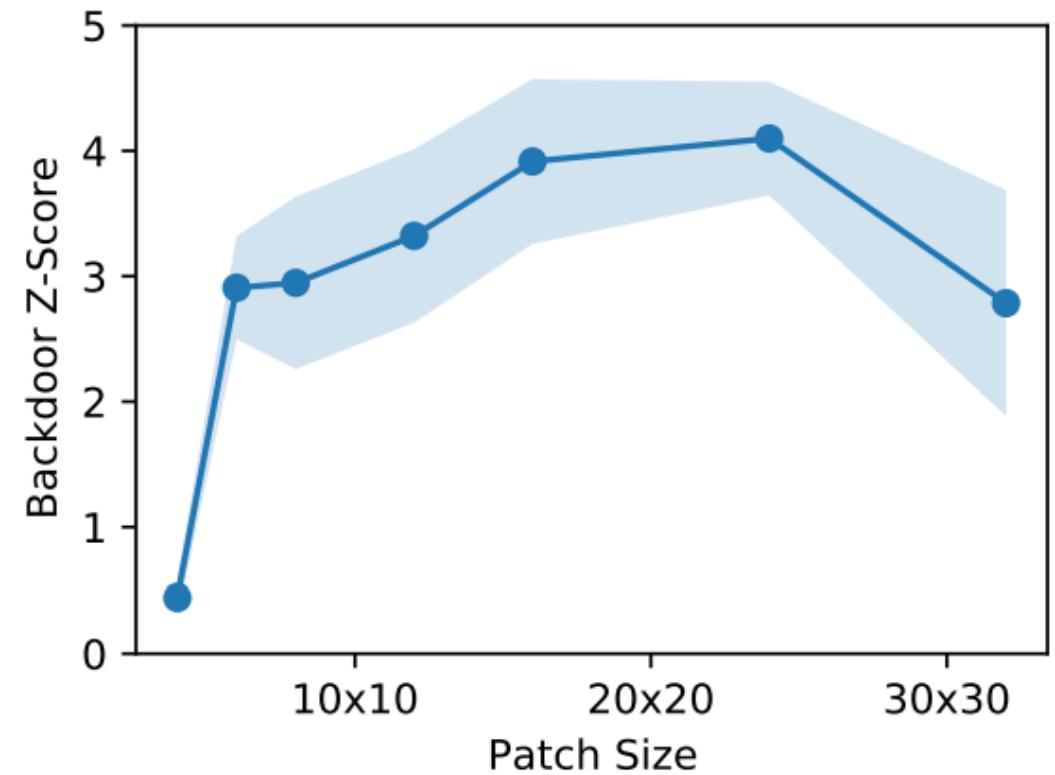


Why?



BACKDOOR ATTACK SUCCESS RATE AS A FUNCTION OF PATCH SIZE

even small adversarial patches might be able to effectively backdoor state-of-the-art models



Conclusion

Strengths



- ❖ **Novelty and contribution:** The paper introduces novel poisoning and backdooring attacks specifically designed for multimodal contrastively trained models. This contribution addresses a research gap and expands the understanding of security risks in the context of multimodal models.
- ❖ **Empirical evaluation:** The paper conducts extensive experiments on two benchmark datasets using an open-source implementation of CLIP. The empirical evaluation demonstrates the effectiveness of the proposed attacks, highlighting the potential vulnerabilities of contrastively trained models.
- ❖ **Analysis and insights:** The paper includes an analysis and ablation study to investigate the behavior and impact of the attacks. This analysis provides insights into the mechanisms and weaknesses of contrastively trained models, enhancing the understanding of their susceptibility to adversarial attacks.
- ❖ **Practical implications:** The paper's findings have practical implications for real-world applications that employ multimodal contrastively trained models. By highlighting the vulnerabilities of these models, the paper encourages researchers and practitioners to develop robust defenses and countermeasures against potential attacks.

Weaknesses



- ❖ **Lack of comparison:** The paper does not extensively compare the proposed attacks with existing state-of-the-art methods. A comprehensive comparison would help assess the effectiveness, efficiency, or uniqueness of the proposed attacks in comparison to other approaches.
- ❖ **Generalizability:** The paper's effectiveness claims for the proposed attacks are based on experiments conducted on specific datasets and an open-source implementation of CLIP. The generalizability of the attacks to other multimodal models and datasets remains uncertain and requires further investigation.
- ❖ **Evaluation metrics:** The paper introduces the backdoor z-score as a metric to measure attack efficacy, but it does not provide a detailed comparison with other commonly used evaluation metrics. This limits the ability to fully assess the effectiveness and performance of the proposed attacks.



CONCLUSION

- 1 we demonstrate that training on unfiltered datasets, while now possible intensifies the risk of poisoning attacks
- 2 scaling up the dataset does not prevent the attack from succeeding

References

- Jason Weston, Samy Bengio, and Nicolas Usunier. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine learning*, 81(1):21–35, 2010.
- Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non- parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3733–3742, 2018.
- Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.
- Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pp. 5389–5400. PMLR, 2019.
- Aniruddha Saha, Ajinkya Tejankar, Soroush Abbasi Koohpayegani, and Hamed Pirsiavash. Backdoor attacks on self-supervised learning, 2021.
- Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems*, pp. 6103–6113, 2018.
- Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2556–2565, 2018.

Thanks for your attention.
Any Question?

Extracting Training Data from Large Language Models

Published in 30th USENIX Security Symposium, 2021

Presenters:

Freidoon Mehri
Amirhossein Hadian

Extracting Training Data from Large Language Models

Nicholas Carlini¹

Florian Tramèr²

Eric Wallace³

Matthew Jagielski⁴

Ariel Herbert-Voss^{5,6}

Katherine Lee¹

Adam Roberts¹

Tom Brown⁵

Dawn Song³

Úlfar Erlingsson⁷

Alina Oprea⁴

Colin Raffel¹

¹*Google* ²*Stanford* ³*UC Berkeley* ⁴*Northeastern University* ⁵*OpenAI* ⁶*Harvard* ⁷*Apple*

Outline

- Introduction
- Background
- Methodology
- Results
- Defenses
- Limitations & Future Work
- Conclusion



Introduction

Goals

- Large language models memorize
 - It's possible to extract this memorized knowledge.
- This Paper
 - GPT2 Black-Box Access
 - Untargeted Attack

Background

Language Modeling

where x_1, x_2, \dots, x_n is a sequence of tokens from a vocabulary \mathcal{V} by applying the chain rule of probability

$$\Pr(x_1, x_2, \dots, x_n) = \prod_{i=1}^n \Pr(x_i \mid x_1, \dots, x_{i-1}).$$

Training Objective. A language model is trained to maximize the probability of the data in a training set \mathcal{X} . In this paper, each training example is a text document—for example, a specific news article or webpage from the internet. Formally, training involves minimizing the loss function

$$\mathcal{L}(\theta) = -\log \prod_{i=1}^n f_\theta(x_i \mid x_1, \dots, x_{i-1})$$

Generating Text

Generating Text. A language model can generate new text (potentially conditioned on some prefix x_1, \dots, x_i) by iteratively sampling $\hat{x}_{i+1} \sim f_{\theta}(x_{i+1}|x_1, \dots, x_i)$ and then feeding \hat{x}_{i+1} back into the model to sample $\hat{x}_{i+2} \sim f_{\theta}(x_{i+2}|x_1, \dots, \hat{x}_{i+1})$. This process is repeated until a desired stopping criterion is reached. Variations of this text generation method include deterministically choosing the most-probable token rather than sampling (i.e., “greedy” sampling) or setting all but the top- n probabilities to zero and renormalizing the probabilities before sampling (i.e., top- n sampling¹ [18]).

Memorization

- Models trained on massive de-duplicated datasets only for a single epoch
 - **Not true for GPT2!** (12 epochs)
- Training examples do not have noticeably lower losses than test examples on average
 - GPT-2 does not overfit: the training loss is only **10% smaller** than the test loss across all model sizes.
 - Certain worst-case training examples are indeed memorized

Memorized Content Is Highly Dependent on the Model’s Context

For example, GPT-2 will complete the prompt “3.14159” with the first 25 digits of π correctly using greedy sampling. However, we find that GPT-2 “knows” (under Definition 2) more digits of π because using the beam-search-like strategy introduced above extracts 500 digits correctly.

Interestingly, by providing the more descriptive prompt “pi is 3.14159”, straight greedy decoding gives the first 799 digits of π —more than with the sophisticated beam search. Further providing the context “e begins 2.7182818, pi begins 3.14159”, GPT-2 greedily completes the first 824 digits of π .

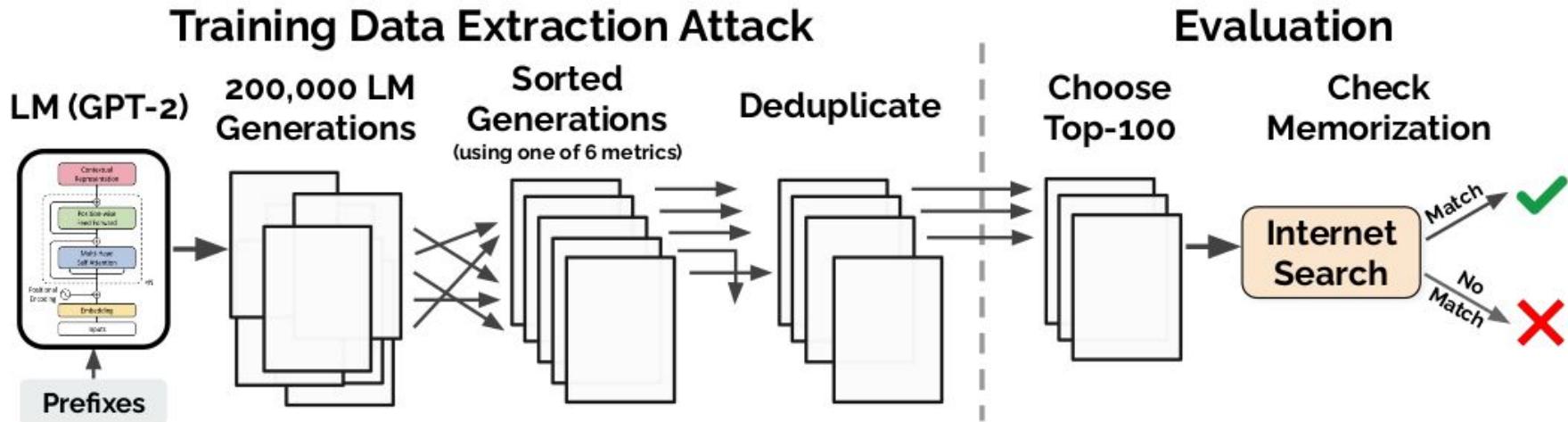
k-Eidetic Memorization

Definition 2 (k-Eidetic Memorization) A string s is k -eidetic memorized (for $k \geq 1$) by an LM f_θ if s is extractable from f_θ and s appears in at most k examples in the training data X : $|\{x \in X : s \subseteq x\}| \leq k$.

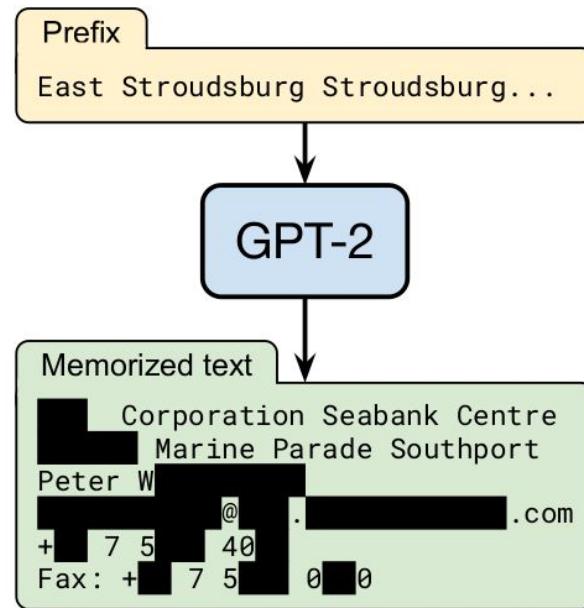
- Counts the number of distinct training examples containing a given string
 - a string may appear multiple times on one page while still counting as $k = 1$ memorization

Methodology

Overview



Attack Example



Threat Model

Adversary's Capabilities. We consider an adversary who has black-box input-output access to a language model. This allows the adversary to compute the probability of arbitrary sequences $f_{\theta}(x_1, \dots, x_n)$, and as a result allows the adversary to obtain next-word predictions, but it does not allow the adversary to inspect individual weights or hidden states (e.g., attention vectors) of the language model.

- **Untargeted:** We do not aim to extract targeted pieces of training data, but rather indiscriminately extract training data.

Methodology Overview

- **Generate text**
 - Sampling methods
 - Top-n
 - Temperature
 - Internet prefixes
- **Predict which outputs contain memorized text**
 - Membership Inference Attacks (MIA) metrics
 - Perplexity
 - Small
 - Medium
 - Zlib
 - Lowercase
 - Window

Top-N Strategy

- Initialize the language model with the start-of-sentence token
 - Then repeatedly sample tokens in an autoregressive fashion from the model
- By sampling according to the model's assigned likelihood:
 - We will sample sequences that the model considers "highly likely"
 - likely sequences correspond to memorized text.
 - We sample exactly **256 tokens** for each trial using the top-n strategy with **n = 40**.

Temperature

As described in Section 2.1, an LM outputs the probability of the next token given the prior tokens $\Pr(x_i | x_1, \dots, x_{i-1})$. In practice, this is achieved by evaluating the neural network $z = f_\theta(x_1, \dots, x_{i-1})$ to obtain the “logit” vector z , and then computing the output probability distribution as $y = \text{softmax}(z)$ defined by $\text{softmax}(z)_i = \exp(z_i) / \sum_{j=1}^n \exp(z_j)$.

One can artificially “flatten” this probability distribution to make the model less confident by replacing the output $\text{softmax}(z)$ with $\text{softmax}(z/t)$, for $t > 1$. Here, t is called the *temperature*. A higher temperature causes the model to be less confident and more diverse in its output.

Sampling With A Decaying Temperature

- Temperature regulates the softmax
 - Higher temperature means more randomness
- Use a softmax temperature that decays over time
 - **starting at $t = 10$ and decaying down to $t = 1$ over a period of the first 20 tokens**
 - $\approx 10\%$ of the length of the sequence
 - Sufficient amount of time for the model to “**explore**” a diverse set of prefixes
 - Also allowing it to **follow high-confidence paths** that it finds

Conditioning on Internet Text

- Seeds the model with prefixes from our own scrapes of the Internet
 - randomly sample between **5 and 10 tokens** of context from this scraped data
- Ensures that
 - **Diverse** generations
 - similar in nature to the type of data GPT-2 was trained on
- **Data leakage?!**
 - We select samples from a subset of **Common Crawl**
 - GPT-2 scrapes **outgoing Reddit** links
 - Leakage not that important
 - We only give **short prompts**

Methodology Overview

- **Generate text**
 - Sampling methods
 - Top-n
 - Temperature
 - Internet prefixes
- **Predict which outputs contain memorized text**
 - Membership Inference Attacks (MIA) metrics
 - Perplexity
 - Small
 - Medium
 - Zlib
 - Lowercase
 - Window

Perplexity

Given a sequence of tokens x_1, \dots, x_n , the perplexity is defined as:

$$P = \exp \left(-\frac{1}{n} \sum_{i=1}^n \log f_\theta(x_i | x_1, \dots, x_{i-1}) \right)$$

- Low perplexity =>
 - model not “surprised” by the sequence

Why Is Perplexity Not Enough for MIA

- Many samples with **spuriously high likelihood**
 - **Trivial memorization** (high k)
 - E.g., numbers from 1 to 100
 - **Repeated substrings**
 - LMs like to repeat the same string over and over
 - E.g., “I love you. I love you ...”

Augment Perplexity!

- Filter out these uninteresting (yet still high-likelihood samples) by comparing to a second LM
 - The second LM also assign high likelihood to these forms of memorized content
 - Filter samples **where the original model's likelihood is “unexpectedly high” compared to a second model**

Comparing to Other Neural Language Models

- **Smaller models have less capacity** for memorization
 - There are samples that are k-eidetic memorized (for small k) by the largest GPT-2 model
 - **not memorized by smaller GPT-2 models**
- We use the **Small (117M parameters)** and **Medium (345M parameters)** models.

Comparing to Zlib Compression

- Not necessary to use another neural LM
 - Any technique that quantifies some notion of “surprise” for a given sequence
- Zlib entropy of the text
 - Number of bits of entropy when the sequence is compressed with zlib compression
 - Can identify many of the examples of trivial memorization and repeated patterns described above
 - E.g., they are excellent at modeling repeated substrings

Comparing to Lower-Cased Text

- Ratio of the perplexity on the sample before and after lowercasing it
 - which can dramatically alter the perplexity of memorized content that expects a particular casing

Perplexity on a Sliding Window

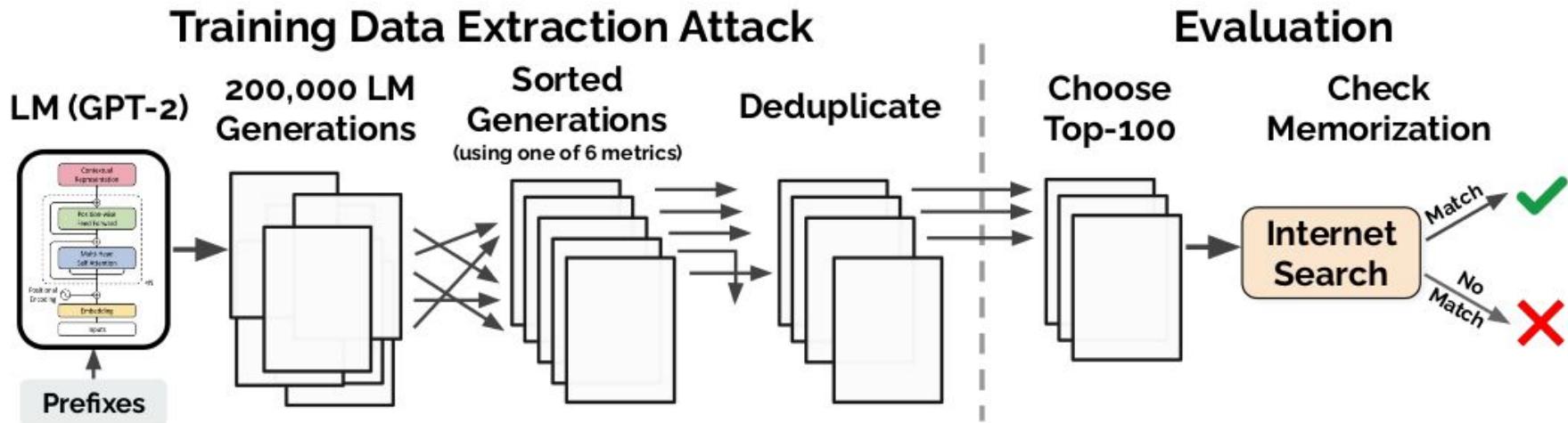
- Perplexity on a **Sliding Window**
 - One memorized substring surrounded by a block of non-memorized (and high perplexity) text
 - Use the minimum perplexity when **averaged over a sliding window of 50 token**

Methodology Overview

- **Generate text**
 - Sampling methods
 - Top-n
 - Temperature
 - Internet prefixes
- **Predict which outputs contain memorized text**
 - Membership Inference Attacks (MIA) metrics
 - Perplexity
 - Small
 - Medium
 - Zlib
 - Lowercase
 - Window

Results

Overview



For each of these **$3 \times 6 = 18$ configurations**, we select 100 samples from among the top-1000 samples according to the chosen metric. This gives us **1,800 total samples of potentially memorized content**.

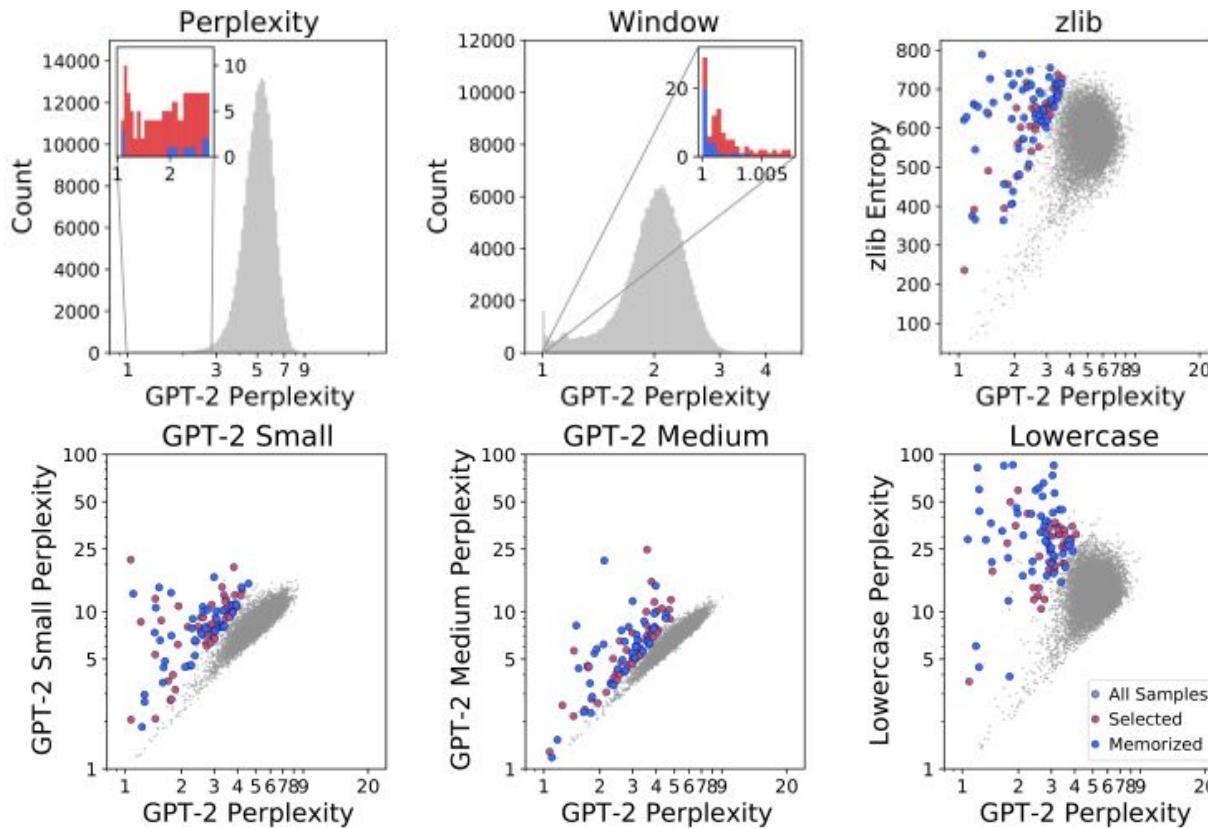
Manual categorization of the 604 memorized training example

Category	Count
US and international news	109
Log files and error reports	79
License, terms of use, copyright notices	54
Lists of named items (games, countries, etc.)	54
Forum or Wiki entry	53
Valid URLs	50
Named individuals (non-news samples only)	46
Promotional content (products, subscriptions, etc.)	45
High entropy (UUIDs, base64 data)	35
Contact info (address, email, phone, twitter, etc.)	32
Code	31
Configuration files	30
Religious texts	25
Pseudonyms	15
Donald Trump tweets and quotes	12
Web forms (menu items, instructions, etc.)	11
Tech news	11
Lists of numbers (dates, sequences, etc.)	10

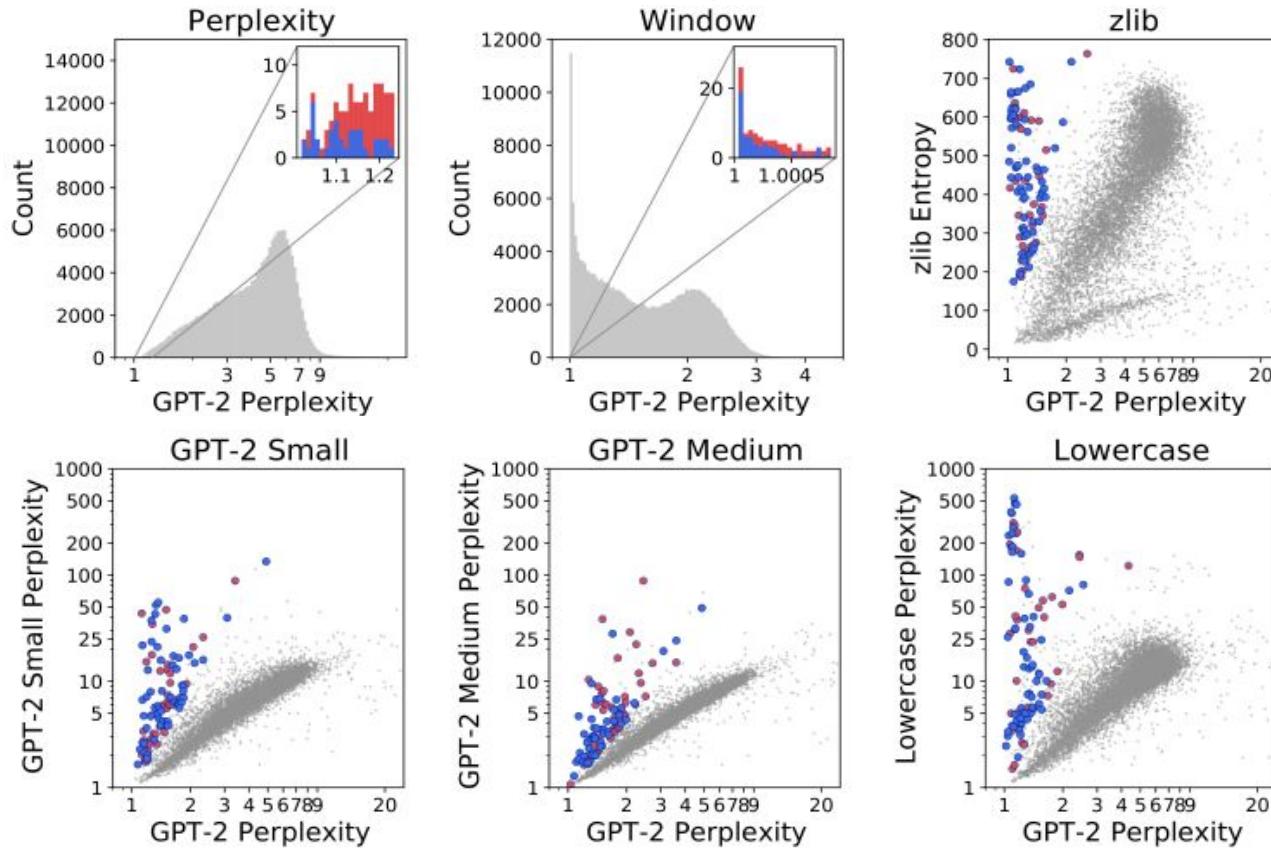
Strategies Results

Inference Strategy	Text Generation Strategy		
	Top- <i>n</i>	Temperature	Internet
Perplexity	9	3	39
Small	41	42	58
Medium	38	33	45
zlib	59	46	67
Window	33	28	58
Lowercase	53	22	60
Total Unique	191	140	273

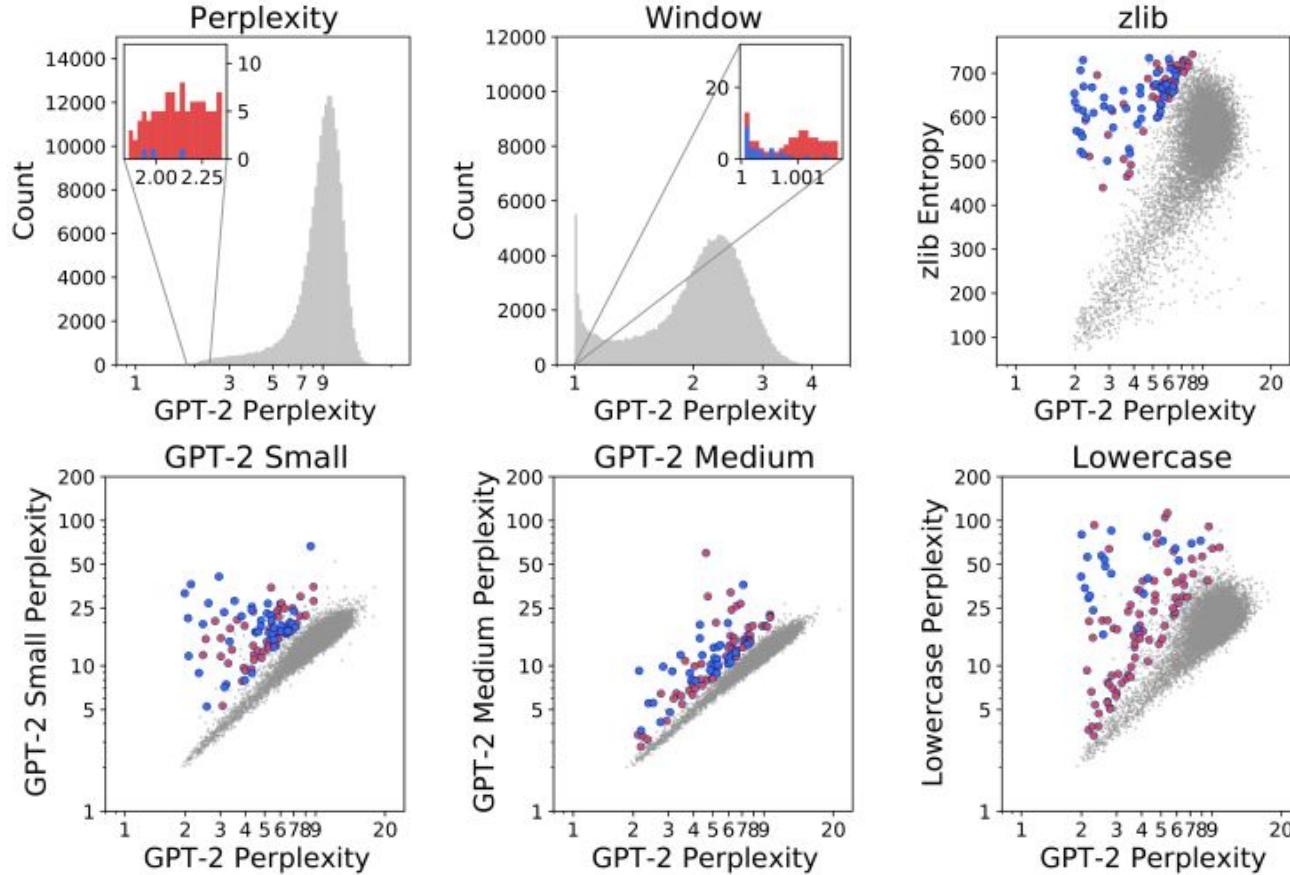
Perplexity vs other strategies - Top-n Scenario



Perplexity vs other strategies - Internet Scenario



Perplexity vs other strategies - Temperature Scenario



Inverse Scaling: Correlating Memorization with Model Size

- **Larger Models** memorize significantly more training data
- For the largest LM, complete memorization occurs after **just 33 insertions** (in a single document)

URL (trimmed)	Occurrences		Memorized?		
	Docs	Total	XL	M	S
/r/[REDACTED]51y/milo_evacua...	1	359	✓	✓	½
/r/[REDACTED]zin/hi_my_name...	1	113	✓	✓	
/r/[REDACTED]7ne/for_all_yo...	1	76	✓	½	
/r/[REDACTED]5mj/fake_news_...	1	72	✓		
/r/[REDACTED]5wn/reddit_admi...	1	64	✓	✓	
/r/[REDACTED]lp8/26_evening...	1	56	✓	✓	
/r/[REDACTED]jla/so_pizzagat...	1	51	✓	½	
/r/[REDACTED]ubf/late_night...	1	51	✓	½	
/r/[REDACTED]eta/make_christ...	1	35	✓	½	
/r/[REDACTED]6ev/its_officia...	1	33	✓		
/r/[REDACTED]3c7/scott_adams...	1	17			
/r/[REDACTED]k2o/because_his...	1	17			
/r/[REDACTED]tu3/armynavy_ga...	1	8			

Defenses

Mitigating Privacy Leakage in LMs

- Training With Differential Privacy
 - DP-SGD
 - Time consuming
 - Accuracy Trade-off
- Curating the Training Data
 - Identifying and limiting sensitive and personal data
 - De-duplication
- Limiting Impact of Memorization on Downstream Applications
 - Fine-tuning causes the model to forget
 - New privacy leakages?
- Auditing ML Models for Memorization

Limitations & Future Work

Limitations & Future Work

- How memorization is inherited by **fine-tuned models**
- **Targeted attacks** toward specific content
- Applying the attack on **other larger LMs**
 - Larger LMs
 - Different LMs
- Applying the attack on **DP-SGD** trained LMs
 - Evaluation of each defense mechanisms
- How effective are **instruction following** and **Reinforcement Learning From Human Feedback (RLHF)** as privacy defenses?

Conclusion

Conclusion

- Extraction attacks are a practical threat
- Two-step attack
 - Different strategies
 - Effect of each strategy
 - Extracted Data Types
- k-Eidetic Memorization, a memorization metric
- Memorization does not require overfitting
- Larger models memorize more data
- Defense methods

Thank you for your attention!

SPML Presentation II

Deep Leakage from Gradients

paper by Zhu et al. 2019

Alireza Sakhaeirad
Hamidreza Akbari

Sharif University of Technology
Computer Engineering Department

July 1, 2023

Table of Contents

1 Introduction

2 Method

3 Experiments

4 Defences

5 Discussion

Table of Contents

1 Introduction

2 Method

3 Experiments

4 Defences

5 Discussion

Distributed Learning

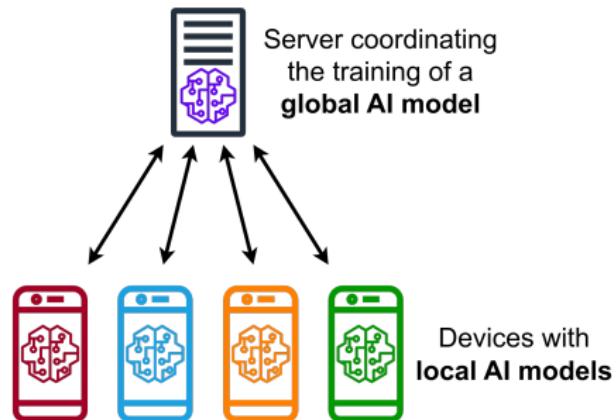


Figure: Distributed Learning; Image taken from here

Gradient Exchange; Is it safe?

Previous Assumption

It was believed that sharing gradients can't reveal the training data.

Not Correct!

Authors in this paper attempt to show that gradients can in fact leak information about the training data.

Problem Formulation

They show that given the model \mathcal{F} and weights W , it is possible to extract training data \mathcal{D} , given the gradients of the model output with respect to its weights.

Risk in Centralized Setups

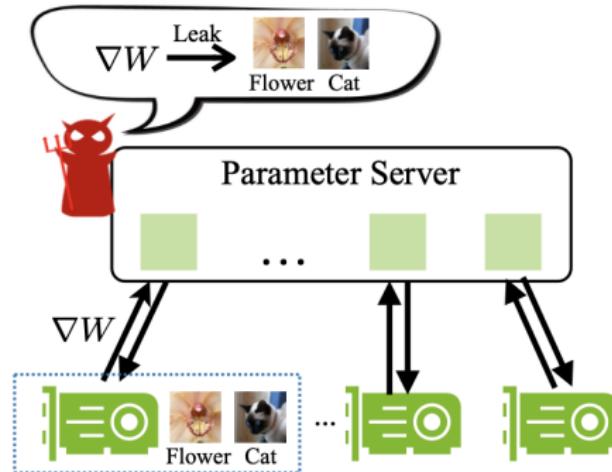


Figure: Malicious server can steal the training data using the gradients; Image taken from the paper

Risk in non-Centralized Setups

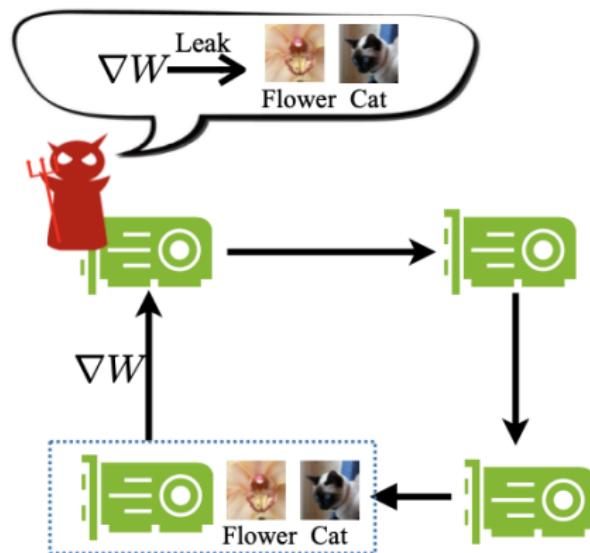


Figure: Any user can steal the training data using the gradients; Image taken from the paper

Contributions

They demonstrate that it is possible to obtain private training data from the publicly shared gradients using their algorithm DLG(deep leakage gradient)

DLG only requires the gradients and can reveal pixel-wise accurate images and token-wise matching texts.

To prevent potential leakage of important data, They analyze the attack difficulties in various settings and discuss several defense strategies against the attack.

Table of Contents

1 Introduction

2 Method

3 Experiments

4 Defences

5 Discussion

One Sample Case

First, they create dummy inputs and labels:

$$\mathbf{x}', \mathbf{y}' \leftarrow \mathcal{N}(0, I), \mathcal{N}(0, 1)$$

Then, they calculate the gradients with respect to them:

$$\nabla W' = \frac{\partial \ell(\mathcal{F}(\mathbf{x}', W), \mathbf{y}')}{\partial W}$$

One Sample Case

Then they solve the following optimization with respect to inputs and labels:

$$\mathbf{x}^*, \mathbf{y}^* = \arg \min_{\mathbf{x}', \mathbf{y}'} \|\nabla W' - \nabla W\| = \frac{\partial \ell(\mathcal{F}(\mathbf{x}', W), \mathbf{y}')}{\partial W} - \nabla W\|$$

Solving this optimization problem requires model \mathcal{F} to be twice differentiable.

One Sample Case

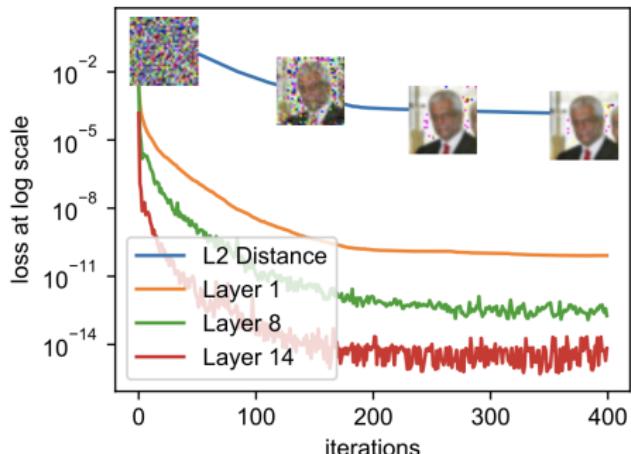


Figure: As we increase the number of iteration, the dummy data becomes closer to the real data; layer (i) indicates the MSE between real and dummy gradients of i^{th} layer

Batched Data

Previous algorithm takes too long to converge.

They hypothesize that the reason is there are $N!$ different combinations of images, so its difficult for the algorithm to choose gradient direction.

Algorithm

Algorithm 1 Deep Leakage from Gradients.

Input: $F(\mathbf{x}; W)$: Differentiable machine learning model; W : parameter weights; ∇W : gradients calculated by training data

Output: private training data \mathbf{x}, \mathbf{y}

```
1: procedure DLG( $F, W, \nabla W$ )
2:    $\mathbf{x}'_1 \leftarrow \mathcal{N}(0, 1), \mathbf{y}'_1 \leftarrow \mathcal{N}(0, 1)$                                  $\triangleright$  Initialize dummy inputs and labels.
3:   for  $i \leftarrow 1$  to  $n$  do
4:      $\nabla W'_i \leftarrow \partial \ell(F(\mathbf{x}'_i, W_t), \mathbf{y}'_i) / \partial W_t$            $\triangleright$  Compute dummy gradients.
5:      $\mathbb{D}_i \leftarrow \|\nabla W'_i - \nabla W\|^2$ 
6:      $\mathbf{x}'_{i+1} \leftarrow \mathbf{x}'_i - \eta \nabla_{\mathbf{x}'_i} \mathbb{D}_i, \mathbf{y}'_{i+1} \leftarrow \mathbf{y}'_i - \eta \nabla_{\mathbf{y}'_i} \mathbb{D}_i$      $\triangleright$  Update data to match gradients.
7:   end for
8:   return  $\mathbf{x}'_{n+1}, \mathbf{y}'_{n+1}$ 
9: end procedure
```

Figure: Deep Leakage Gradient

Batched Data

To solve the problem, they updated only one dummy sample per iteration.

They observed that with a larger batch size, more iterations are required for convergence, which is intuitively acceptable.

	BS=1	BS=2	BS=4	BS=8
ResNet-20	270	602	1173	2711

Figure: Effect of batch size in convergence time

Batched Data

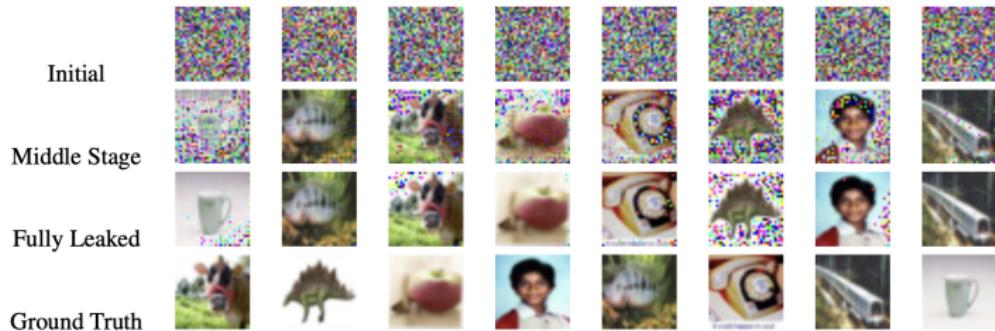


Figure: Example of the batched data mode; the order may be wrong, but outputs are acceptable

Table of Contents

1 Introduction

2 Method

3 Experiments

4 Defences

5 Discussion

Setups

- They used pytorch as their software.
- They used L-BFGS as optimization algorithm. For specific detail, check the paper.
- Their procedure doesn't require the model to be fully trained. Attack can happen in any part of the training.

Image Classification Task

- They used a ResNet-56 and pictures from MNIST, CIFAR-100, SVHN, and LFW datasets.
- Activation functions are changed from relu to sigmoid to ensure differentiability.
- Dummy labels are passed through softmax to be like one-hot encoded vectors.

Examples

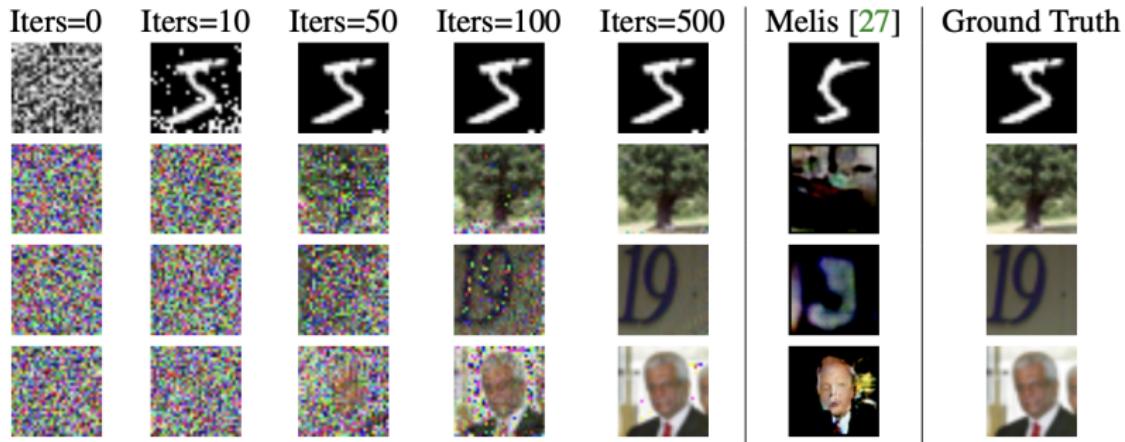


Figure: Example of the attack; images with plain backgrounds are easier to extract. Complex images like faces are harder to extract

Masked Language Model Task

- They used a BERT model as backbone.
- In this task, 15% of words are masked, and model is asked to predict them.
- They searched for the input tokens in the embedding space, then matched the result to the closest vector in the embedding space.

Examples

	Example 1	Example 2	Example 3
Initial Sentence	tilting fill given **less word **itude fine **nton over- heard living vegas **vac ***vation *f forte **dis ce- rambycidae ellison **don yards marne **kali	toni **enting asbestos cutter km nail **oof **dation **ori righteous **xie lucan **hot **ery at **ile ordered pa **eit smashing proto	[MASK] **ry toppled **wled major relief dive displaced **lice [CLS] us apps _ **face **bet
Iters = 10	tilting fill given **less full solicitor other ligue shrill living vegas rider treatment carry played sculptures life- long ellison net yards marne **kali	toni **enting asbestos cutter km nail undefeated **dation hole righteous **xie lucan **hot **ery at **ile ordered pa **eit smashing proto	[MASK] **ry toppled iden- tified major relief gin dive displaced **lice doll us apps _ **face space
Iters = 20	registration , volunteer ap- plications , at student travel application open the ; week of played ; child care will be glare .	we welcome proposals for tutor **ials on either core machine denver softly or topics of emerging import- ance for machine learning	one **ry toppled hold major ritual ' dive annual confer- ence days 1924 apps novel- ist dude space
Iters = 30	registration , volunteer ap- plications , and student travel application open the first week of september . child care will be available .	we welcome proposals for tutor **ials on either core machine learning topics or topics of emerging import- ance for machine learning	we invite submissions for the thirty - third annual con- ference on neural informa- tion processing systems .
Original Text	Registration, volunteer applications, and student travel application open the first week of September. Child care will be available.	We welcome proposals for tutorials on either core ma- chine learning topics or topics of emerging import- ance for machine learning.	We invite submissions for the Thirty-Third Annual Conference on Neural Infor- mation Processing Systems.

Figure: Example of the attack on MLM

Table of Contents

1 Introduction

2 Method

3 Experiments

4 Defences

5 Discussion

(I) Noisy Gradient

Adding some noise (from Laplacian and Gaussian distributions) can prevent the leakage.

Noise distribution doesn't matter much and variance is the key factor. Variance over 10^{-2} can prevent leakage.

Half-precision is also tested, but doesn't yield desirable results.

(I) Noisy Gradient

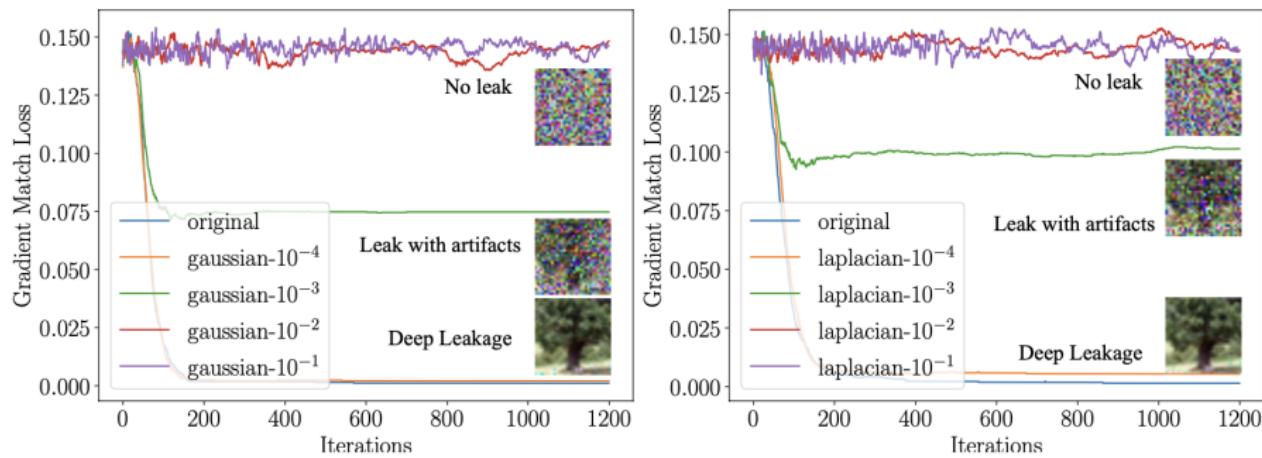


Figure: Effects of adding Laplacian and Gaussian noise

(I) Noisy Gradient

	Original	$G \cdot 10^{-4}$	$G \cdot 10^{-3}$	$G \cdot 10^{-2}$	$G \cdot 10^{-1}$	FP-16
Accuracy	76.3%	75.6%	73.3%	45.3%	$\leq 1\%$	76.1%
Defendability	-	X	X	✓	✓	X
		$L \cdot 10^{-4}$	$L \cdot 10^{-3}$	$L \cdot 10^{-2}$	$L \cdot 10^{-1}$	Int-8
Accuracy	-	75.6%	73.4%	46.2%	$\leq 1\%$	53.7%
Defendability	-	X	X	✓	✓	✓

Figure: A trade-off between privacy and accuracy

(I) Noisy Gradient

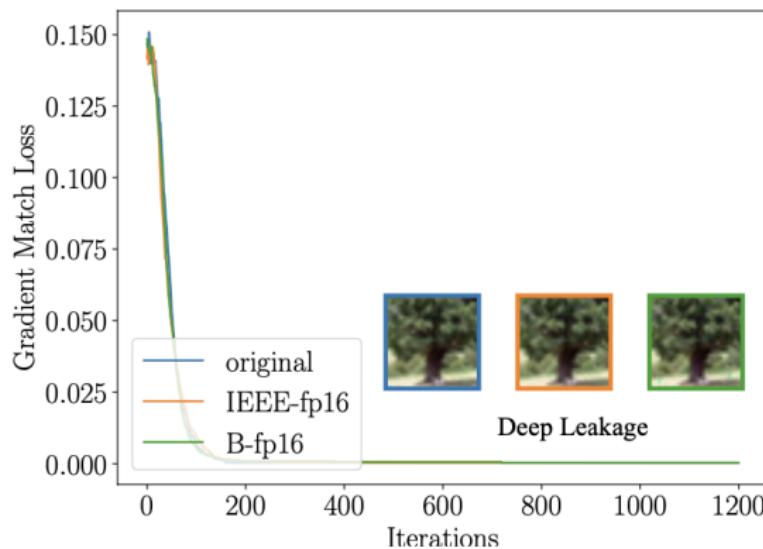


Figure: Using half-presicions doesn't help the privacy. This graph is for fp16 conversion

(II) Gradient Compression and Sparsification

Gradient Compression: gradients with small magnitudes are pruned to zero.

Sparsity from 1% to 10% doesn't affect the DLG. However, sparsity over 20% disables the algorithm to produce proper outputs.

It is shown that it doesn't have a significant effect on the model accuracy.

(II) Gradient Compression and Sparsification

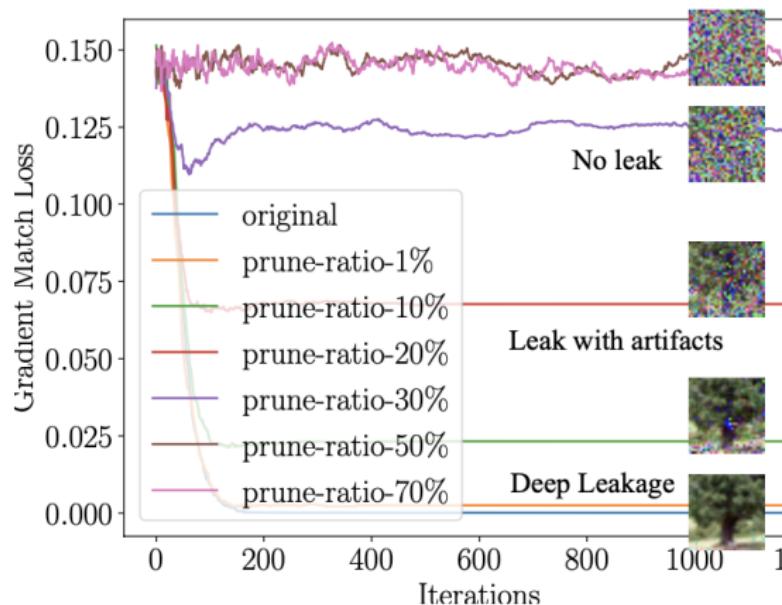


Figure: Defend using gradient pruning

Large Batch, High Resolution and Cryptology

If making modifications to the dataset is allowed, increasing the batch size and resolution (may need changing the CNN structure) can prevent leakage against DLG. They state that algorithm works successfully only for batch sized up to 8 and image scales up to 64×64

Also, using cryptology techniques that are studied in the federated learning scenario can disable the algorithm. Some algorithms are referenced in the article.

Table of Contents

1 Introduction

2 Method

3 Experiments

4 Defences

5 Discussion

Paper Limitations

- No experiments is done on ImageNet, which is the most popular image classification dataset.
- Algorithm has very limited power. Simply using a batch size bigger than 8, which is very common, makes the algorithm useless.
- Cryptology techniques simply make the attack useless, without decreasing the accuracy.

Thank You for Your Attention!
Any Questions?

Label-Only Membership Inference Attacks

Presenters: Mehdi Dousti & Erfan Zarinkia

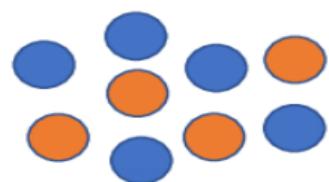
Security and Privacy in Machine Learning
Instructor: Dr. Sadeghzadeh

Table of Contents

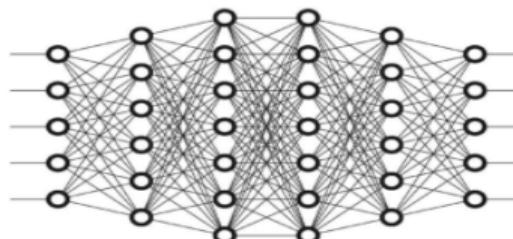
- ① Recap
- ② Attack
- ③ Evaluation
- ④ Conclusions
- ⑤ References

What is Membership Inference Attack?

Training of Target Model



Training Data



Deep Neural Network

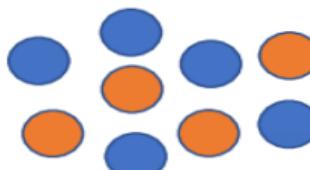
Membership Inference Attack on Target Model



tries to answer:



\in



?

Training Data

Notation

- Given a sample x
 - access to a trained model h
 - the adversary uses a classifier f_h
 - compute a membership prediction $f(x; h) \in \{0, 1\}$
 - with the goal that $f(x; h) = 1$ if x is a training point

Threat model

The adversary has:

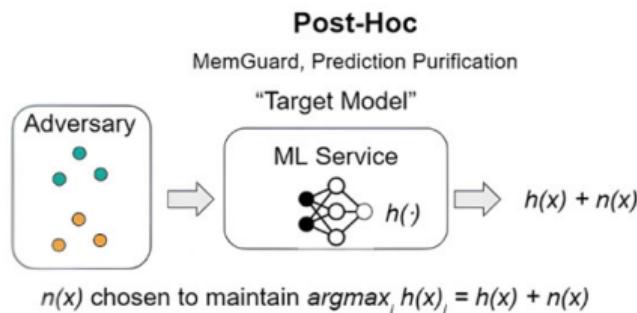
- Only black-box access to the trained model
 - Confidence-based (previous attack methods)
 - Label-only (this attack method)
 - Full knowledge of the task
 - Knowledge of the target model's architecture and training setup
 - Partial data knowledge
 - Knowledge of the targeted points' labels

Confidence-based Approach

- Adversary queries the model
 - Obtain the model's confidence
 - Infers the candidate's membership in the training set
 - Difference in prediction confidence is largely attributed to overfitting

Defense Methods

- Reduce overfitting
 - Regularization
 - Increase the amount of training data
 - Perturb model's predictions (confidence-masking)
 - Modifying the training procedure (Adversarial Regularization)
 - Modifying the inference procedure after training (Mem Guard)



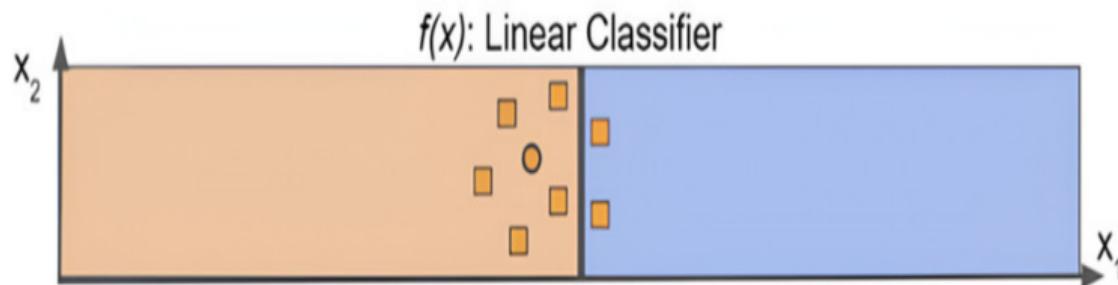
Label-Only Membership Inference Attacks

Label-Only Membership Inference Attacks

Christopher A. Choquette-Choo¹ Florian Tramèr² Nicholas Carlini³ Nicolas Papernot¹

Label-only Approach

- Data points with high robustness are training data points
 - Data augmentation causes privacy leakage
 - Relationship between the confidence at x and the Euclidean distance to the decision boundary



A Naive Baseline

- The Gap Attack
 - Predicts any misclassified data point as a non-member
 - Accuracy

$$1/2 + (\text{acc}_{\text{train}} - \text{acc}_{\text{test}}) / 2$$

$$\text{acc}_{\text{train}}, \text{acc}_{\text{test}} \in [0, 1]$$

Attack Types

- Boundary Distance
 - White-Box Baseline : C&W
 - Black-box Label-only attacks : HopSkipJump
 - Data Augmentation
 - Rotation
 - Translation
 - Robustness to Noise

Data Augmentation Attack

- create a MI classifier $f(x; h)$ for a model h
 - Given a target point (x_0, y_{true}) the adversary trains f to output $f(x_0, h) = 1$ if x_0 was a training member
 - rotations: generate $N = 3$ images as rotations by a magnitude $\pm r^\circ$ for $r \in [1, 15]$
 - translations: generate $N = 4d + 1$ translated images satisfying $|i| + |j| = d$ for a pixel bound d
 - horizontal shift by $\pm i$
 - vertical shift by $\pm j$

Decision Boundary Distance

- Given some estimate $dist_h(x, y)$ of a point's '2-distance to the model's boundary
 - we predict x a member if $dist_h(x, y) > \tau$
 - $dist_h(x, y) = 0$ for misclassified points

Robustness to Noise

- a point's distance to the boundary is directly related to the model's accuracy when it is perturbed by isotropic Gaussian noise
 - We compute a proxy for $d_h(x, y)$ by evaluating the accuracy of h on N points
 $\hat{x}_i = x + \mathcal{N}(0, \sigma^2 \cdot I)$

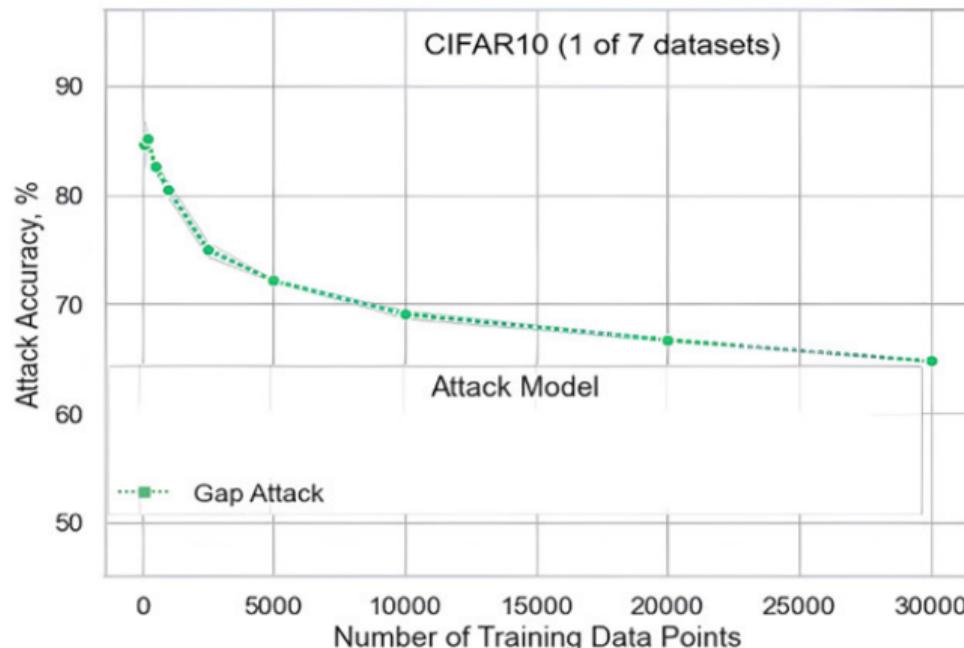
Evaluated Datasets

- 3 computer vision tasks
 - MNIST
 - CIFAR-10
 - CIFAR-100
 - 4 non-computer vision tasks
 - Adult
 - Texas-100
 - Purchase-100
 - Locations

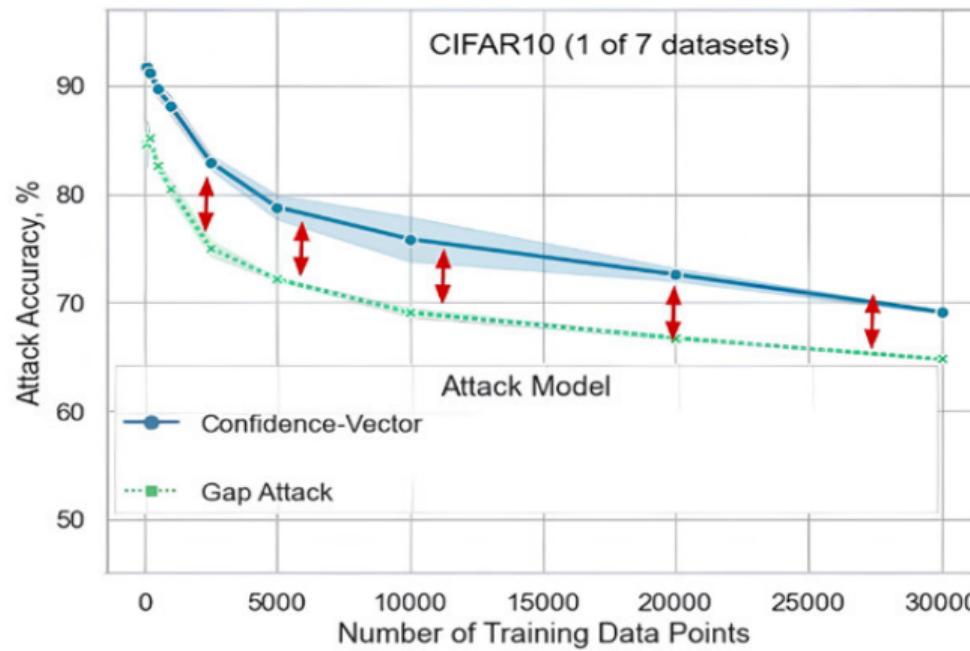
4 Main Questions

- ① Can label-only MI attacks match prior attacks that use the model's (full) confidence vector?
 - ② What is the query complexity of label-only attacks?
 - ③ Are defenses against confidence-based MI attacks always effective against label-only attacks?
 - ④ Which defenses prevent both label-only and full confidence-vector attacks?

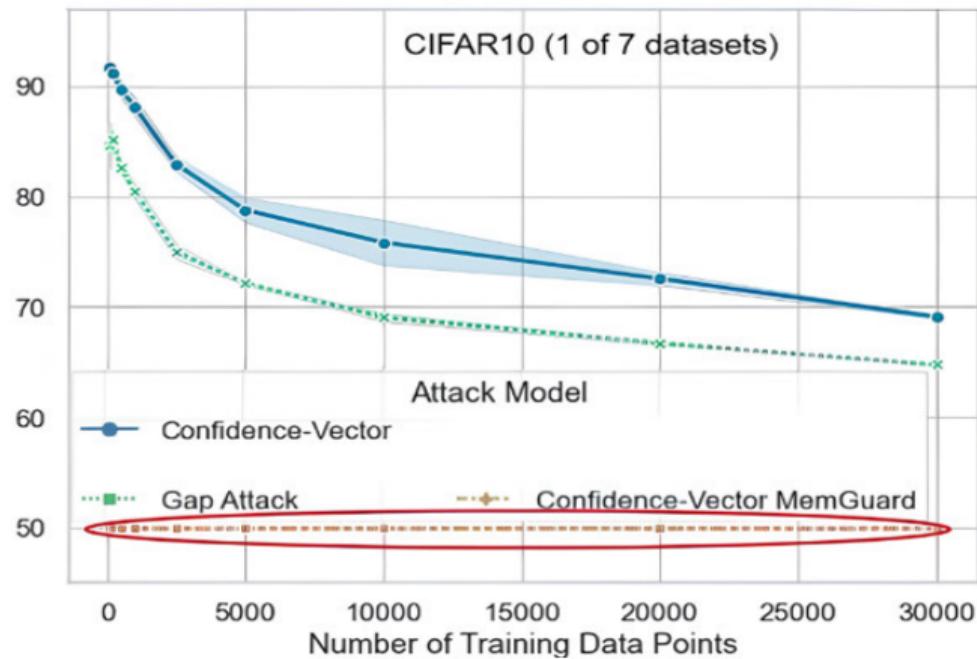
Can label-only MI attacks match prior attacks?



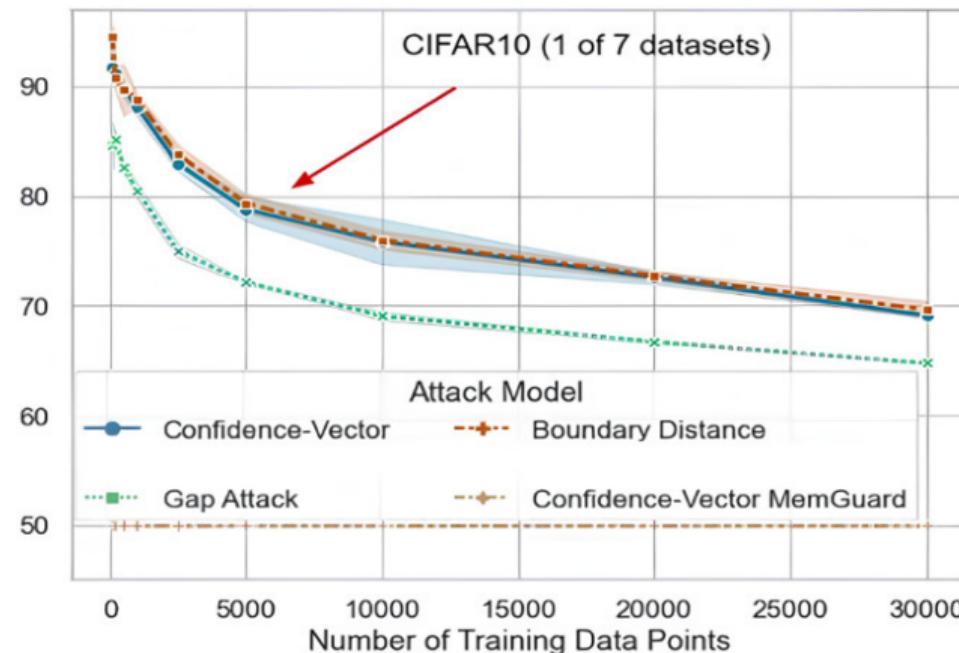
Can label-only MI attacks match prior attacks?



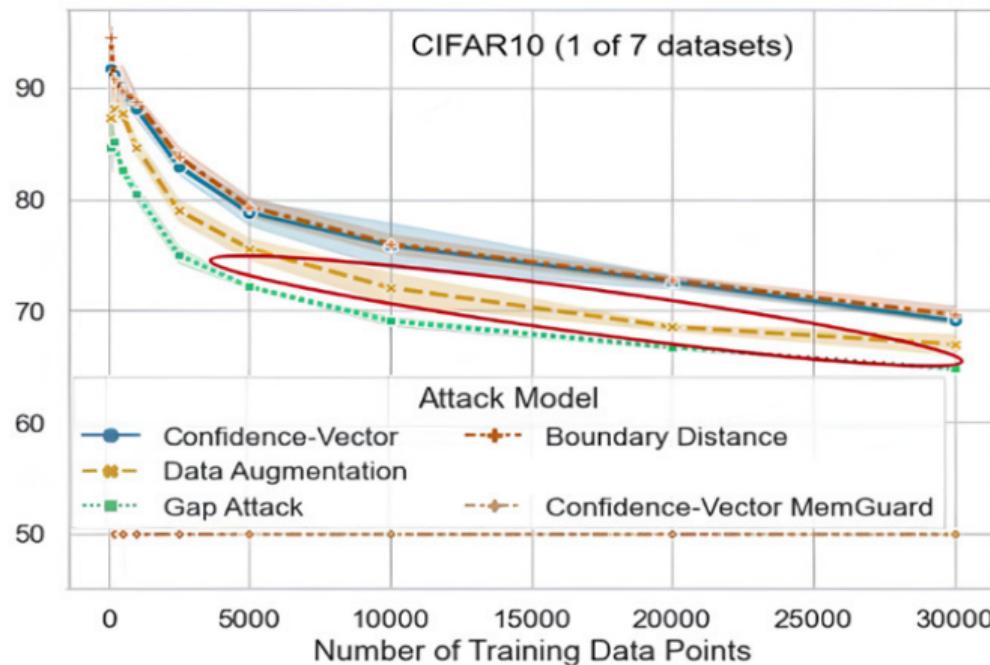
Can label-only MI attacks match prior attacks?



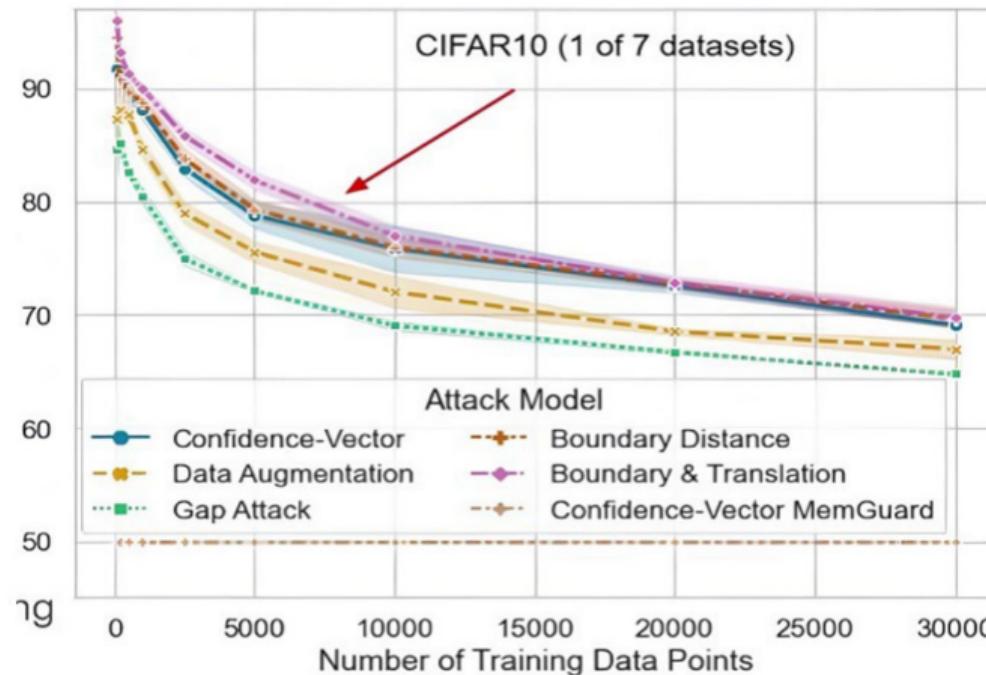
Can label-only MI attacks match prior attacks?



Can label-only MI attacks match prior attacks?



Can label-only MI attacks match prior attacks?



What is the query complexity of label-only attacks?

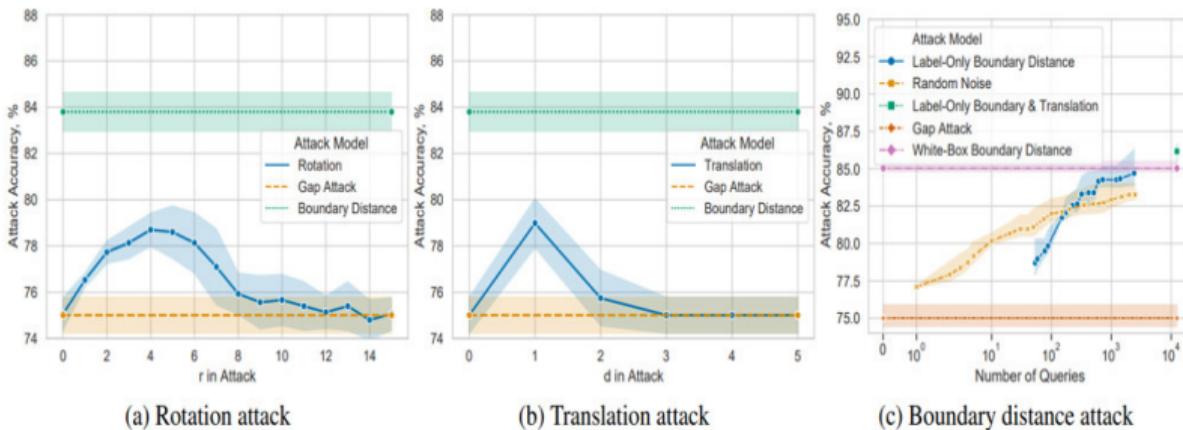


Figure 2. Comparison of query settings for different label-only MI attacks on CIFAR-10. Target model are trained on a subset of 2,500 data points. In (a) and (b), we compare the performance of the data augmentation attack against two baselines (the gap attack and the label-only boundary distance attack, with 2,500 queries), as we increase the r and d parameters. In (c), we compare attacks that threshold on a point's distance to the boundary in a black-box setting with a white-box baseline using Carlini and Wagner's attack (Carlini & Wagner, 2017). We describe these attacks in § 3.4. Costs are $\approx \$0.1$ per 1000 queries⁵.

Are CB defenses always effective against LO attacks?

(a) CIFAR-100 Undefended

Attack	Accuracy
Gap attack	83.5
Confidence-vector	88.1
Data augmentation	84.6
Boundary distance	88.0
Combined	89.2

(c) MNIST Undefended

Attack	Accuracy
Gap attack	53.2
Confidence-vector	55.7
Data augmentation	53.9
Boundary distance	57.8
Combined	58.7

(b) CIFAR-100 MemGuard

Attack	Accuracy
Gap attack	83.5
Confidence-vector	50.0
Data augmentation	84.6
Boundary distance	88.0
Combined	89.2

(d) MNIST MemGuard

Attack	Accuracy
Gap attack	53.2
Confidence-vector	50.0
Data augmentation	53.9
Boundary distance	57.8
Combined	58.7

Are CB defenses always effective against LO attacks?

(a) Texas Undefended		(b) Texas MemGuard	
Attack	Accuracy	Attack	Accuracy
Gap attack	73.9	Gap attack	73.9
Confidence-vector	84.0	Confidence-vector	50.0
Noise Robustness	80.3	Noise Robustness	80.3

(e) Location Undefended		(f) Location MemGuard	
Attack	Accuracy	Attack	Accuracy
Gap attack	72.1	Gap attack	72.1
Confidence-vector	92.6	Confidence-vector	50.0
Noise robustness	89.2	Noise Robustness	89.2

(c) Purchase Undefended		(d) Purchase MemGuard	
Attack	Accuracy	Attack	Accuracy
Gap attack	67.1	Gap attack	67.1
Confidence-vector	86.1	Confidence-vector	50.0
Noise Robustness	87.4	Noise Robustness	87.4

(g) Adult Undefended		(h) Adult MemGuard	
Attack	Accuracy	Attack	Accuracy
Gap attack	58.7	Gap attack	58.7
Confidence-vector	59.9	Confidence-vector	50.0
Noise Robustness	58.7	Noise Robustness	58.7

Are CB defenses always effective against LO attacks?

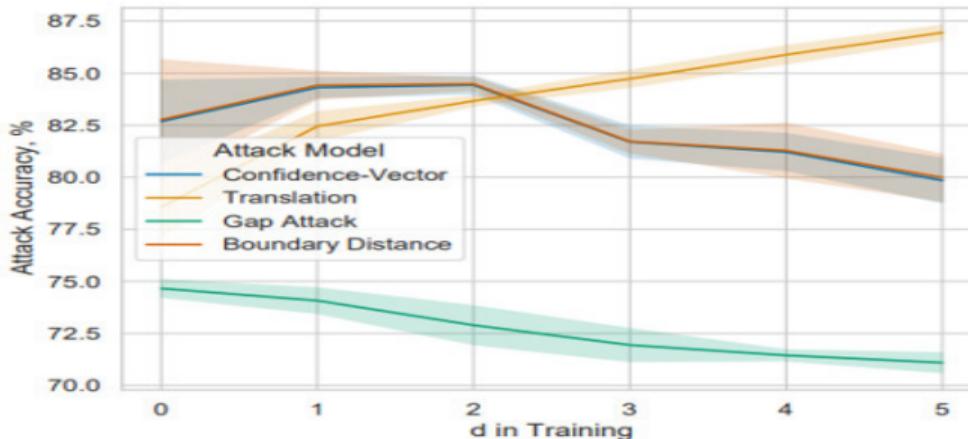
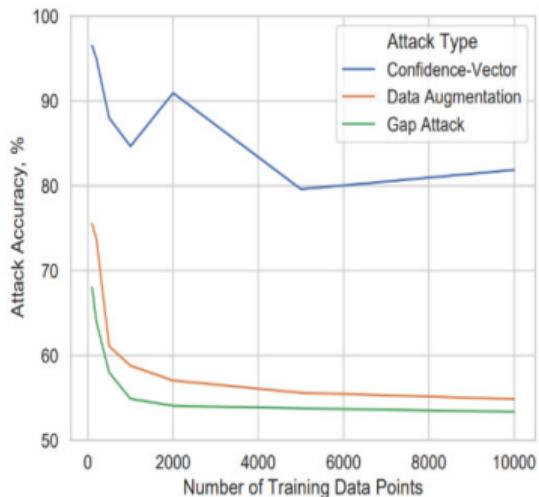
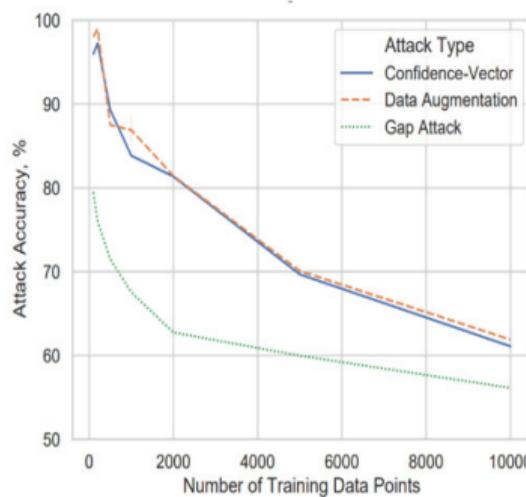


Figure 3. Accuracy of MI attacks on CIFAR-10 models trained with data augmentation on a subset of 2500 images. As in our attack, d controls the number of pixels by which images are translated during training, where no augmentation is $d = 0$. For models trained with significant amounts of data augmentation, MI attacks become *stronger* despite it generalizing better.

Are CB defenses always effective against LO attacks?

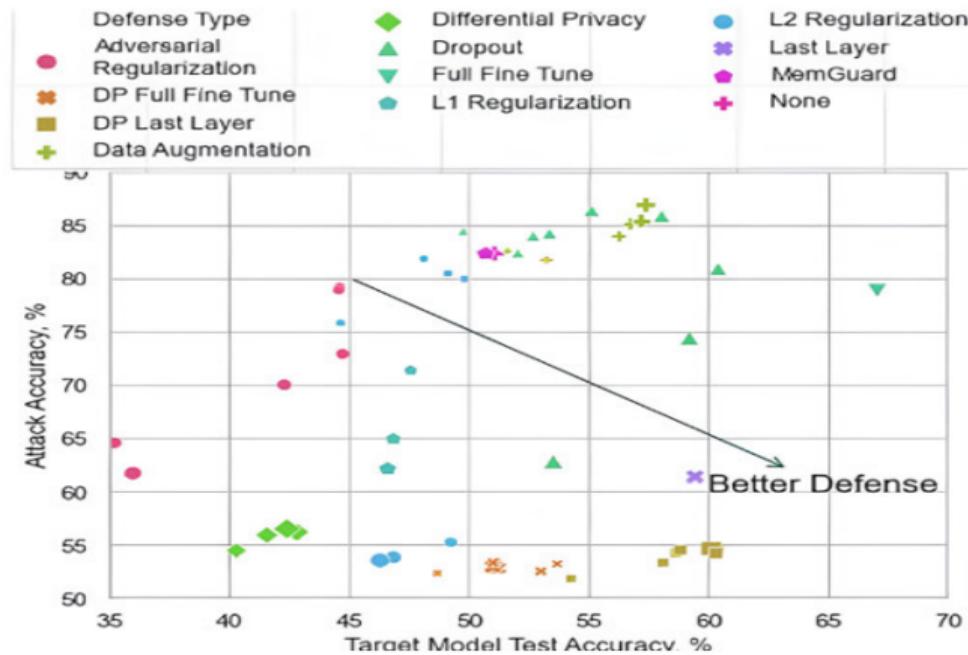


(a) Without Augmentations

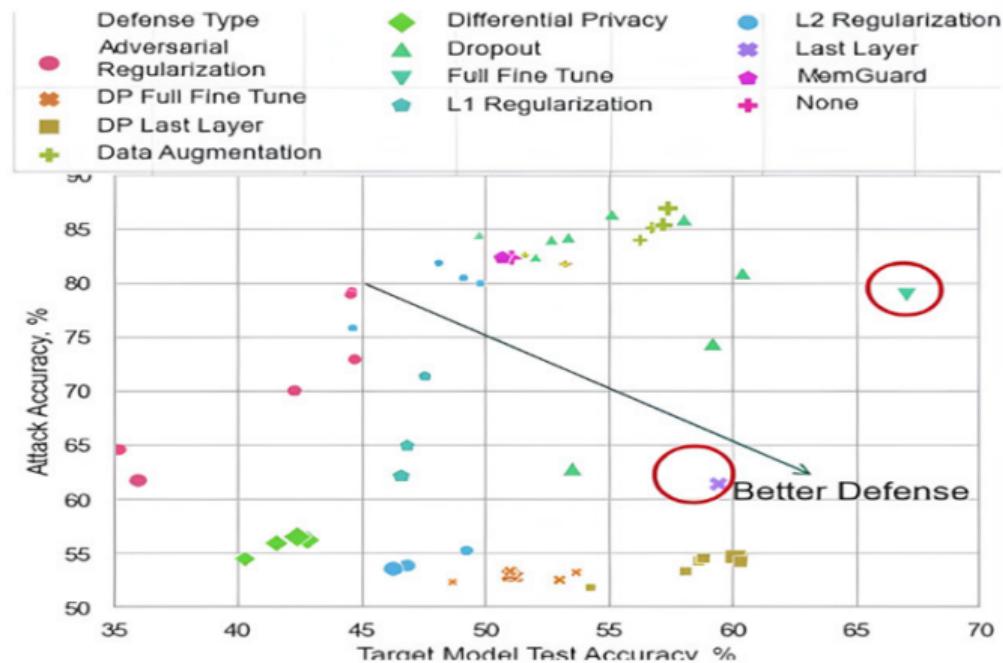


(b) With Augmentations

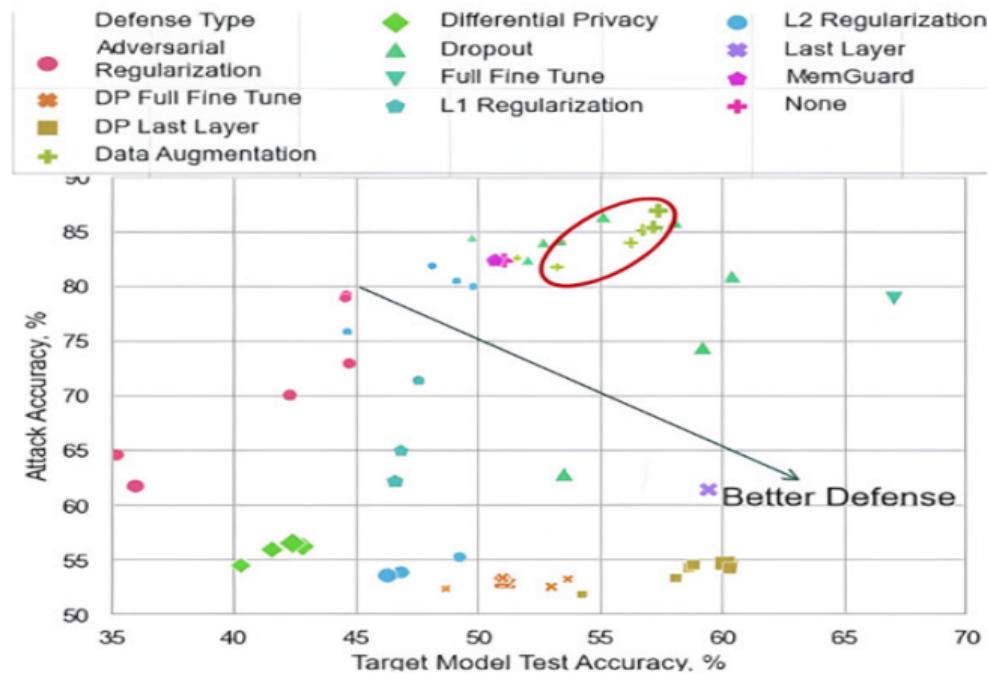
Which defenses prevent both LO and CB attacks?



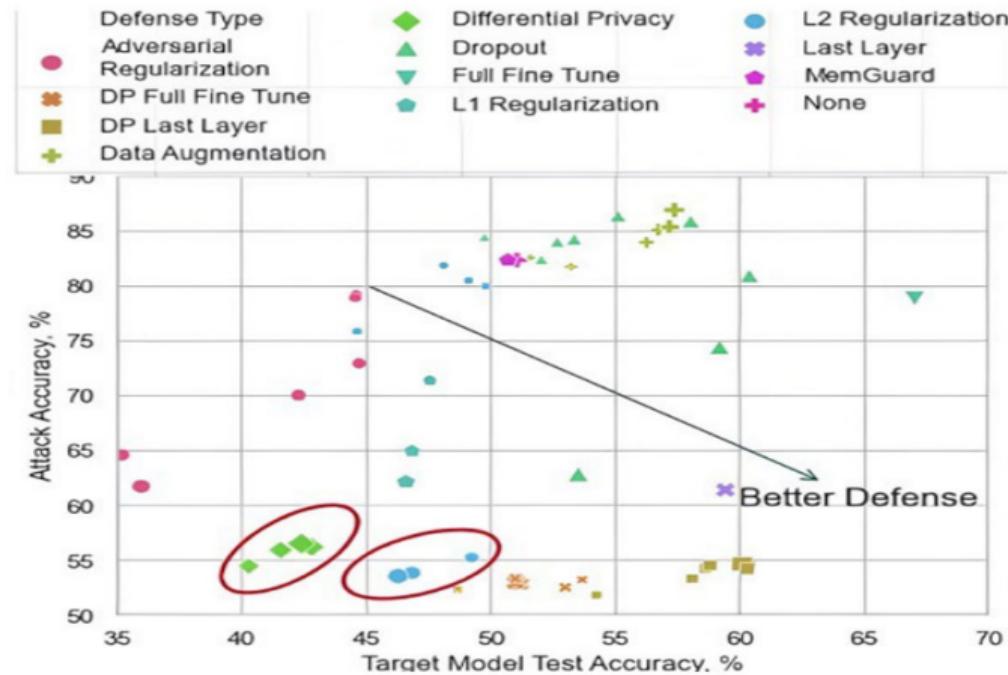
Which defenses prevent both LO and CB attacks?



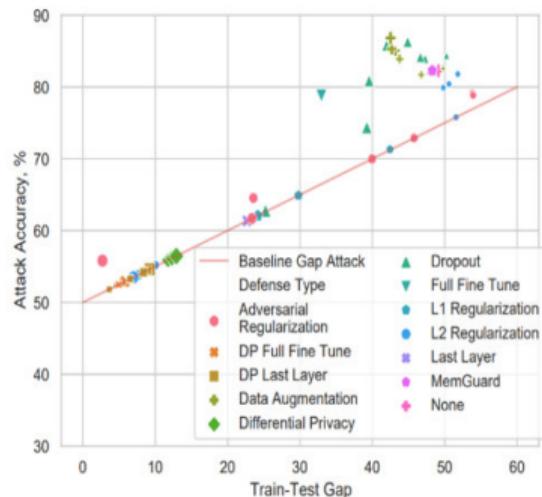
Which defenses prevent both LO and CB attacks?



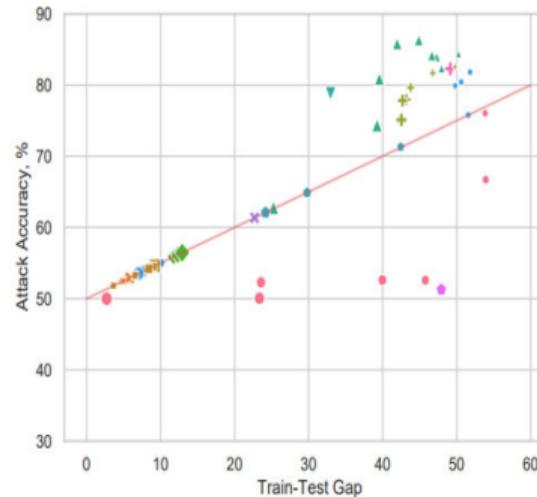
Which defenses prevent both LO and CB attacks?



Which defenses prevent both LO and CB attacks?



(a) Label-Only Attacks



(b) Full Confidence-Vector Attacks

Which defenses prevent both LO and CB attacks?

Query Interface	Attack Feature	Knowledge	Source
confidence vector	$h(x), y$	train, data, label	(Shokri et al., 2016)
confidence vector	$h(x)$	train, data	(Long et al., 2017)
confidence vector	$h(x)$	-	(Salem et al., 2018)
confidence vector	$\mathcal{L}(h(x), y)$	label	(Yeom et al., 2018)
confidence vector	$\mathcal{L}(h(x), y) + \tau(x)$	label	(Sablayrolles et al., 2019)
confidence vector	$-(\theta - \theta_0^*)^T \nabla_{\theta} \mathcal{L}(h(x), y)$	train, data, label, model	(Sablayrolles et al., 2019)
confidence vector	$h(x'), y$	train, data, label	(Song et al., 2019)
label-only	$\text{argmax } h(x), y$	label	(Yeom et al., 2018)
<hr/>			
label-only	$\text{argmax } h(\text{aug}(x)), y$	train, data, label	ours
label-only	$\text{dist}_h(x, y)$	train, data, label	ours
label-only	$\text{dist}_h(\text{aug}(x), y)$	train, data, label	ours

Which defenses prevent both LO and CB attacks?

- Differential privacy (with transfer learning)
- Strong L2 regularization

Conclusions

- Pros

- are more realistic in the real world problems
- can match, and even exceed, the success of prior confidence-vector attacks
- can break confidence masking defenses
- could also be instantiated in audio or natural language domains

- Cons

- need much more query than confidence-based MI attacks
- can not break differential privacy (with transfer learning)

References

- [1] C. A. ChoquetteChoo, F. Tramer, N. Carlini, N. Papernot "Label-Only Membership Inference Attacks," Proceedings of the 38th International Conference on Machine Learning, ICML, 2021
- [2] R. Shokri, M. Stronati, C. Song, V. Shmatikov "Membership Inference Attacks Against Machine Learning Models," IEEE Symposium on Security and Privacy, SP, 2017

Questions?

Renyi Differential Privacy

A paper by Ilya Minorov

Javad Hezareh, Mehraneh Najafi

Importance of Privacy-Preserving Techniques

- Protecting sensitive information in data analysis is crucial in our data-driven society.
- Privacy-preserving techniques, like differential privacy, ensure the confidentiality of individuals' personal data.
- Data-sharing initiatives are encouraged as individuals feel confident about their privacy being respected.
- Fairness and non-discrimination are promoted by preventing biases based on personal characteristics.

What is Differential Privacy as a concept?

- Differential privacy is a privacy-preserving framework for data analysis.
- It provides a mathematical definition of privacy guarantees.
- The goal is to enable accurate analysis while protecting the privacy of individuals.
- Differential privacy achieves privacy by adding controlled noise to query responses or data.



What is Differential Privacy as a concept?

- It ensures that the presence or absence of specific individuals in the dataset cannot be determined.
- The level of privacy is quantified by a parameter called epsilon (ϵ).
- A smaller epsilon value indicates stronger privacy protection.
- Differential privacy offers a trade-off between privacy and utility of the data.

The traditions: ϵ -differential privacy

- ϵ -differential privacy provides a formal definition of privacy guarantees.
- It ensures limited impact on query output when including or excluding an individual's data.
- Mechanism satisfies ϵ -differential privacy if probabilities of outcomes are approximately the same for similar datasets.
- Parameter ϵ represents the privacy budget or allowable privacy loss.

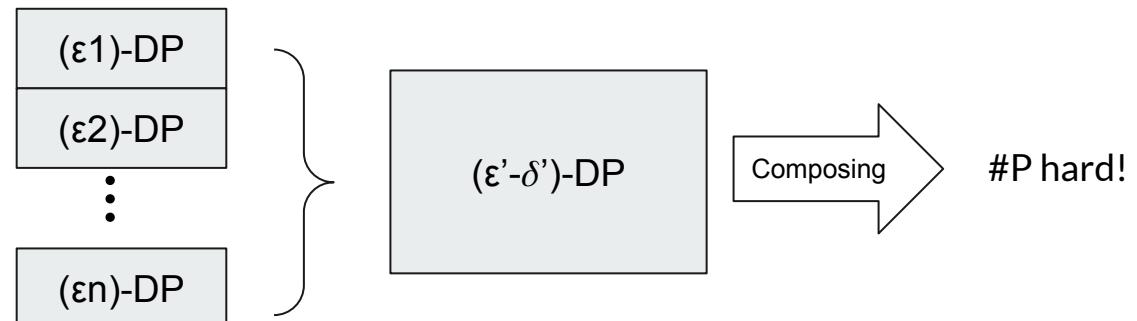
Relaxation: ϵ - δ differential privacy

- ϵ - δ Differential Privacy:
 - Incorporates an additional privacy parameter, δ .
 - Provides a more stringent privacy guarantee.
 - Bounds the probability of any arbitrary event an adversary can observe.
 - Establishes an upper bound on the overall privacy risk.
- ϵ -Differential Privacy vs. ϵ - δ Differential Privacy:
 - ϵ -differential privacy focuses on distinguishing neighboring datasets.
 - ϵ - δ differential privacy goes beyond distinguishability by bounding any possible event an adversary can observe.
 - ϵ - δ differential privacy ensures a stronger level of privacy protection.

What is wrong with (ϵ, δ) -differential privacy

$$\Pr[f(D) \in S] \leq e^\epsilon \Pr[f(D') \in S] + \delta.$$

- Gaussian Mechanism does not have catastrophic failure!
- Composing advanced composition





Renyi Divergence

$$D_\alpha(P\|Q) \triangleq \frac{1}{\alpha - 1} \log \mathbb{E}_{x \sim Q} \left(\frac{P(x)}{Q(x)} \right)^\alpha.$$

- For $\alpha = 1$:

$$\lim_{\alpha \rightarrow 1} D_\alpha(P\|Q)$$

$$D_1(P\|Q) = \mathbb{E}_{x \sim P} \log \frac{P(x)}{Q(x)}.$$

- For $\alpha = \infty$:

$$D_\infty(P\|Q) = \sup_{x \in \text{supp } Q} \log \frac{P(x)}{Q(x)}.$$

Renyi Divergence

- The relationship between the Renyi divergence with $\alpha = \infty$ and differential privacy is immediate:
 - A randomized mechanism f is ϵ -differentially private iff its distribution over any two adjacent inputs D and D' satisfies:

$$D_\infty(f(D) \| f(D')) \leq \epsilon.$$

Renyi Properties: “Bad Outcomes” Guarantee

- Consider a person, concerned about some adverse consequences, deliberating whether to withhold her record from the database.
- Let’s say some outputs of the mechanism are labeled “Bad”. DP guarantees that the probability of observing a bad outcome will not change (either way) by more than a factor of e^ϵ whether anyone’s record is part of the input or not.
- This guarantee is relaxed for Renyi differential privacy.

Renyi Properties: Robustness to Auxiliary Information

- Critical to the adoption of differential privacy as an operationally useful definition is its lack of assumptions on the adversary's knowledge.
- Assume that the adversary has a prior $p(D)$ over the set of possible inputs $D \in \mathcal{D}$, and observes an output X of an ϵ -differentially private mechanism f .
- Its posterior satisfies the following guarantee for all pairs of adjacent inputs $D, D' \in \mathcal{D}$ and all $X \in \mathcal{R}$:

$$\frac{p(D | X)}{p(D' | X)} \leq e^\epsilon \frac{p(D)}{p(D')}.$$

Renyi Properties: Robustness to Auxiliary Information

- In other words, evidence obtained from an ϵ -differentially private mechanism does not move the relative probabilities assigned to adjacent inputs by more than e^ϵ .
- Let the random variable $R(D,D')$ be defined as follows:

$$R(D, D') \sim \frac{p(D'|X)}{p(D|X)} = \frac{p(X|D') \cdot p(D')}{p(X|D) \cdot p(D)},$$

where $X \sim f(D)$.

$$\mathbb{E}_Q \left[\left\{ \frac{R_{\text{post}}(D, D')}{R_{\text{prior}}(D, D')} \right\}^\alpha \right] = \mathbb{E}_Q [P(x)^\alpha Q(x)^{-\alpha}] =$$
$$\exp[(\alpha - 1) D_\alpha(f(D') \| f(D))].$$

- In simpler terms, this means that the Renyi divergence of order α between the probability distributions induced by neighboring datasets D and D' provides an upper bound on the expected change in the Bayes factor.

Renyi Properties: Post-Processing

- If we have a differential private mechanism $f : \mathcal{D} \rightarrow \mathcal{R}$, can we diminish its privacy by manipulating its output?
- Consider any randomized mapping $g : \mathcal{R} \rightarrow \mathcal{R}'$
- ϵ -DP mechanism has the post-processing property
 - $\Pr[g \circ f(D) \in S] \leq e^\epsilon \Pr[g \circ f(D') \in S]$
- (α, ϵ) -RDP mechanism also has this property
 - $D_\alpha(P||Q) \geq D_\alpha(g(P)||g(Q))$

Renyi Properties: Preservation Under Adaptive Sequential Composition

- Consider two mechanism $f(\cdot)$ ϵ_1 -DP and $g(\cdot)$ ϵ_2 -DP
- If we release $f(D)$ and $g(D)$, do we still have privacy?
- Composition of $f(\cdot)$ and $g(\cdot)$ is $\epsilon_1 + \epsilon_2$ -DP
- If $f(\cdot)$ be (α, ϵ_1) -RDP and $g(\cdot)$ be (α, ϵ_2) -RDP, the the composition of $f(\cdot)$ and $g(\cdot)$ is $(\alpha, \epsilon_1 + \epsilon_2)$ -RDP

Renyi Properties: Preservation Under Adaptive Sequential Composition (Proof)

$$\begin{aligned} & \exp [(\alpha - 1)D_\alpha(h(D)\|h(D'))] \\ &= \int_{\mathcal{R}_1 \times \mathcal{R}_2} Z(x, y)^\alpha Z'(x, y)^{1-\alpha} dx dy \\ &= \int_{\mathcal{R}_1} \int_{\mathcal{R}_2} (X(x)Y(x, y))^\alpha (X'(x)Y'(x, y))^{1-\alpha} dy dx \\ &= \int_{\mathcal{R}_1} X(x)^\alpha X'(x)^{1-\alpha} \left\{ \int_{\mathcal{R}_2} Y(x, y)^\alpha Y'(x, y)^{1-\alpha} dy \right\} dx \\ &\leq \int_{\mathcal{R}_1} X(x)^\alpha X'(x)^{1-\alpha} dx \cdot \exp((\alpha - 1)\epsilon_2) \\ &\leq \exp((\alpha - 1)\epsilon_1) \exp((\alpha - 1)\epsilon_2) \\ &= \exp((\alpha - 1)(\epsilon_1 + \epsilon_2)), \end{aligned}$$

Renyi Properties: Group Privacy

- What if our assumptions about our data is not correct?
 - For example, a single family contributing to a survey will likely share many socio-economic, demographic, and health characteristics
- Will privacy collapse or we still have something to say?
 - The differential privacy guarantee will scale down linearly with the number of family members

Renyi Properties: Group Privacy

- **Definition:** We say that $g : D \rightarrow D'$ is c -stable if $g(A)$ and $g(B)$ are adjacent in D' implies that there exists a sequence of length $c + 1$ so that $D_0 = A, \dots, D_c = B$ and all (D_i, D_{i+1}) are adjacent in D .
- If f is ϵ -DP and $g : D' \rightarrow D$ is c -stable, then $f \circ g$ is $(c\epsilon)$ -DP.
- If f is (α, ϵ) -RDP and $g : D' \rightarrow D$ is 2^c -stable and $\alpha \geq 2^{c+1}$, then $f \circ g$ is $(\alpha/2^c, 3^c\epsilon)$ -RDP

RDP and (ϵ, δ) -DP

- If f is ϵ -DP, then f is (∞, ϵ) -RDP. And by monotonicity f is also (α, ϵ) -RDP for all $\alpha < \infty$.
- If f is (α, ϵ) -RDP, then f is $\left(\epsilon + \frac{\log 1/\delta}{\alpha - 1}, \delta\right)$ -DP for any $0 < \delta < 1$

Denote $\Pr[f(D') \in S]$ by Q and consider two cases.

Case I. $e^\epsilon Q > \delta^{\alpha/(\alpha-1)}$. Continuing the above,

$$\begin{aligned}\Pr[f(D) \in S] &\leq \{e^\epsilon Q\}^{1-1/\alpha} = e^\epsilon Q \cdot \{e^\epsilon Q\}^{-1/\alpha} \\ &\leq e^\epsilon Q \cdot \delta^{-1/(\alpha-1)} \\ &= \exp\left(\epsilon + \frac{\log 1/\delta}{\alpha - 1}\right) \cdot Q.\end{aligned}$$

Case II. $e^\epsilon Q \leq \delta^{\alpha/(\alpha-1)}$. This case is immediate since

$$\Pr[f(D) \in S] \leq \{e^\epsilon Q\}^{1-1/\alpha} \leq \delta,$$

RDP vs (ϵ, δ) -DP

- Probabilistic Privacy Guarantee
- Baseline-Dependent Guarantees

Probabilistic Privacy Guarantee

- The standard “bad outcomes” guarantee of ϵ -DP is independent of the probability of a bad outcome
- (ϵ, δ) -DP, allows for an additional δ term, which allows for a complete privacy compromise with probability δ
- RDP even with very weak parameters never allows a total breach of privacy with no residual uncertainty

Baseline-Dependent Guarantees

- RDP bound gets weaker for less likely outcomes
- Contrasted with the pure ϵ -DP this type of guarantee is conceptually weaker and more onerous in application
- However, in comparison with (ϵ, δ) -DP the analysis via RDP simpler and, especially for probabilities that are smaller than δ , leads to stronger bounds

Conclusions

- (RDP) is a natural generalization of pure differential privacy
 - RDP shares, with some adaptations, many properties that make differential privacy a useful and versatile tool
 - RDP analysis of Gaussian noise is particularly simple
- ★ RDP yields useful insight into analysis of differentially private mechanisms.

**Thanks for listening
Any question?**



LARGE LANGUAGE MODELS CAN BE STRONG DIFFERENTIALLY PRIVATE LEARNERS

Xuechen Li¹, Florian Tramèr², Percy Liang¹, Tatsunori Hashimoto¹

¹Stanford University ²Google Research

{lxuechen, tramer, pliang}@cs.stanford.edu, thashim@stanford.edu

SPML Presentation 2
Spring 2023

Hamidreza Amirzadeh
Ali Abdollahi

- LLMs can memorize training data
- Data extraction attacks are surprisingly effective for LLMs (Carlini et al., 2021)
- These extracted examples include (public) personally identifiable information ***names, phone numbers, and email addresses***
- We need provable guarantees that models won't leak private data

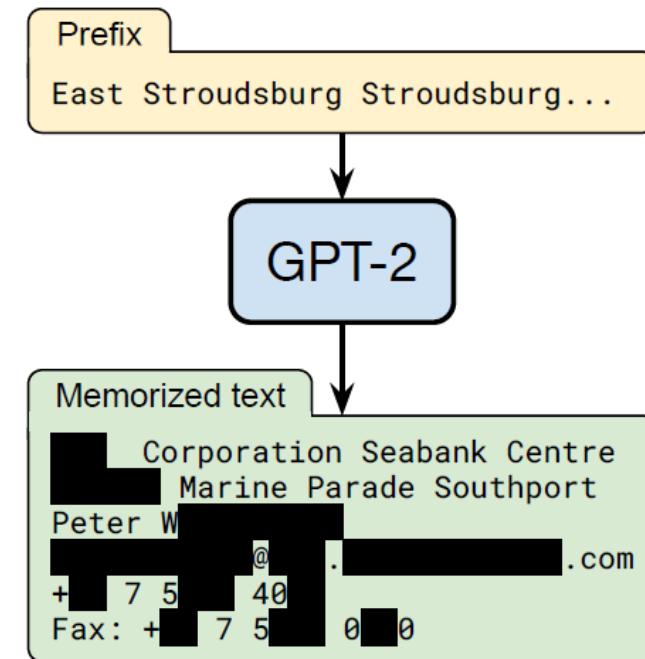


Figure 1: **Our extraction attack.** Given query access to a neural network language model, we extract an individual person's name, email address, phone number, fax number, and physical address. The example in this figure shows information that is all accurate so we redact it to protect privacy.

(Carlini et al., 2021)

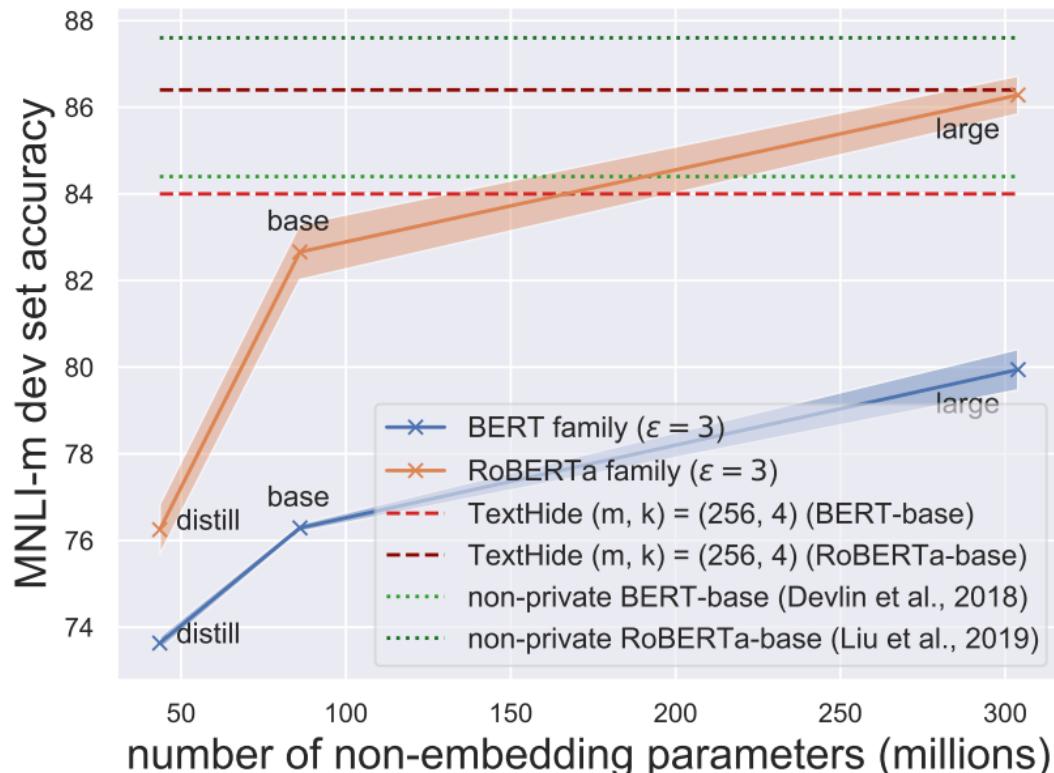
- Straightforward attempts to apply DP-SGD to NLP tasks ?!
 - High computational overhead
 - Large performance drops
- Why is so ?
 - DP-SGD injects **noise** that must scale with **parameter dimensions** → problematic in LLMs
- This paper shows that this performance drop can be mitigated with use of :
 - Large pretrained language models
 - Non-standard **hyperparameters** that suit DP optimization
 - Fine-tuning objectives which are **aligned** with the pretraining procedure

- Contributions :
 1. Obtain **DP models** for two main NLP tasks. Performance comparable to Non-private ones.
 - Text classification → fine-tuning BERT / RoBERTa
 - Text generation → fine-tuning GPT-2
 2. Propose a memory saving technique called **ghost clipping**.
 3. Shows that **larger** pretrained models lead to **better** private fine-tuning results.

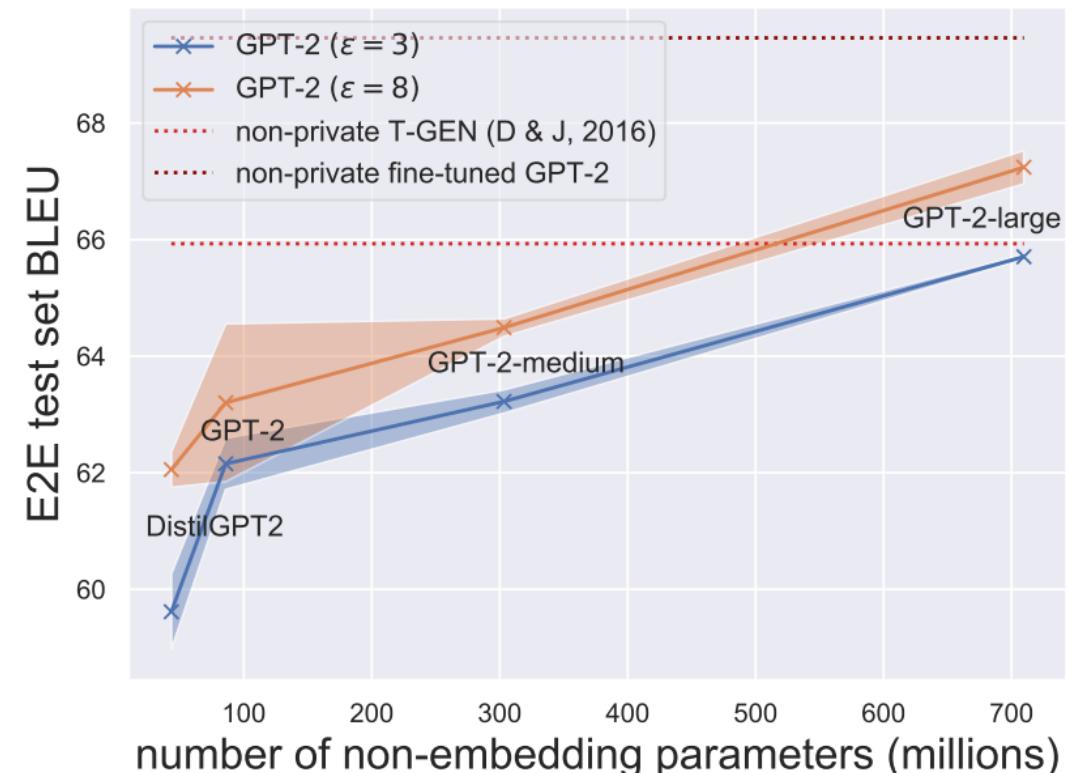
Definition 1 $((\epsilon, \delta)$ -DP). A randomized algorithm $\mathcal{M} : \mathcal{X} \rightarrow \mathcal{Y}$ is (ϵ, δ) -differentially private if for all adjacent datasets $X, X' \in \mathcal{X}$ and all $Y \subset \mathcal{Y}$, $\mathbb{P}(\mathcal{M}(X) \in Y) \leq \exp(\epsilon)\mathbb{P}(\mathcal{M}(X') \in Y) + \delta$.

- DP algorithms ensure that random outputs obtained from similar inputs are difficult to distinguish.
- DP learning → DP optimizers → rely on privatizing gradients
 1. Clipping the gradient → ensures that each example has bounded influence on the parameter update
 2. Adding noise → prevents exact tracing of particular examples $z \sim \mathcal{N}(0, C^2\sigma^2 I_p)$
- Problem : norm of p -dimensional Gaussian $\|z\|_2$ scales as $C\sigma\sqrt{P}$
 - Larger models would experience heavier noise per update
 - So DP optimization perform poorly at training high dimensional deep models

- This paper:
 - Q : Is it possible to build high quality DP NLP models on moderate amounts of private training data?
 - A : Yes!
- Overview of results (see next slide):
 - For text classification, DP fine-tuning can outperform TextHide (InstaHide for text)
 - For text generation, DP fine-tuning can outperform strong non-private baselines
 - Larger and better pretrained models result in better fine-tuned performance



(a) Sentence classification
MNLI-matched (Williams et al., 2018)



(b) Natural language generation
E2E (Novikova et al., 2017)

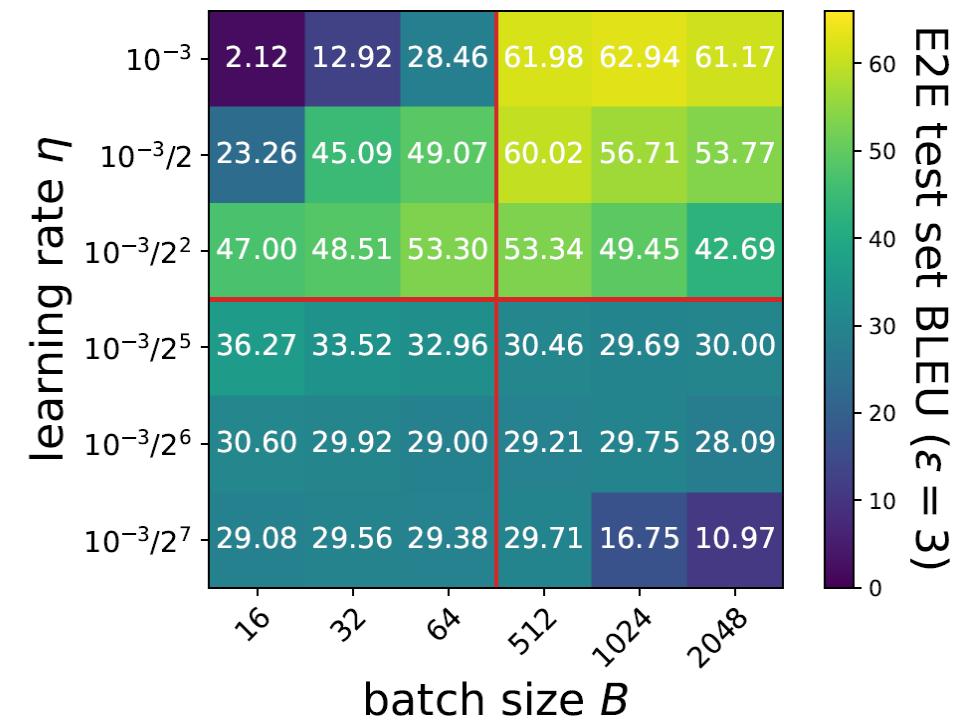
- How did they get these results ?!
 1. Making DP fine-tuning **effective**
 - Hyperparameters
 - Fine-tuning objective
 2. Making DP fine-tuning **efficient**
 - Ghost clipping

Effective DP fine-tuning

- Approach : fine-tune public pretrained models with DP-Adam
- What do we need for good performance ?
 - Good hyperparameters
 - DP learning is sensitive to choices of hyperparameters
 - Studied of how hyperparameters affect performance
 - Good hyperparameters tend to transfer across tasks
 - Fine-tuning objective
 - Objectives that make learning easy results in better private models
 - We want the fine-tuning objective to be close to the pretraining objective

Effective DP fine-tuning : Hyperparameters → batch size, learning rate

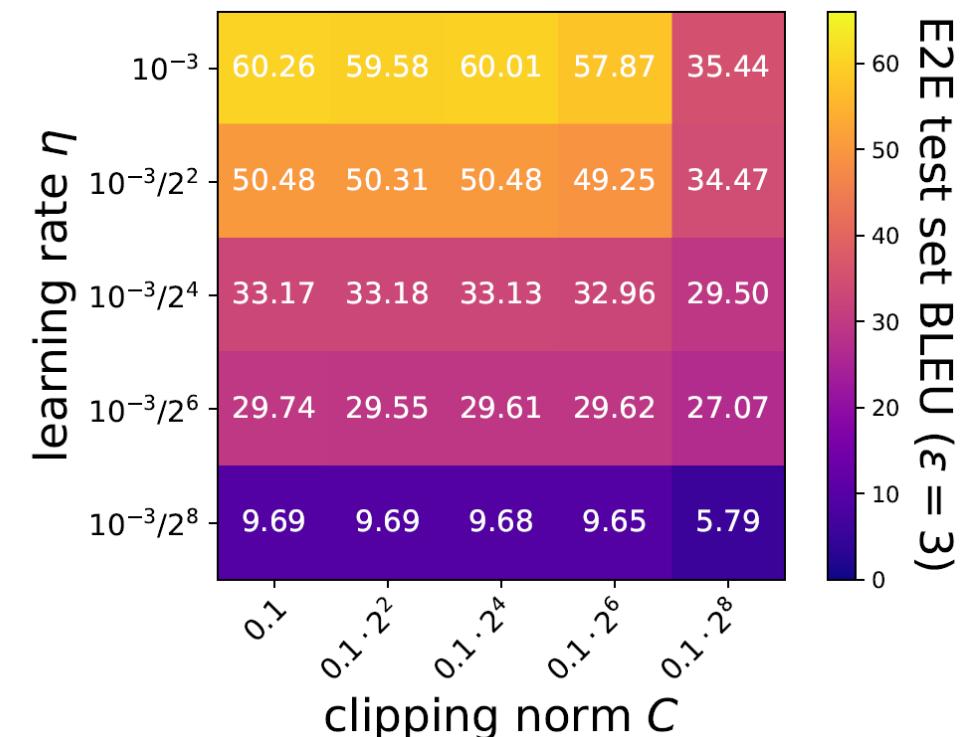
- Good batch sizes and learning rates for private learning is different from those typical for non-private learning
- Dependence of the optimal batch size on the learning rate and training epochs makes its selection complex
- Learning rate and batch size jointly affect performance
- Need large batch size with large learning rate



Non-private models : BLEU of ~65

Effective DP fine-tuning : Hyperparameters → Clipping norm C

- Scale of noise injected depends on this clipping norm
- Experiments show that small clipping norm is suitable for private learning
- Not setting this properly → large performance drop



Non-private models : BLEU of ~65

Effective DP fine-tuning : Fine-tuning objectives

- Fine-tuned models on language generation tasks work well since the pretraining objective and downstream tasks are **aligned**. → both involved predicting sequences of tokens
- This alignment simplifies the task and benefits private learning
- There is no objective alignment in classification task
 - *Pretraining* : predicts masked out words from a large vocabulary
 - *fine-tuning* : predicts integer labels
- To eliminate this discrepancy:
 - Instead of predicting integer labels, we ask the model to predict **textualized labels** during fine-tuning



- How did they get these results ?!

1. Making DP fine-tuning **effective**

- Hyperparameters
- Fine-tuning objective

2. Making DP fine-tuning **efficient**

- Ghost clipping

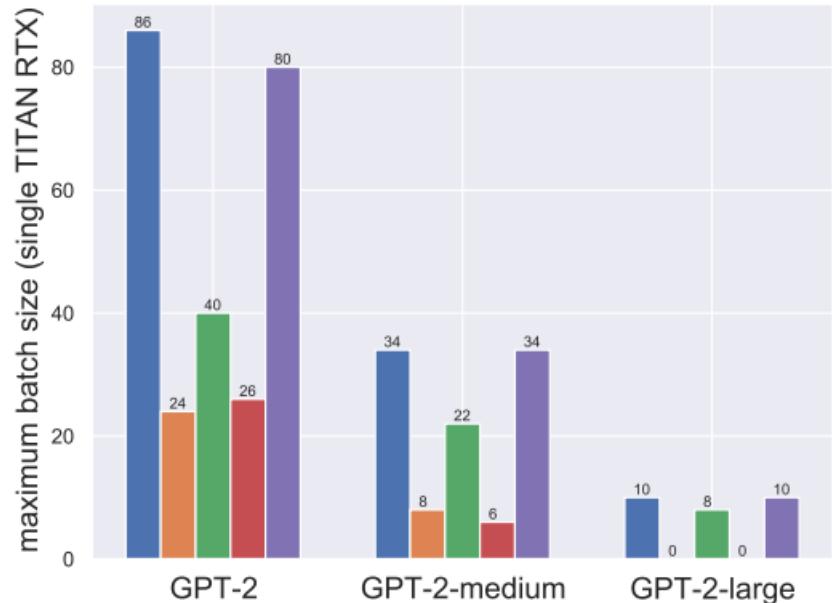
Efficient DP fine-tuning

1. A time-costly solution to the memory problem is micro-batching: Split large batches into multiple smaller ones and aggregate the results after processing each small batch individually.
2. For clipping we first compute the scaling factor : $c_i = \min(1, C/\|\nabla \mathcal{L}_i\|_2)$, and the difficulty is computing the per-example gradient norm and it uses much memory but computing the per-example gradient norm can be done by computing the per-example gradient norms for individual layers of the neural net $\|\nabla_{W^{(1)}} \mathcal{L}_i\|_2, \dots, \|\nabla_{W^{(L)}} \mathcal{L}_i\|_2$ one at a time. So the per-example gradient norm of any network can be computed in a layer-by-layer fashion with only one per-example gradient tensor for a single layer being instantiated at any time.
3. GHOST CLIPPING

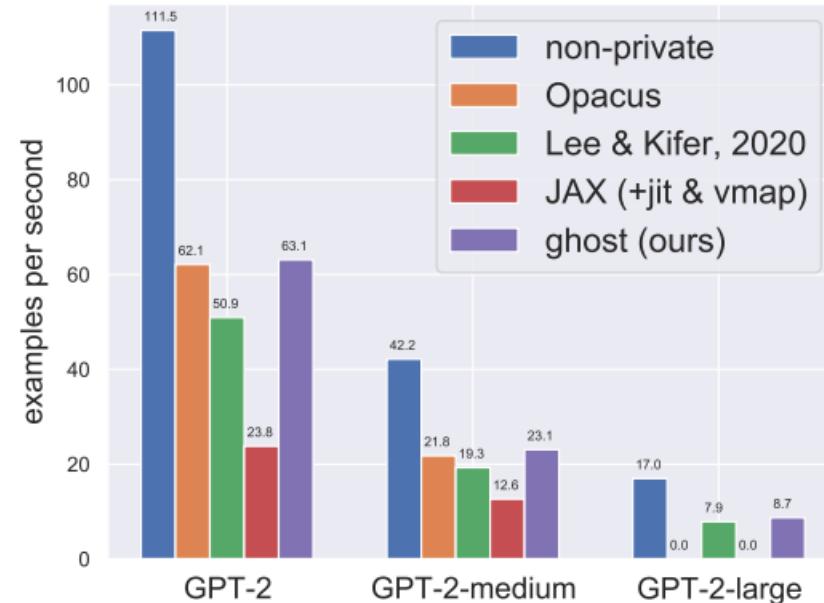
GHOST CLIPPING

- Let $a \in \mathbb{R}^{B \times T \times d}$ be the input to a linear layer and let $g_i \in \mathbb{R}^{B \times T \times p}$ be the gradient of the loss w.r.t. the output of the layer.
- per-example gradient is the product of two matrices: $\nabla_W \mathcal{L}_i = g_i^\top a_i \in \mathbb{R}^{p \times d}$.
- the squared per example gradient norm for this layer $\|\nabla_W \mathcal{L}_i\|_F^2$ obeys the following key identity: $\|\nabla_W \mathcal{L}_i\|_F^2 = \text{vec}(a_i a_i^\top)^\top \text{vec}(g_i g_i^\top)$.
- memory complexity is of the order $\mathcal{O}(BT^2)$ when $a_i a_i^\top \in \mathbb{R}^{T \times T}$ and $g_i g_i^\top \in \mathbb{R}^{T \times T}$ as opposed to $\mathcal{O}(Bpd)$ in the naive approach.
- For instance, for GPT-2, $d \approx 50,000$ and $p = 768$ for the embedding layer, and the context window $T \leq 1024$.

GHOST CLIPPING



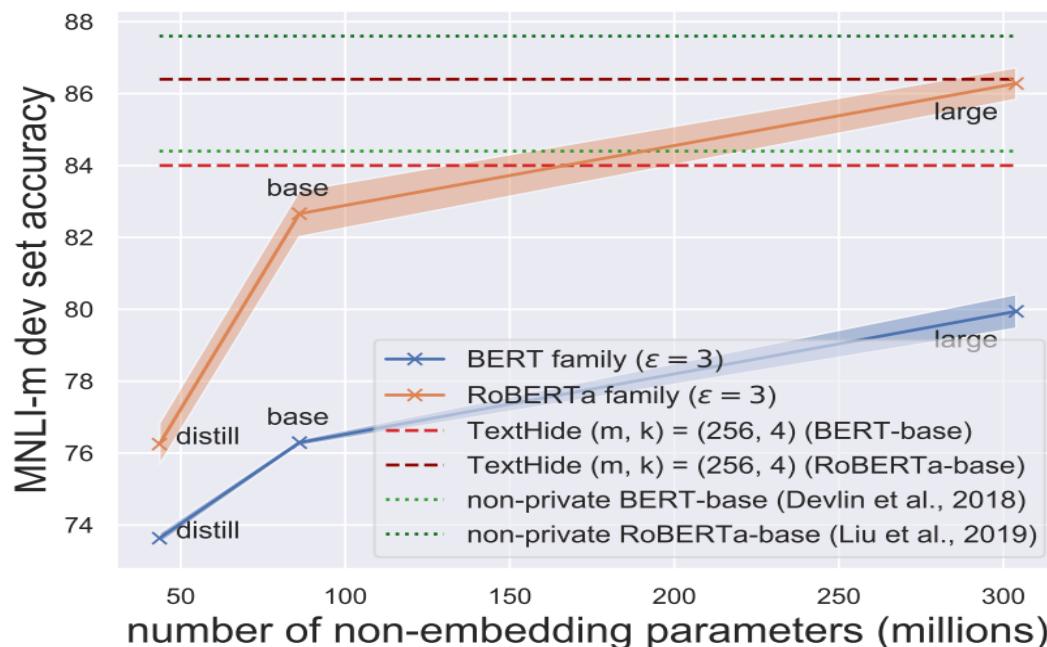
(a) Memory



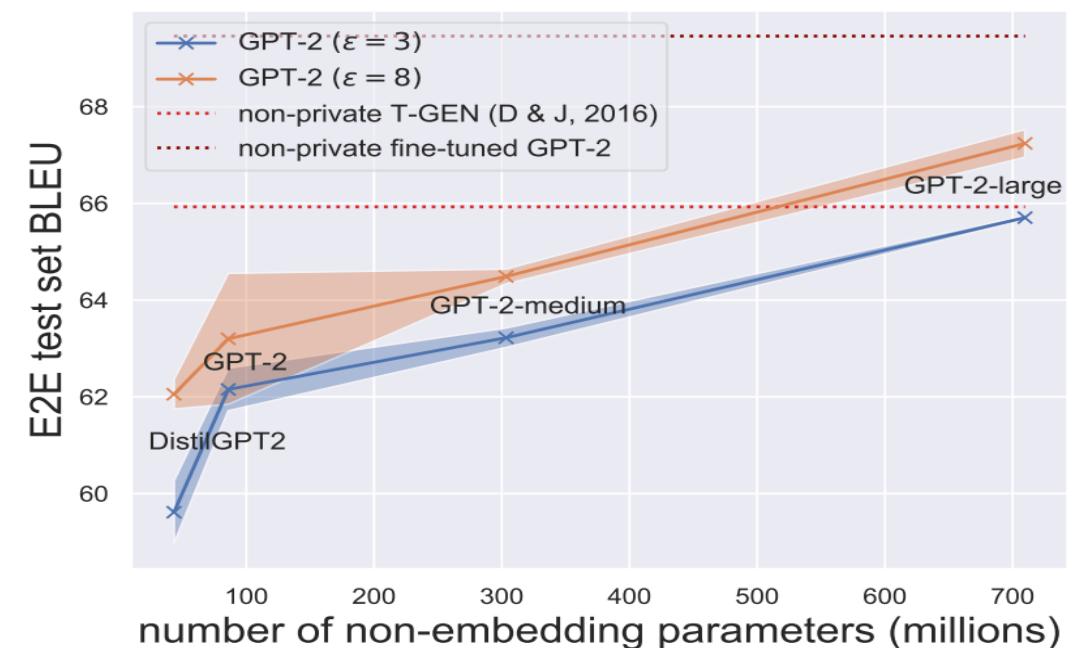
(b) Throughput

Figure 4: **Left:** Ghost clipping is 3 times more memory efficient than Opacus and is almost as efficient as non-private training for typical sequences across model sizes. For GPT-2-large, we were unable to fit single-example micro batches together with gradient accumulation with Opacus or JAX on a TITAN RTX GPU (24 GBs of VRAM). **Right:** DP optimization with ghost clipping processes ~10% more examples than the approach by Lee & Kifer (2020) under unit time for GPT-2-large.

- Low dimensional updates are not necessarily better
 1. We observe that larger pretrained models lead to better private fine-tuned performance.
 2. Do methods that optimize fewer parameters lead to better results under DP even if they perform similarly non-privately? Empirical results suggest otherwise and that full fine tuning is a strong baseline that even matches specialized low-dimensional DP learning methods for both classification and generation .



(a) Sentence classification
MNLI-matched (Williams et al., 2018)



(b) Natural language generation
E2E (Novikova et al., 2017)

- SENTENCE CLASSIFICATION

- The table shows that using larger pretrained models and the text-infilling objective generally improve classification accuracy.
- We compare full fine-tuning with reparameterized gradient perturbation (RGP). The method is designed to privatize gradients projected onto low dimensional subspaces and was motivated to reduce DP noise in high dimensional models.

Table 1: Full fine-tuning larger pretrained models with text infilling has best performance. Results are dev set accuracies. Best numbers based on two-sample test for each privacy level are in bold.

Method	$\epsilon = 3$				$\epsilon = 8$			
	MNLI-(m/mm)	QQP	QNLI	SST-2	MNLI-(m/mm)	QQP	QNLI	SST-2
RGP (RoBERTa-base)	-	-	-	-	80.5/79.6	85.5	87.2	91.6
RGP (RoBERTa-large)	-	-	-	-	86.1/86.0	86.7	90.0	93.0
full (RoBERTa-base)	82.47/82.10	85.41	84.62	86.12	83.30/83.13	86.15	84.81	85.89
full (RoBERTa-large)	85.53/85.81	86.65	88.94	90.71	86.28/86.54	87.49	89.42	90.94
full + infilling (RoBERTa-base)	82.45/82.99	85.56	87.42	91.86	83.20/83.46	86.08	87.94	92.09
full + infilling (RoBERTa-large)	86.43/86.46	86.43	90.76	93.04	87.02/87.26	87.47	91.10	93.81
$\epsilon \approx$ (Gaussian DP + CLT)	2.52	2.52	2.00	1.73	5.83	5.85	4.75	4.33
$\epsilon \approx$ (Compose tradeoff func.)	2.75	2.75	2.57	2.41	7.15	7.16	6.87	6.69

- **Table-To-Text**

1. they studies different fine-tuning methods under DP for table-to-text generation where the goal is to generate natural language descriptions of table entries.
2. they compared full fine-tuning (full) against a suite of parameter-efficient approaches which includes LoRA, prefix-tuning (prefix), RGP, and fine-tuning the top 2 Transformer blocks (top2), all of which optimize few parameters.

Table 2: Full fine-tuning performs on par with or outperforms others methods that execute gradient update in low dimensional spaces. Results are on E2E from fine-tuning GPT-2.

Metric	DP Guarantee	Gaussian DP	Compose	Method					
		+ CLT	tradeoff func.	full	LoRA	prefix	RGP	top2	retrain
BLEU	$\epsilon = 3$	$\epsilon \approx 2.68$	$\epsilon \approx 2.75$	61.519	58.153	47.772	58.482	25.920	15.457
	$\epsilon = 8$	$\epsilon \approx 6.77$	$\epsilon \approx 7.27$	63.189	63.389	49.263	58.455	26.885	24.247
	non-private	-	-	69.463	69.682	68.845	68.328	65.752	65.731
ROUGE-L	$\epsilon = 3$	$\epsilon \approx 2.68$	$\epsilon \approx 2.75$	65.670	65.773	58.964	65.560	44.536	35.240
	$\epsilon = 8$	$\epsilon \approx 6.77$	$\epsilon \approx 7.27$	66.429	67.525	60.730	65.030	46.421	39.951
	non-private	-	-	71.359	71.709	70.805	68.844	68.704	68.751

- Chit-Chat Dialog

Table 3: Fine-tuning with DP-Adam yields high quality chit-chat dialog generation models.

Model	DP Guarantee	Gaussian DP +CLT	Compose tradeoff func.	Metrics		
				F1 ↑	Perplexity ↓	Quality (human) ↑
GPT-2	$\epsilon = 3$	$\epsilon \approx 2.54$	$\epsilon \approx 2.73$	15.90	24.59	-
	$\epsilon = 8$	$\epsilon \approx 6.00$	$\epsilon \approx 7.13$	16.08	23.57	-
	non-private	-	-	17.96	18.52	-
GPT-2-medium	$\epsilon = 3$	$\epsilon \approx 2.54$	$\epsilon \approx 2.73$	15.99	20.68	-
	$\epsilon = 8$	$\epsilon \approx 6.00$	$\epsilon \approx 7.13$	16.53	19.25	-
	non-private	-	-	18.64	15.40	-
DialoGPT-medium	$\epsilon = 3$	$\epsilon \approx 2.54$	$\epsilon \approx 2.73$	17.37	17.64	2.82 (2.56, 3.09)
	$\epsilon = 8$	$\epsilon \approx 6.00$	$\epsilon \approx 7.13$	17.56	16.79	3.09 (2.83, 3.35)
	non-private	-	-	19.28	14.28	3.26 (3.00, 3.51)
HuggingFace (ConvAI2 winner)	non-private	-	-	19.09	17.51	-
HuggingFace (our implementation)	non-private	-	-	16.36	20.55	3.23 (2.98, 3.49)
Reference	-	-	-	-	-	3.74 (3.49, 4.00)

- Conclusion :
 - DP fine-tuning with a proper setup is a competitive baseline that is worth trying before shifting to less formal notions of privacy
- But :
 - Did not study how **weight decay, learning rate schedule, clipping norm schedule** affect performance
- For future work :
 - Study how pretraining helps private learning
 - Whether better pretrained models for private learning could be built