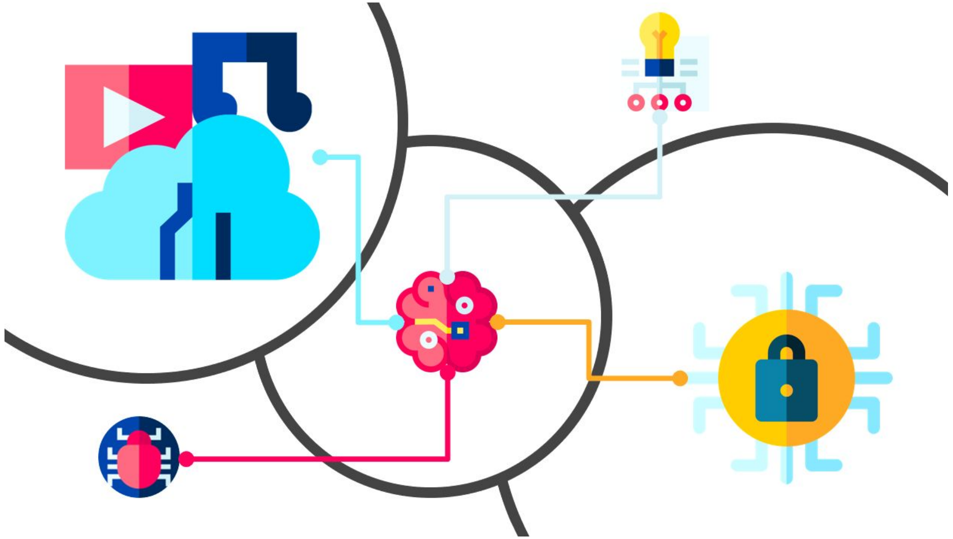
Verification



Quinlan Sykora, Alexander Cui, Yi Tao

UNIVERSITY OF TORONTO

[1] Image from *What Machine Learning can do and what can not? [Difficult stuff in simple words]* - Medium.com. The article can be found at https://medium.com/@stefanovskyi/what-machine-learning-can-do-and-what-can-not-difficult-stuff-in-simple-words-ef9f245dd6f6

Machine Learning on the Cloud

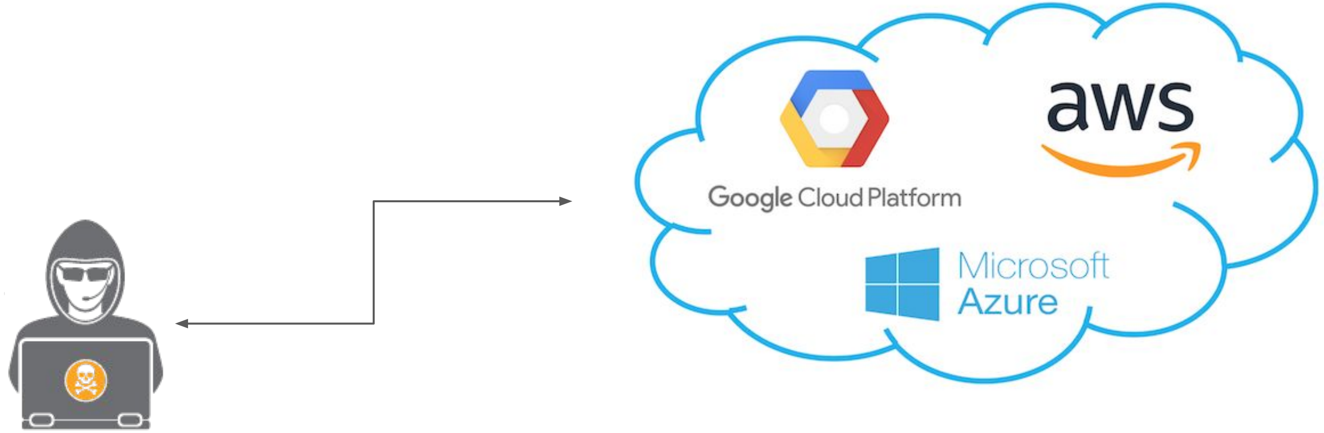
Machine learning requires lots of resources, often making in inaccessible to most individuals

The has resulted in outsourcing the process to training servers on the cloud

[1] Image from *Comparing The Top 3: Google Cloud, AWS & Microsoft Azure* - clouve.com. The article can be found at https://www.clouve.com/blog/comparing-the-top-3-google-cloud-aws-microsoft-azure/

Machine Learning on the Cloud

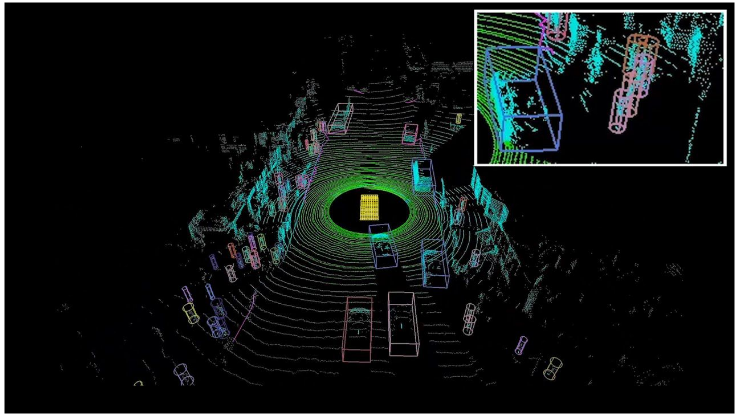
But can we really trust them and the training they do?



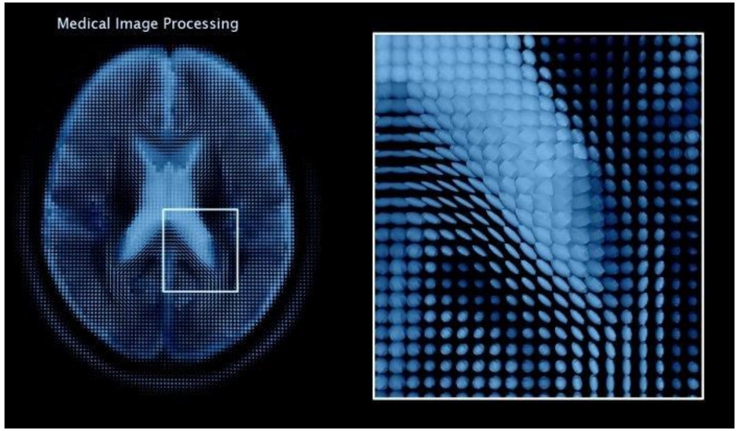
[1] Image from *Comparing The Top 3: Google Cloud, AWS & Microsoft Azure* - clouve.com. The article can be found at https://www.clouve.com/blog/comparing-the-top-3-google-cloud-aws-microsoft-azure/

Machine Learning for Mission Critical Tasks

Some scenarios require training to occur exactly as specified to be safe



[1]



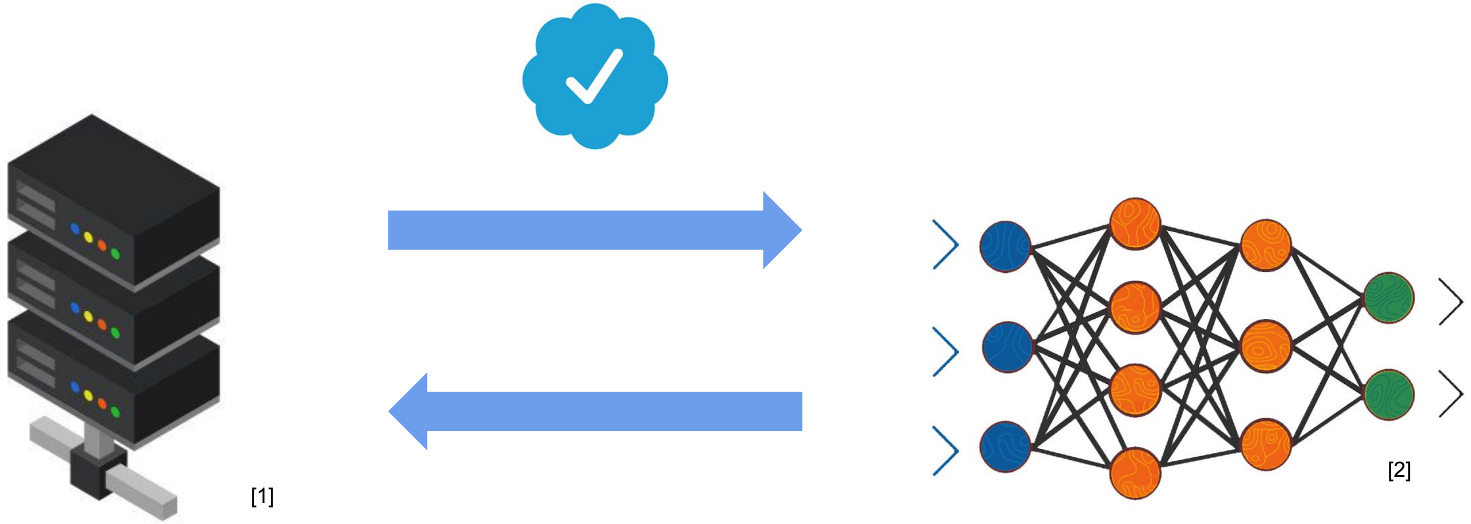
[2]

[1] Image from *Laser Focused: How Multi-View LidarNet Presents Rich Perspective for Self-Driving Cars* - Nvidia blog. The article can be found at https://blogs.nvidia.com/blog/2020/03/11/drive-labs-multi-view-lidarnet-self-driving-cars/
[2] Image from *Medical Image Analysis Deep Learning* - kdnuggets.com. The article can be found at https://www.kdnuggets.com/2017/03/medical-image-analysis-deep-learning.html

How do you make sure training has occurred?

You by definition don't have the resources to check all facets of the model yourself

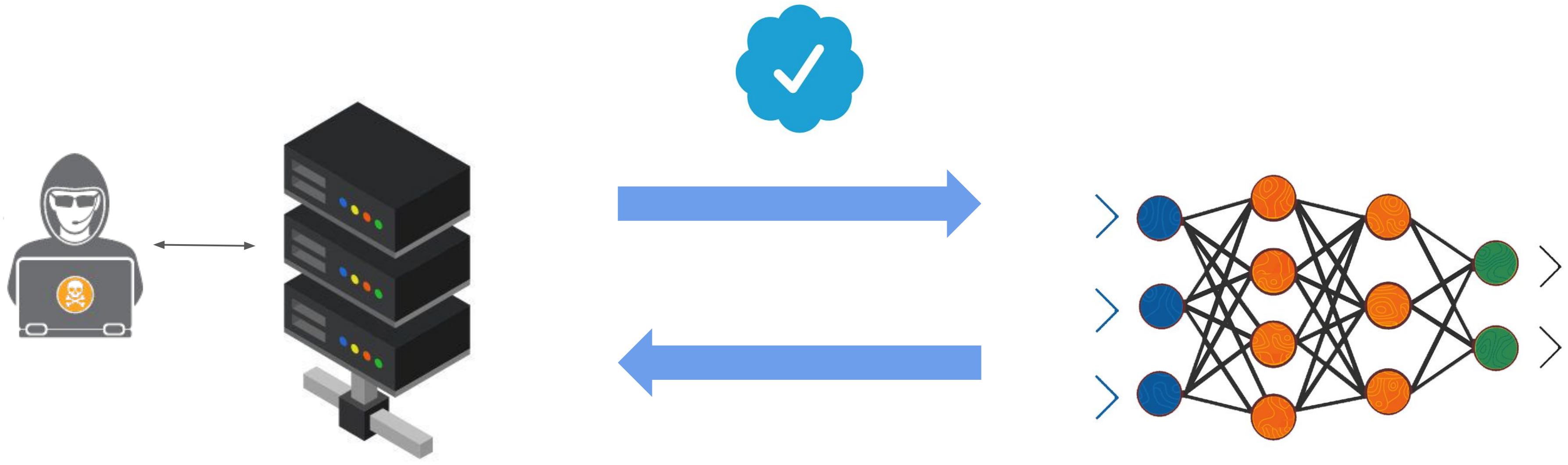
- Could have been subtly poisoned even if it performs reasonably



[1] 3d vector art by vecteezy. Source can be found at: https://www.vecteezy.com/vector-art/1975290-isometric-server-on-white-background
[2] Image from *CNN vs. RNN vs. ANN – Analyzing 3 Types of Neural Networks in Deep Learning* - Analytics Vidhya. The article can be found at https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/

How do you make sure confidentiality wasn't broken?

You often have to use proprietary data



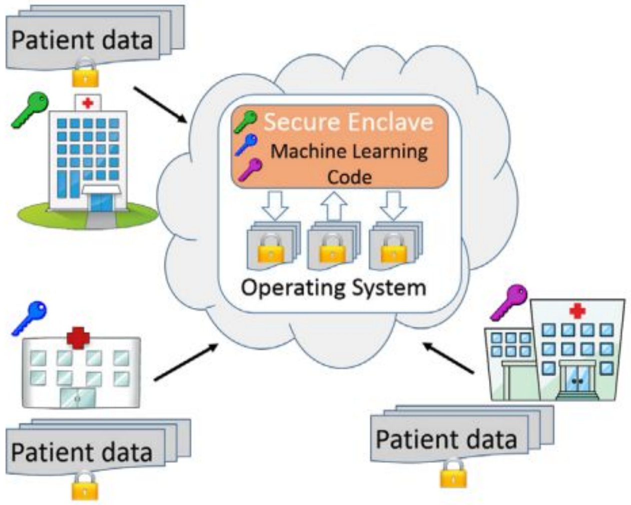
[1] 3d vector art by vecteezy. Source can be found at: https://www.vecteezy.com/vector-art/1975290-isometric-server-on-white-background
[2] Image from *CNN vs. RNN vs. ANN – Analyzing 3 Types of Neural Networks in Deep Learning* - Analytics Vidhya. The article can be found at https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/

General Verification Threat Model

- You requires services from some third party entity
  - Training on the cloud
  - Running inference on your data
- You assume that this third party is the attacker
  - Wish to collect proprietary data, or alter your training in some way
- **Objective:**
  - Get your attacker to perform the service you requested from them and **verify** that this service was in fact completed without accessing anything proprietary

UNIVERSITY OF TORONTO

Oblivious Multi-Party Machine Learning on Trusted Processors

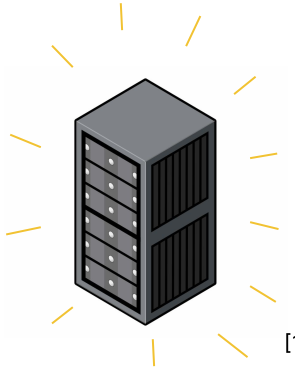


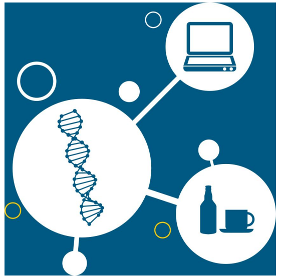
UNIVERSITY OF TORONTO

Olga Ohrimenko, Felix Schuster, and Cédric Fournet, Microsoft Research; Aastha Mehta, Microsoft Research and Max Planck Institute for Software Systems (MPI-SWS); Sebastian Nowozin, Kapil Vaswani, and Manuel Costa, Microsoft Research (2016)

Multiple Data Sources

Congratulations! You have a bioinformatics startup that needs data to begin training, and you don't have enough


[1]

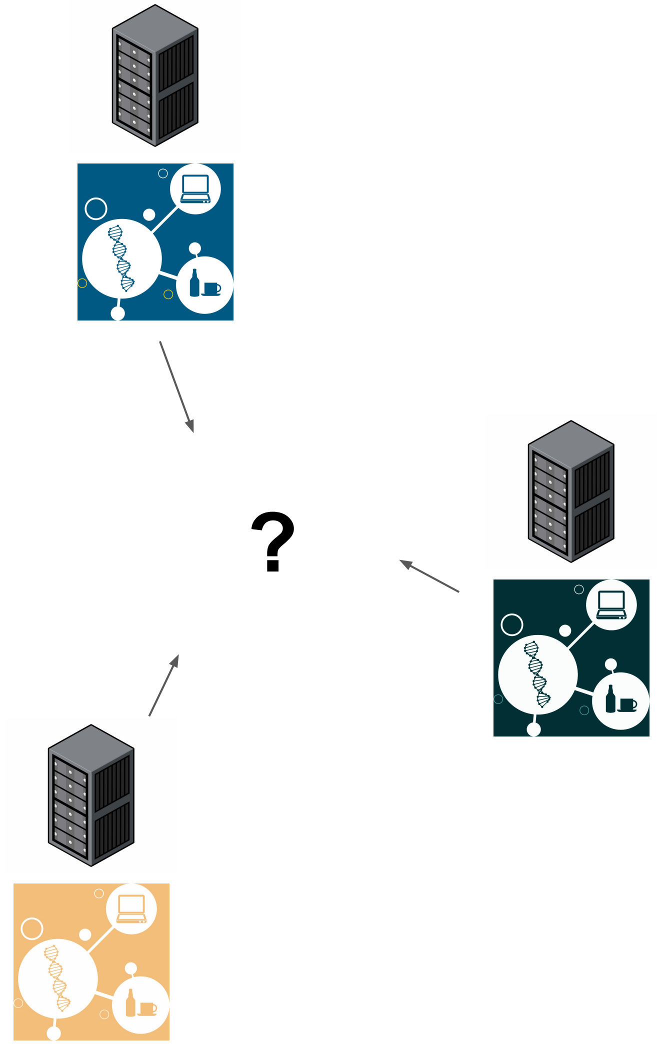

[2]

UNIVERSITY OF TORONTO

[1] 3d vector art by Public domain vector art. Source can be found at: https://publicdomainvectors.org/en/free-clipart/Isometric-server-cabinet-vector-graphics/13444.html
[2] Bioinformatics hackathon 2015. Source can be found at https://bioinformatics.mdc-berlin.de/hackathon2015.html

Multiple Data Sources

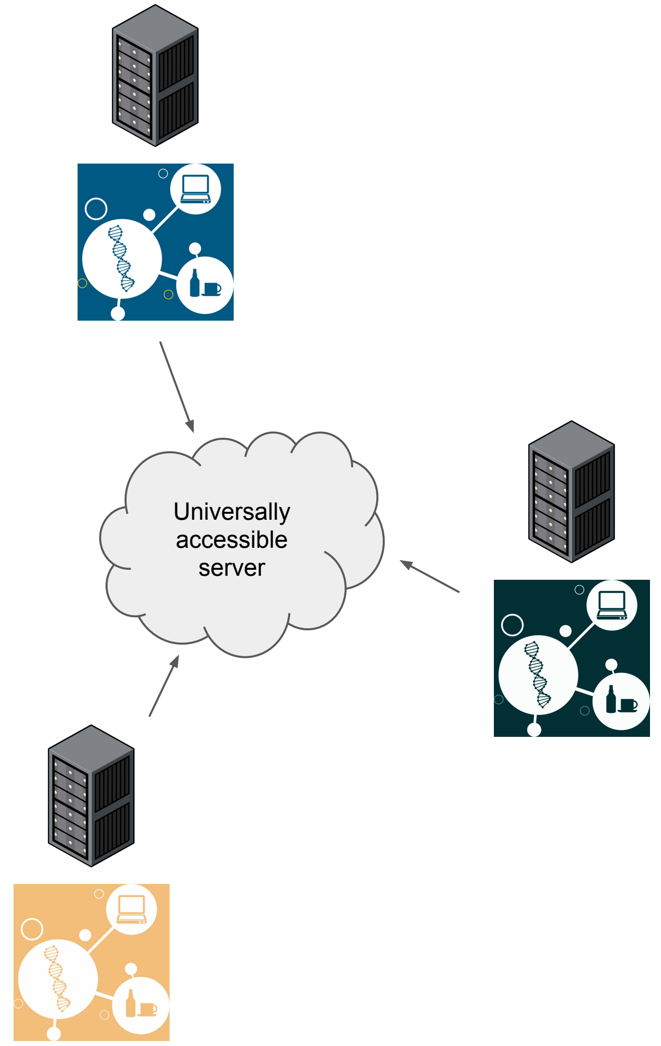
There are other startups that are willing to share their data with you, but no one trusts anyone else



Naive Care-Bear Solution

Everyone gets access to the data!

(the share everything solution)

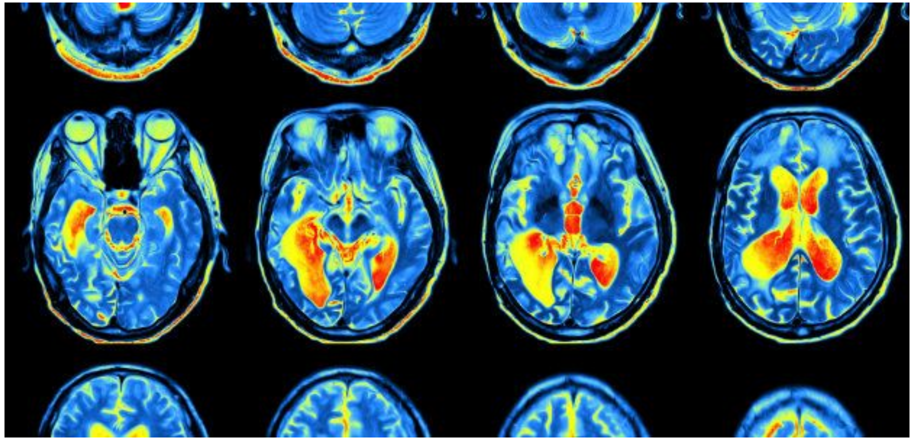


UNIVERSITY OF TORONTO

Real World Data Complications

What if the data is private due to legal or ethical concerns and **can't** be shared under essentially any circumstances?

- Your startup may happen to use the DNA of their clients



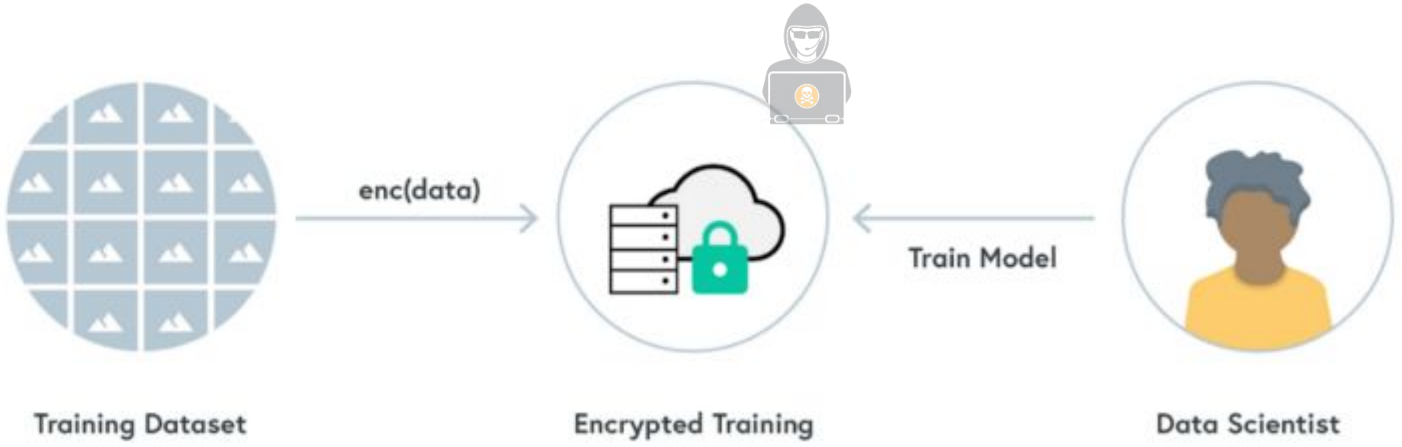
UNIVERSITY OF TORONTO

[1] Image from *CWoznicki Think Out Loud* - cwoznicki.com. The article can be found at https://cwoznicki.com/category/uncategorized/

Cryptographic solutions

Cryptographically encoding the entire dataset works, but fails if someone observes the training process too closely

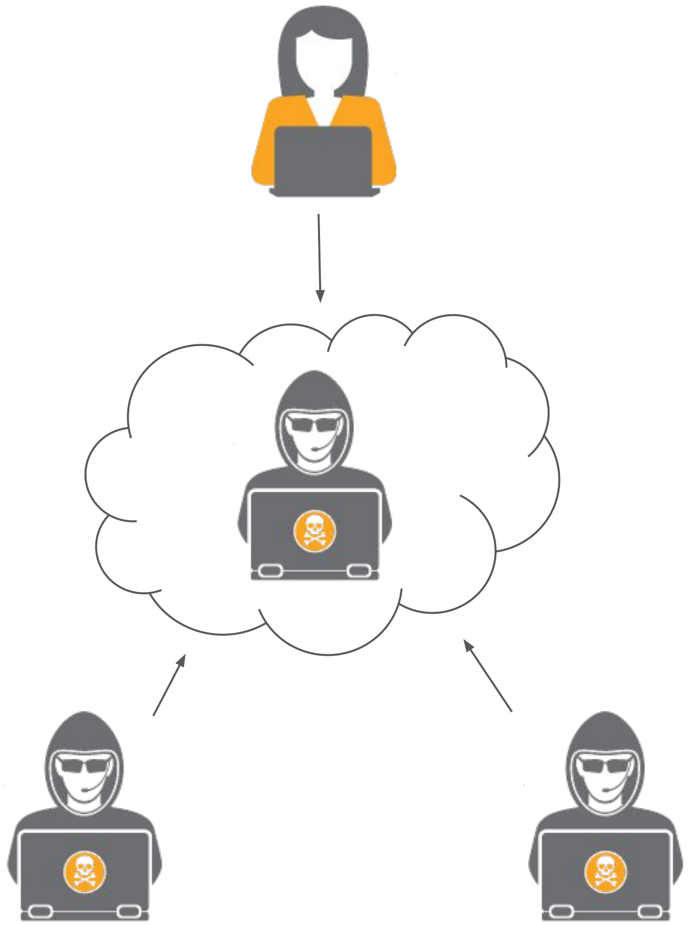
- Observing the training process on the hardware level **after** data decryption



Threat Model

Strong adversary:

- Access to the cloud training provider
- Controls all the hardware (except the actual processor chips)
- Accesses network packets and communications
- Read the physical memory left on the processor chip by probing
- Controls all software at an administrative level
- **Can’t** access processor SGX chips
- **Can’t** use compromised machine learning code

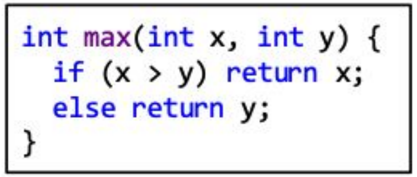


UNIVERSITY OF TORONTO

Data Obliviousness

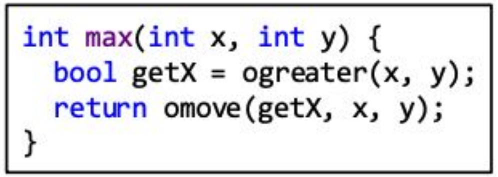
- Internal data remains oblivious to the runner of the code
- Perform operations in a way that does not depend on the contents of the data itself.

Non-oblivious



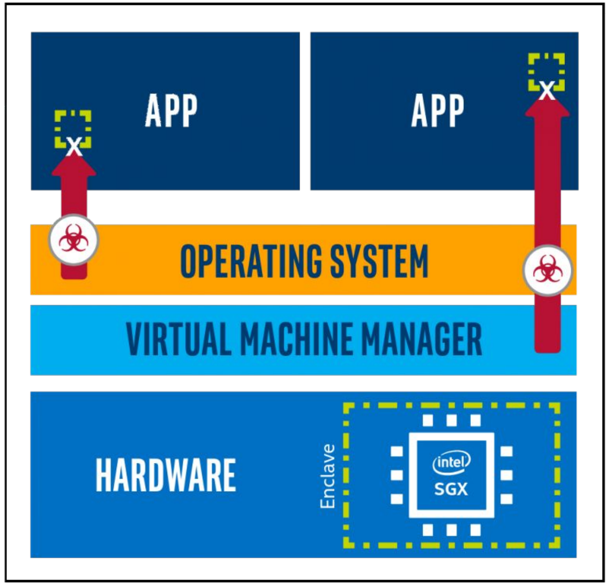
```
int max(int x, int y) {
  if (x > y) return x;
  else return y;
}
```

Oblivious



```
int max(int x, int y) {
  bool getX = ogreater(x, y);
  return omove(getX, x, y);
}
```

First step to a more secure process: Intel SGX



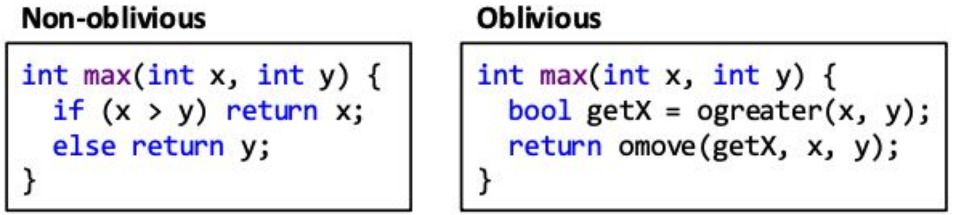
- Allows for the creation of a **Trusted Execution Environment** (TEE)
    - Called an **Enclave**
- Runs protected software and data without disclosure or modification
    - Only protected code can access protected memory
    - Even if OS is compromised
- Guarantees that the code is executing on the protected platform using a hardware level private key

UNIVERSITY OF TORONTO

Next step: Defining the SGX primitives

Define basic operations in Neural Networks in a data oblivious manner:

Max function:



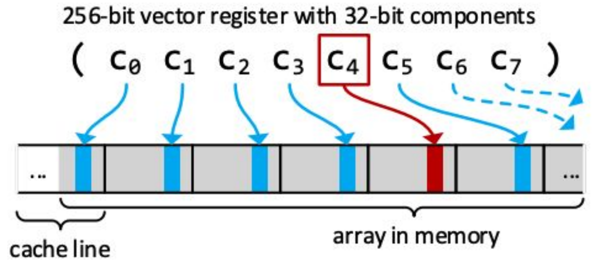
Non-oblivious

```
int max(int x, int y) {
  if (x > y) return x;
  else return y;
}
```

Oblivious

```
int max(int x, int y) {
  bool getX = ogreater(x, y);
  return omove(getX, x, y);
}
```

Array Lookup:



256-bit vector register with 32-bit components
(c_0 c_1 c_2 c_3 c_4 c_5 c_6 c_7)
...
...
cache line
array in memory

UNIVERSITY OF TORONTO

Overall Proposed Process

1) Establish secure authentication based channel with enclave
2) Send encrypted data directly to enclave
3) Use remote attestation to ensure enclave code and data is good
4) Run the agreed upon code

UNIVERSITY OF TORONTO

Algorithm 1: K-means

Issue 1 : Even if the algorithm is largely independent of the data, the final clusters could leak information

Issue 2: Intermediate clusters could also leak information

Solution: maintain current and next centroid so the transition can be done obliviously (does not affect time complexity)

[1] Image from Understanding K-Means Clustering: Hands-on Visual Approach - plainenglish.io. The article can be found at https://ai.plainenglish.io/understanding-k-means-clustering-hands-on-visual-approach-c2dc46f0ed18

Algorithm 2: Supervised Learning

We can’t randomly access the data at each iteration without extra work.

1. Shuffle the data beforehand

2. Run the algorithm sequentially

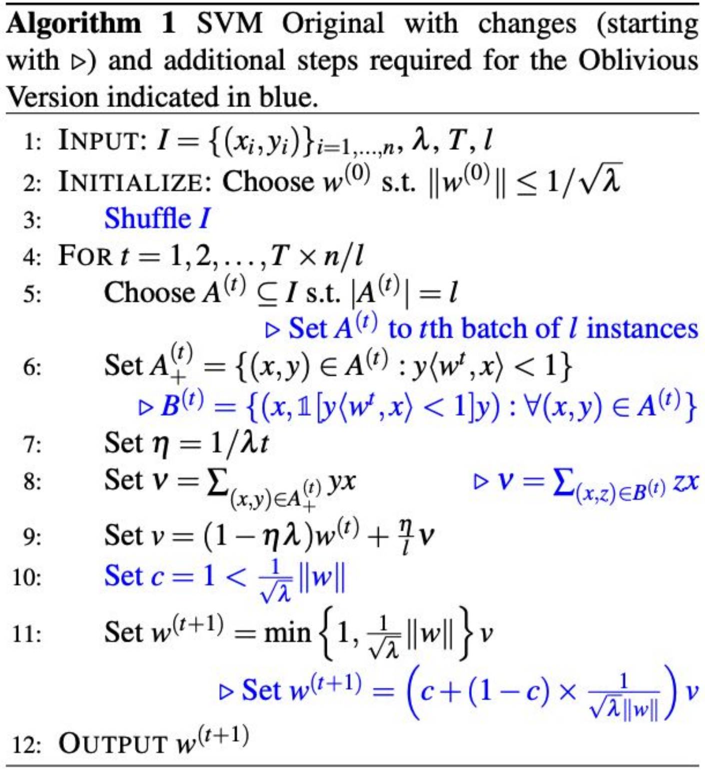
$$\min_{w} \Omega(w) + \frac{1}{n}\sum_{i=1}^{n} L(y_i, f_w(x_i)).$$

UNIVERSITY OF TORONTO

Algorithm 2: Supervised Learning (SVMs)

They generally pre-compute a lot of the steps to keep the data anonymous

$$O(n(\log n)^2)$$


Algorithm 1 SVM Original with changes (starting with ▷) and additional steps required for the Oblivious Version indicated in blue.

```
1:  INPUT: $I = \{(x_i, y_i)\}_{i=1,\ldots,n}, \lambda, T, l$
2:  INITIALIZE: Choose $w^{(0)}$ s.t. $\|w^{(0)}\| \leq 1/\sqrt{\lambda}$
3:      Shuffle $I$
4:  FOR $t = 1, 2, \ldots, T \times n/l$
5:      Choose $A^{(t)} \subseteq I$ s.t. $|A^{(t)}| = l$
                ▷ Set $A^{(t)}$ to $t$th batch of $l$ instances
6:      Set $A_+^{(t)} = \{(x, y) \in A^{(t)} : y\langle w^t, x\rangle < 1\}$
          ▷ $B^{(t)} = \{(x, \mathbb{1}[y\langle w^t, x\rangle < 1]y) : \forall (x, y) \in A^{(t)}\}$
7:      Set $\eta = 1/\lambda t$
8:      Set $v = \sum_{(x,y)\in A_+^{(t)}} yx$          ▷ $v = \sum_{(x,z)\in B^{(t)}} zx$
9:      Set $v = (1 - \eta\lambda)w^{(t)} + \frac{\eta}{l} v$
10:     Set $c = 1 < \frac{1}{\sqrt{\lambda}}\|w\|$
11:     Set $w^{(t+1)} = \min\left\{1, \frac{1}{\sqrt{\lambda}}\|w\|\right\} v$
             ▷ Set $w^{(t+1)} = \left(c + (1 - c) \times \frac{1}{\sqrt{\lambda}\|w\|}\right) v$
12: OUTPUT $w^{(t+1)}$
```

Algorithm 2: Supervised Learning (Neural Networks)

Gradient based optimization

- They actually already process the data in a dense largely oblivious manner
- Replace piecewise approximations of functions like tanh with oblivious “move” operations

$$\nabla_w \Omega(w) + \frac{1}{l} \sum_{i \in S} \nabla_w L(y_i, f(x_i)).$$

UNIVERSITY OF TORONTO

Algorithm 3: Decision Trees

Highly instance specific

Require that evaluation also operate in an oblivious manner

- This does force each decision to take the same number of steps

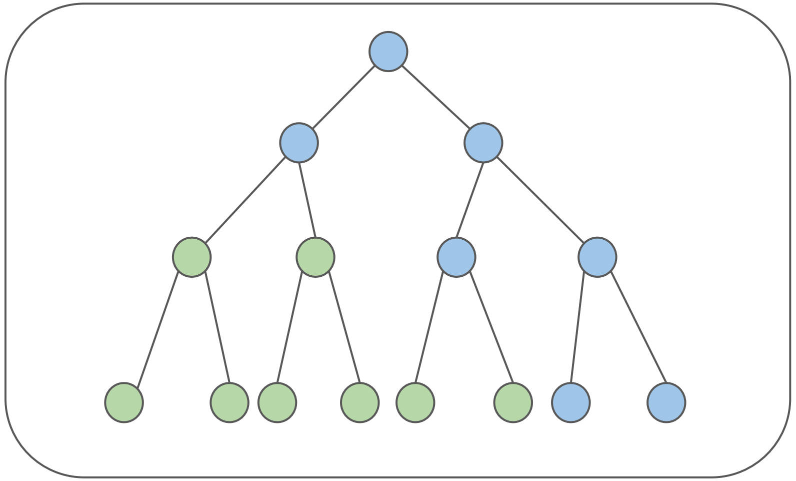
*note that this does imply that the model can never be removed from the SGX cloud instance

Can't be truly interpretable in this case

Decision Stump

$$\phi(x; j, t) = \begin{cases} \text{left,} & \text{if } x(j) \le t, \\ \text{right,} & \text{otherwise,} \end{cases}$$

Blackbox functions



UNIVERSITY OF TORONTO

Algorithm 4: Matrix Factorization

Ratings	User	Items
$R \in \mathbb{R}^{n\times m}$	$U \in \mathbb{R}^{n\times d}$	$V \in \mathbb{R}^{m\times d}$

Output vectors usually have some semantic meaning to them that comes from the input data

$$R \approx U V^\top$$

$$\min \frac{1}{M} \sum (r_{i,j} - \langle u_i, v_j \rangle)^2 + \lambda \sum \|u_i\|_2^2 + \mu \sum \|v_j\|_2^2 \quad (4)$$

Consider a basic rating task:

$$u_i^{(t+1)} \leftarrow u_i^{(t)} + \gamma \left[\sum_j e_{i,j} v_j^{(t)} - \lambda u_i^t \right]$$
$$v_j^{(t+1)} \leftarrow v_j^{(t)} + \gamma \left[\sum_i e_{i,j} u_i^{(t)} - \mu v_j^t \right]$$

Issue: We can't reveal which people correspond to which items

UNIVERSITY OF TORONTO

Algorithm 4: Matrix Factorization

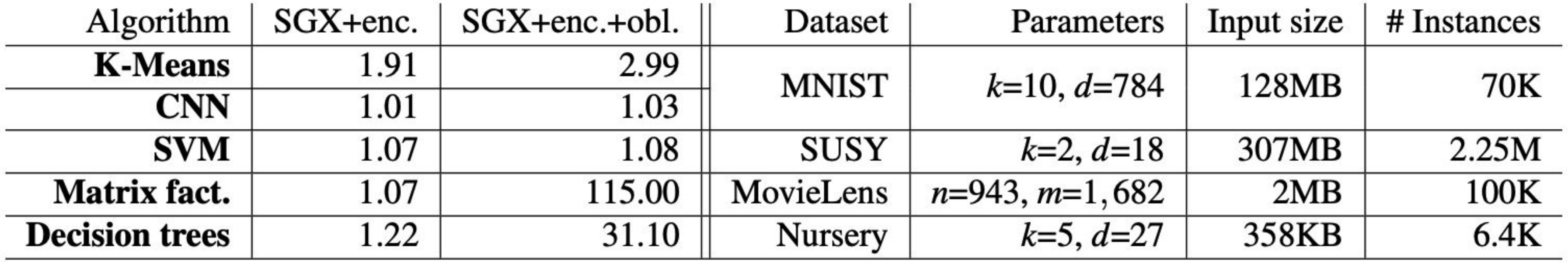
Idea 1: Change data to universally sized tuples with all the information needed

- *Update* the user profiles u_i within $\mathbf{U}^{(t)}$ using Equation (5); let $\tilde{\mathbf{U}}$ be the updated data structure;
- *Extract* $U^{(t+1)}$ from $\tilde{\mathbf{U}}$;
- *Copy* $U^{(t+1)}$ into the rating tuples of $\tilde{\mathbf{V}}$ to obtain $\mathbf{V}^{(t+1)}$ for the next iteration.

Rating Tuples:	$(i, j, r_{i,j}, v_j)$
User Tuples:	$(i, 0, 0, u_i)$
Item Tuples:	$(0, j, 0, v_j)$

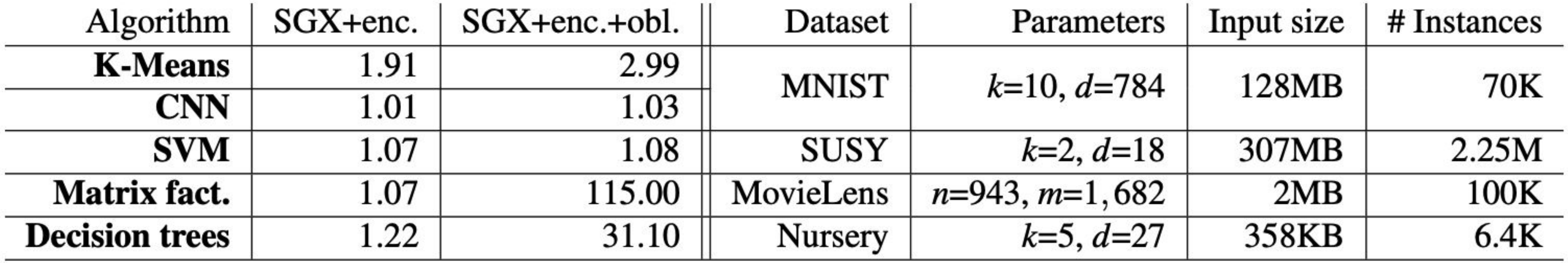
UNIVERSITY OF TORONTO

Results (Runtime Relative to no Protection)



Algorithm	SGX+enc.	SGX+enc.+obl.	Dataset	Parameters	Input size	# Instances
K-Means	1.91	2.99	MNIST	k=10, d=784	128MB	70K
CNN	1.01	1.03				
SVM	1.07	1.08	SUSY	k=2, d=18	307MB	2.25M
Matrix fact.	1.07	115.00	MovieLens	n=943, m=1,682	2MB	100K
Decision trees	1.22	31.10	Nursery	k=5, d=27	358KB	6.4K

Results (Runtime Relative to no Protection)



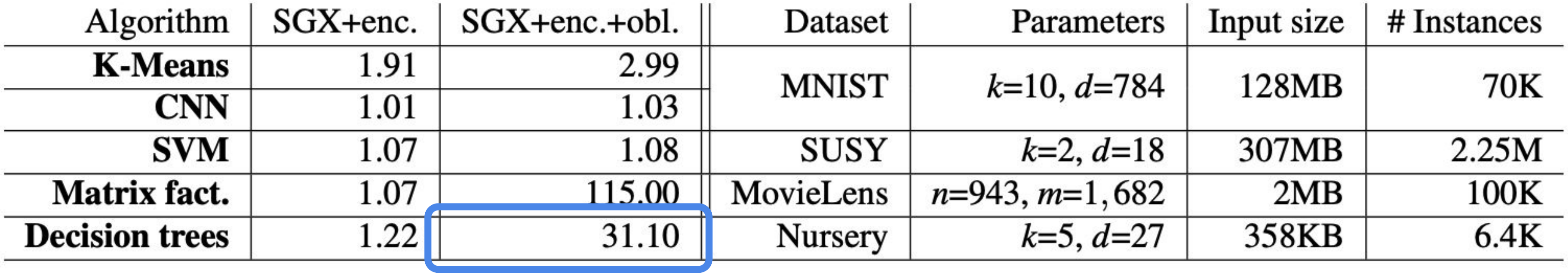
Algorithm	SGX+enc.	SGX+enc.+obl.	Dataset	Parameters	Input size	# Instances
K-Means	1.91	2.99	MNIST	k=10, d=784	128MB	70K
CNN	1.01	1.03				
SVM	1.07	1.08	SUSY	k=2, d=18	307MB	2.25M
Matrix fact.	1.07	115.00	MovieLens	n=943, m=1,682	2MB	100K
Decision trees	1.22	31.10	Nursery	k=5, d=27	358KB	6.4K

Only analyzed the increase for training, not for execution time

Doesn’t necessarily work with their paradigm for some algorithms

UNIVERSITY OF TORONTO

Results (Runtime Relative to no Protection)



Algorithm	SGX+enc.	SGX+enc.+obl.	Dataset	Parameters	Input size	# Instances
K-Means	1.91	2.99	MNIST	k=10, d=784	128MB	70K
CNN	1.01	1.03				
SVM	1.07	1.08	SUSY	k=2, d=18	307MB	2.25M
Matrix fact.	1.07	115.00	MovieLens	n=943, m=1,682	2MB	100K
Decision trees	1.22	31.10	Nursery	k=5, d=27	358KB	6.4K

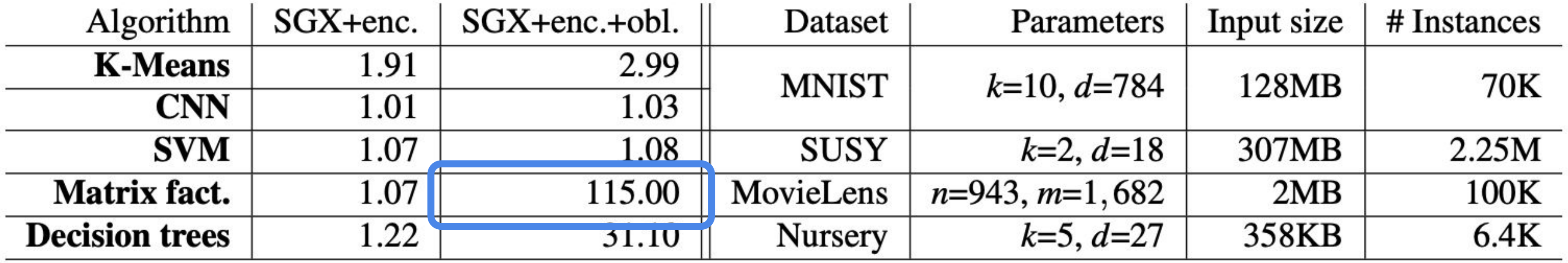
31.1x slower to add data oblivious guarantees

Largely due to the oblivious array lookup and scanning

Doubling to tripling the tree depth increases relative runtime by 63.16x

UNIVERSITY OF TORONTO

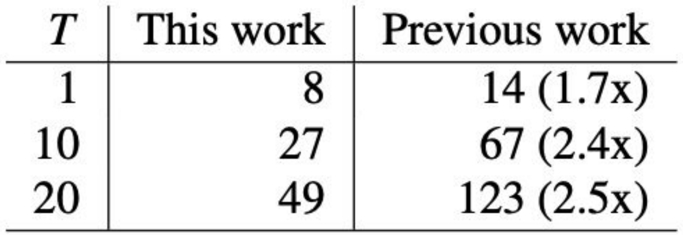
Results (Runtime Relative to no Protection)



Algorithm	SGX+enc.	SGX+enc.+obl.	Dataset	Parameters	Input size	# Instances
K-Means	1.91	2.99	MNIST	k=10, d=784	128MB	70K
CNN	1.01	1.03				
SVM	1.07	1.08	SUSY	k=2, d=18	307MB	2.25M
Matrix fact.	1.07	115.00	MovieLens	n=943, m=1,682	2MB	100K
Decision trees	1.22	31.10	Nursery	k=5, d=27	358KB	6.4K

Huge change in matrix factorization runtime

Better than previous work (from 2013)



T	This work	Previous work
1	8	14 (1.7x)
10	27	67 (2.4x)
20	49	123 (2.5x)

General Summary

Summary of the Proposed Method

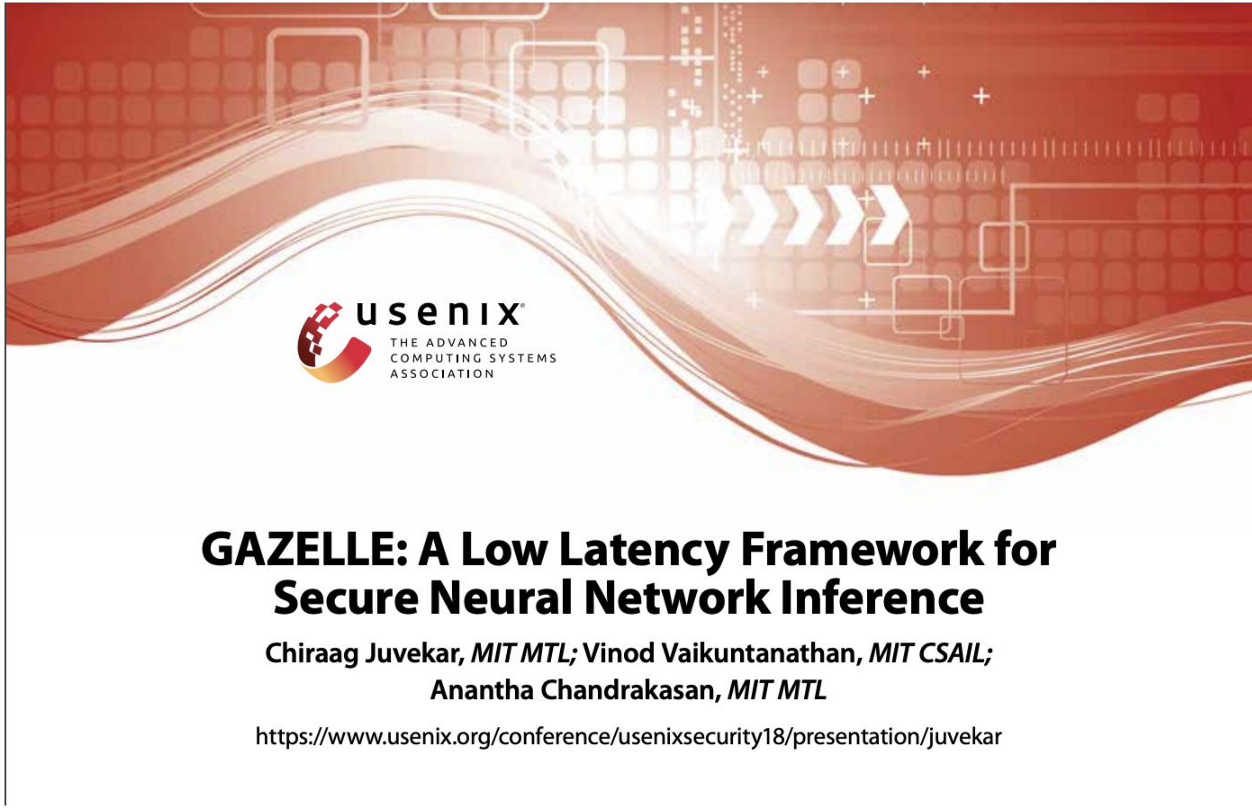
- It is able to perform the data oblivious training for a variety of methods
- It keeps the time complexity of it's algorithms where other oblivious methods may not
- Works under very strong assumptions

Limitations

- Requires some of the algorithms be run on the enclave during inference to preserve data privacy
- Has some significant overhead on some of the algorithms due to array lookup
- Requires some hardware to be inaccessible to any physical attacks for this method to be safe
- To be extended, an additional method must be manually created for every new method, and therefore there will always be some implementation overhead

UNIVERSITY OF TORONTO

Paper 2



usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

GAZELLE: A Low Latency Framework for Secure Neural Network Inference

Chiraag Juvekar, *MIT MTL;* Vinod Vaikuntanathan, *MIT CSAIL;* Anantha Chandrakasan, *MIT MTL*

https://www.usenix.org/conference/usenixsecurity18/presentation/juvekar

UNIVERSITY OF TORONTO

Outline of the talk

1. Problem motivation and overview of Gazelle
2. Homomorphic encryption
3. Garbled Circuits
4. Protocol
5. Results
6. Discussion

Motivation

- Increasingly popular to offload machine learning inference to 3rd party
  - Releasing model to clients doesn’t please the shareholders
  - Exposes training data to privacy risk (membership inference)
- However, prediction-as-a-service exposes privacy and liability risks
- Risk to client:
  - Risk of showing sensitive information to 3rd party
- Risk to server:
  - Liability issues in protecting privacy of users

Example areas of concern

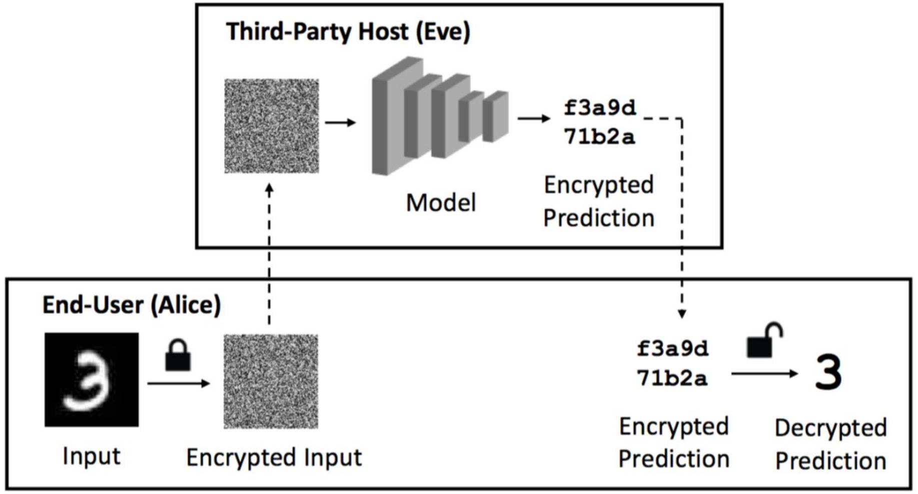
- Medical diagnosis
- Facial recognition
- Credit risk assessment
- Bail assessment
- ...

UNIVERSITY OF TORONTO

Secure neural network inference

- Client: the end-user who queries the API
- Server: the third-party host who trained and owns the model

- Model is trained by the third party unencrypted, then converted
- Model is private to client
- Data and outcome is private to server



UNIVERSITY OF TORONTO

Prior works

- CryptoNets
  - HE only - non-linear layers take 500 seconds
- Secure ML and MiniONN
  - HE not actually used for linear algebra
- DeepSecure
  - GC only - binary circuits generated for linear layers are 400x larger

Problems:

- Prohibitively high runtime during inference (5 minutes / example)
- Prohibitively large bandwidth during inference between the server and the client (372 MB / example)

UNIVERSITY OF TORONTO

Overview of Gazelle

1. Efficient, low-bandwidth secure linear algebra
2. Efficient, low-bandwidth secure non-linear layer
3. Protocol to efficiently switch between the linear algebra and non-linear libraries
4. 2-party computation (no 3rd party required)

Results in 5-30x faster inference, and 9-750x lower bandwidth required for CNNs

UNIVERSITY OF TORONTO

Gazelle threat model

	A (server)	B (client)
Exposes	Network structure, # layers, # of hidden nodes	Image size
Hides	Weights, biases, filter and stride sizes, layer types	Image contents, label

Why optimize differently for linear vs. non-linear?

Homomorphic encryption is used for linear algebra

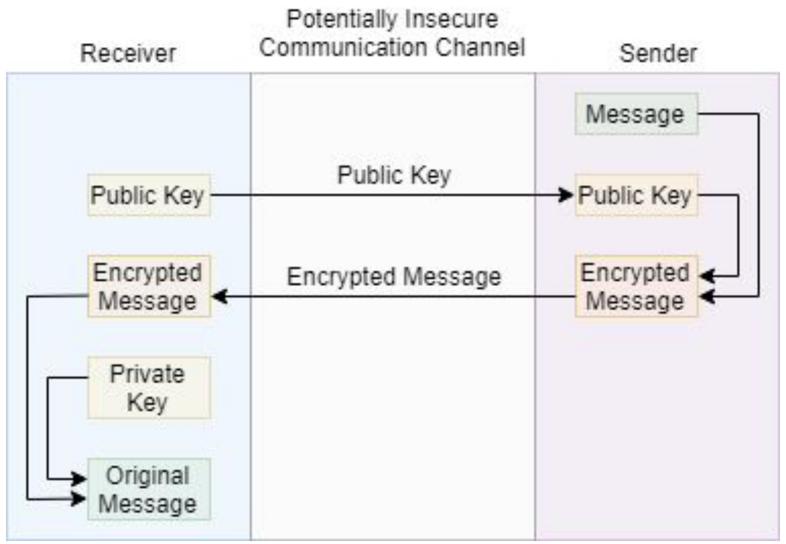
- better for operations that scale quickly with the input
- have low composition of operations

Garbled circuits is used for non-linear layers

- HE struggles with high multiplicative depth operations (non-linear functions)
- A Quick Garbled Circuits Primer

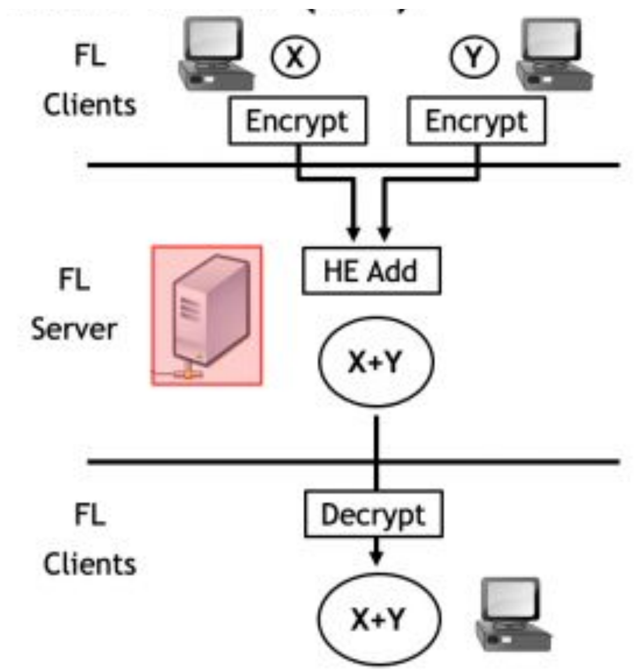
Homomorphic Encryption

How does encryption work?



Homomorphic encryption for secure linear algebra

Given encrypted x, compute encrypted f(x) without decrypting the input



UNIVERSITY OF TORONTO

Homomorphic encryption for secure linear algebra

Example with unpadded RSA encryption:

Given message *m*, modulus *n* and exponent *e*: $\mathcal{E}(m) = m^e \mod n.$

This allows us to do multiplication directly on the encrypted messages, and still allow for correct decoding:

$$\begin{aligned} \mathcal{E}(m_1) \cdot \mathcal{E}(m_2) &= m_1^e m_2^e \mod n \\ &= (m_1 m_2)^e \mod n \\ &= \mathcal{E}(m_1 \cdot m_2) \end{aligned}$$

Homomorphic encryption for secure linear algebra

Specifically, they used packed additively homomorphic encryption (PAHE)

Define 3 elementary operations:

- SIMD add (between vectors)
- SIMD scalar multiplication (between vector and scalar/vector)
- Permutations of a vector

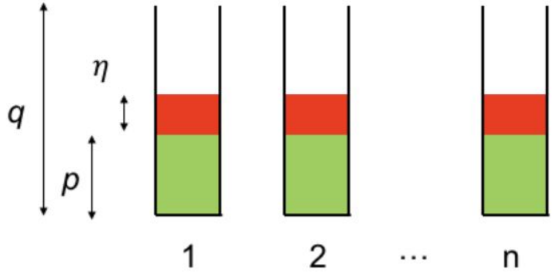
Note: SIMD = single-instruction, multiple-data

UNIVERSITY OF TORONTO

Parameters for homomorphic encryption

- Cyclotomic order m
- Ciphertext modulus q
  - Ciphertext space ≥ ⌈$\log_2 q$⌉ bits
- Plaintext modulus p
  - Plaintext space is ≤ ⌊$\log_2 p$⌋ bits
- Standard deviation σ

Used in RSA encryption

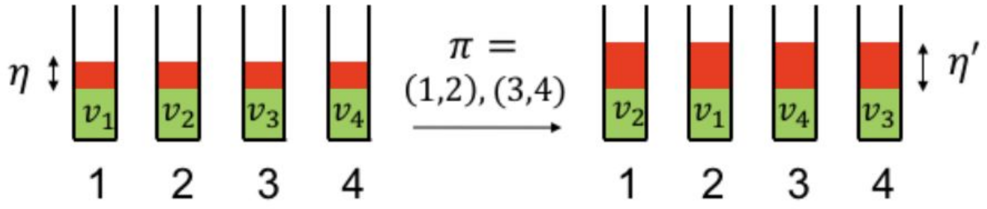


Other variables:

- Noise in cipher text: η
- Number of slots: n = ϕ(m)
  - ϕ(m) = # of positive integers k ≤ m and relatively prime with m

Cost for elementary operations

	Variables	Noise growth	Run-time
SIMD add	[u] + [v] → [u+v]	$\eta_u + \eta_v$	n * CostAdd(q)
SIMD scalar multiply	[u] * v → [u * v]	$\eta_u + \eta_{mult}$	n * CostMult(q)
Permutation	[u] → [u_π]	$\eta_u + \eta_{rot}$	Θ(n log n log q)

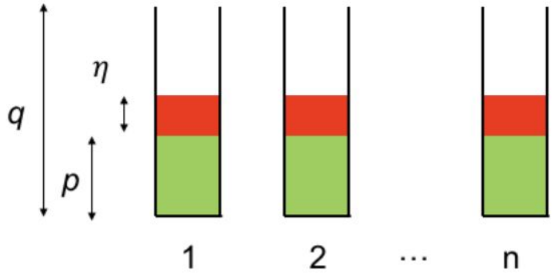


Example of permutation

Requirements on HE parameters

TLDR: we need to select parameters and operations such that noise is minimized,

so that decrypting will be correct



- For decoding correctness:
  - $|\eta| < q/(2p)$
- Guarantee security:
  - minimum bound of σ
  - $q \equiv 1 \bmod m$
  - $\gcd(p, q) = 1$
- Minimize growth of noise:
  - Minimize $r \equiv q \bmod p \rightarrow r = \pm 1$
- Support element-wise SIMD product
  - $p \equiv 1 \bmod m$

Parameter selection for HE

TDLR: We choose a combination of p, q, m, n that allow us to do HE operations

- Set p near 2^{20}
- Set q near 2^{60} to fit inside a word (8 bytes)
- Set m = 4096 = 2^{12} for fast permutation (but only half rotations allowed)
- Set σ = 4 to get target security level of 128 bits
- Set n = 2048

Examples:

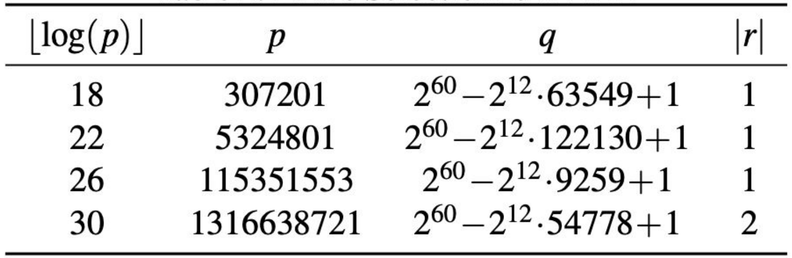


Table 1: Prime Selection for PAHE

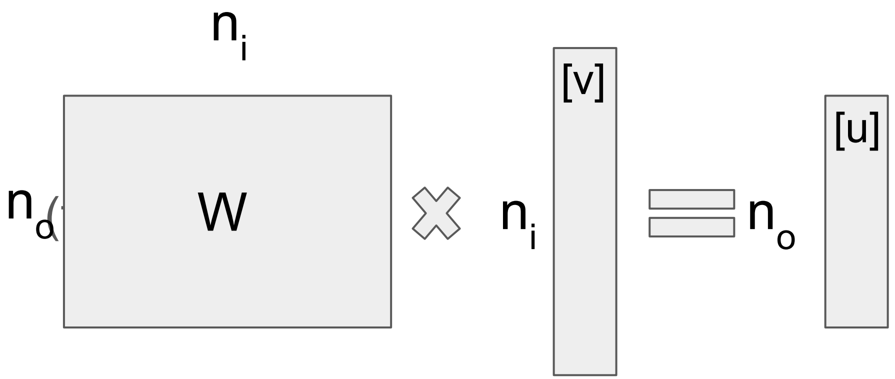
$\lfloor \log(p) \rfloor$	p	q	$\lvert r \rvert$
18	307201	$2^{60} - 2^{12} \cdot 63549 + 1$	1
22	5324801	$2^{60} - 2^{12} \cdot 122130 + 1$	1
26	115351553	$2^{60} - 2^{12} \cdot 9259 + 1$	1
30	1316638721	$2^{60} - 2^{12} \cdot 54778 + 1$	2

Fast homomorphic matrix-vector multiplication

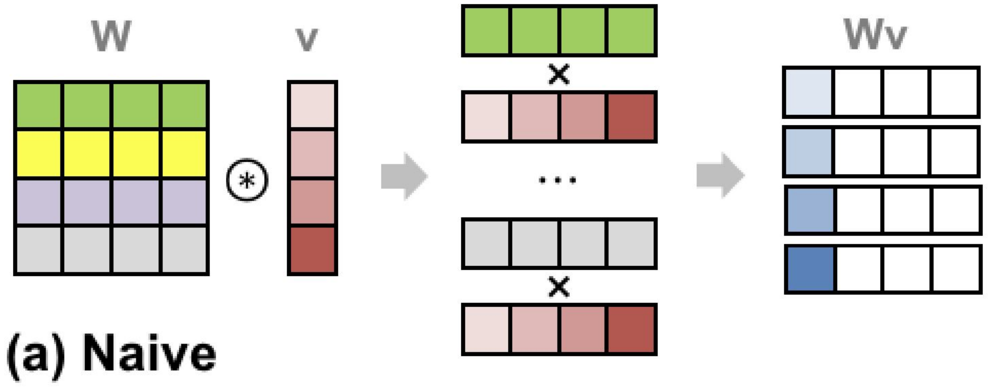
Fully-connected layers use this

What we want:

- Low noise increase
- Few operations
- Low number of output ciphertexts reduce bandwidth)
- No information leaked

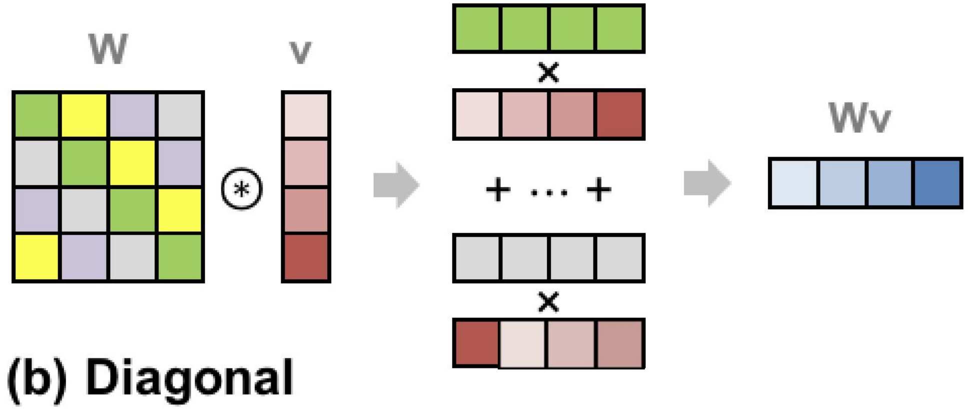


Naive mat-vec multiplication



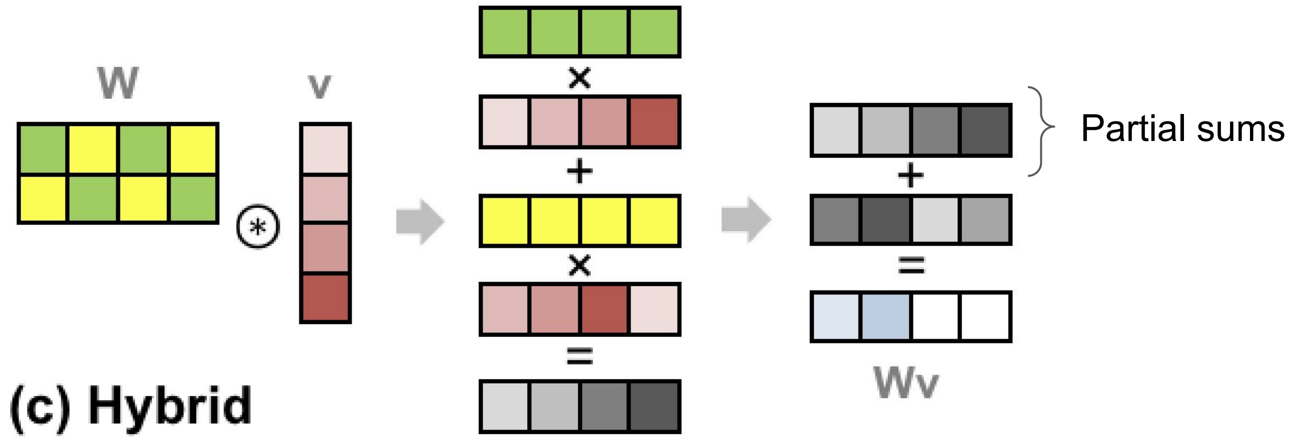
- Treat matrix as a collection of rows
- Large overhead needed to sum the element-wise product (quadratic network bandwidth with size), and destroy extraneous information

Diagonal mat-vec multiplication



- Use diagonals so that summing within a ciphertext is not required
- Can do vector sums between ciphertexts to get single ciphertext output
- Requires n_i perms, which is costly since usually $n_i > n_o$

Hybrid mat-vec multiplication

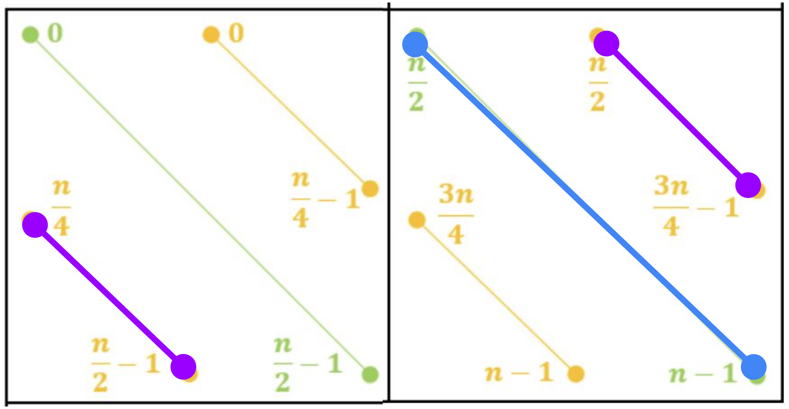


- Do n_o rotations instead of n_i
- Do input packing (not pictured)
- Rotate-and-sum at end like in naive method

Visual difference between diagonal and hybrid

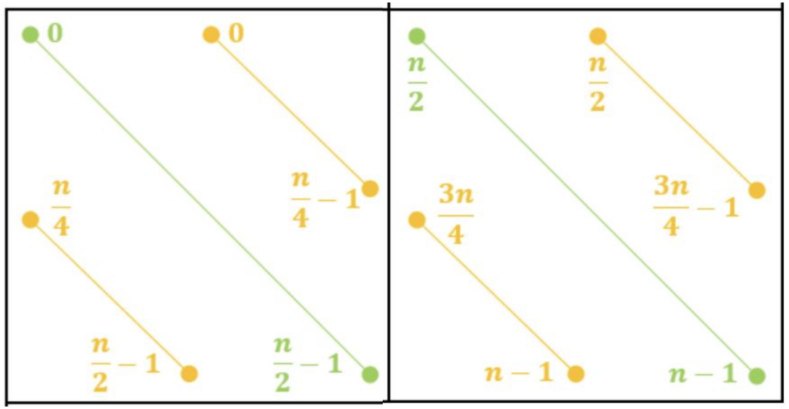
Diagonal

- produces n_i rows of length n_o
- requires input vector to rotate n_i times



Hybrid

- produces n_o rows of length n_i
- requires input vector to rotate n_o times



Comparison of matrix-vector product algorithms

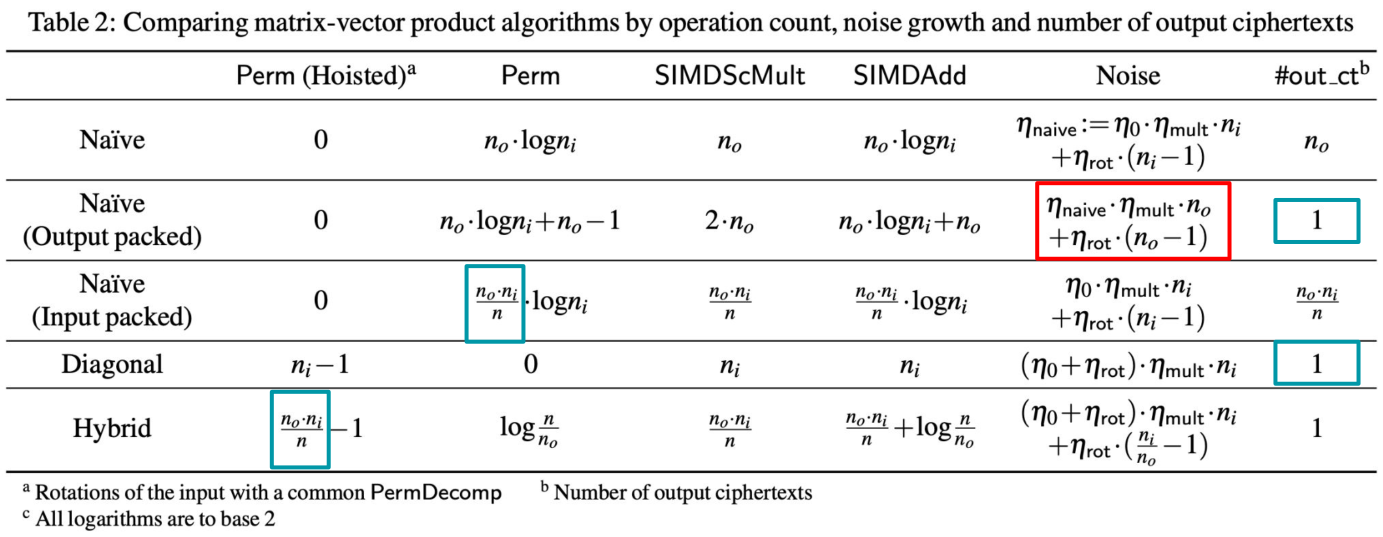


Table 2: Comparing matrix-vector product algorithms by operation count, noise growth and number of output ciphertexts

	Perm (Hoisted)[a]	Perm	SIMDScMult	SIMDAdd	Noise	#out_ct[b]
Naïve	0	$n_o \cdot \log n_i$	n_o	$n_o \cdot \log n_i$	$\eta_{\text{naive}} := \eta_0 \cdot \eta_{\text{mult}} \cdot n_i + \eta_{\text{rot}} \cdot (n_i - 1)$	n_o
Naïve (Output packed)	0	$n_o \cdot \log n_i + n_o - 1$	$2 \cdot n_o$	$n_o \cdot \log n_i + n_o$	$\boxed{\eta_{\text{naive}} \cdot \eta_{\text{mult}} \cdot n_o + \eta_{\text{rot}} \cdot (n_o - 1)}$	$\boxed{1}$
Naïve (Input packed)	0	$\boxed{\frac{n_o \cdot n_i}{n}} \cdot \log n_i$	$\frac{n_o \cdot n_i}{n}$	$\frac{n_o \cdot n_i}{n} \cdot \log n_i$	$\eta_0 \cdot \eta_{\text{mult}} \cdot n_i + \eta_{\text{rot}} \cdot (n_i - 1)$	$\frac{n_o \cdot n_i}{n}$
Diagonal	$n_i - 1$	0	n_i	n_i	$(\eta_0 + \eta_{\text{rot}}) \cdot \eta_{\text{mult}} \cdot n_i$	$\boxed{1}$
Hybrid	$\boxed{\frac{n_o \cdot n_i}{n}} - 1$	$\log \frac{n}{n_o}$	$\frac{n_o \cdot n_i}{n}$	$\frac{n_o \cdot n_i}{n} + \log \frac{n}{n_o}$	$(\eta_0 + \eta_{\text{rot}}) \cdot \eta_{\text{mult}} \cdot n_i + \eta_{\text{rot}} \cdot (\frac{n_i}{n_o} - 1)$	1

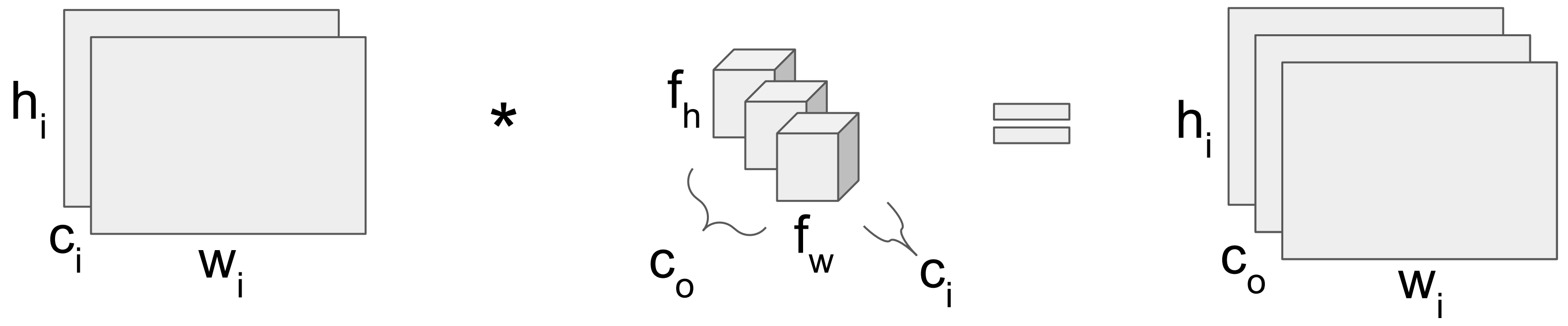
[a] Rotations of the input with a common PermDecomp [b] Number of output ciphertexts
[c] All logarithms are to base 2

Note: In CIFAR, $n_o = n_i = 32$, whereas $n = 2048$

Fast Homomorphic Convolutions

Convolutional layers use same-padding

First, let's give examples with single-input, single-output (SISO), so $c_i = c_o = 1$



Ciphertext input	Plaintext filters/weights	Ciphertext output

Padded SISO convolution

- Fit padded image into single ciphertext
- Do this $f_w * f_h$ times:
  - Rotate the image
  - Do multiplication with scalar
  - Add to current partial sum
- Replace padding on output with noise

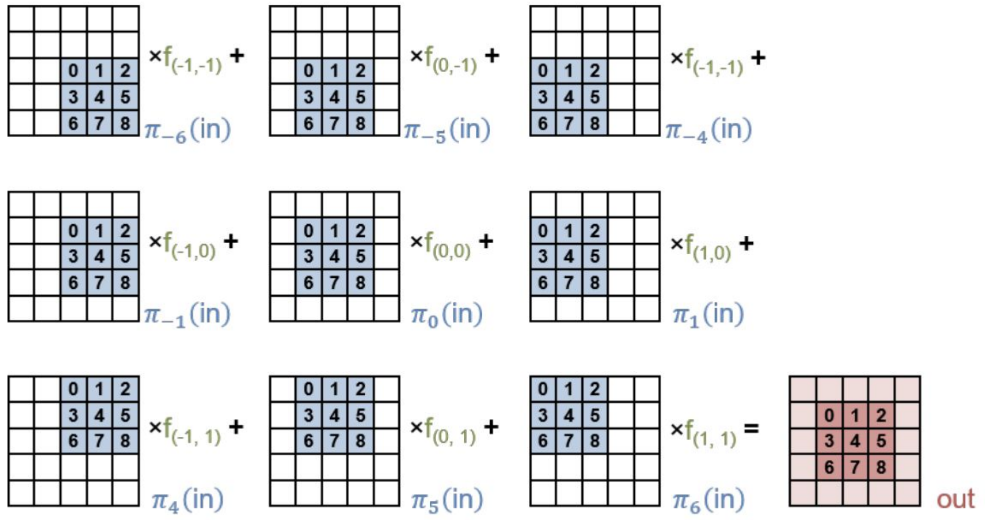


Figure 8: Padded SISO Convolution

Packed SISO convolution

- Rotating the padding is wasteful, so rotate and crop the filter instead
- Do this $f_w * f_h$ times:
  - Rotate the image
  - Rotate the filter
  - Do vector multiplication
  - Add to current partial sum

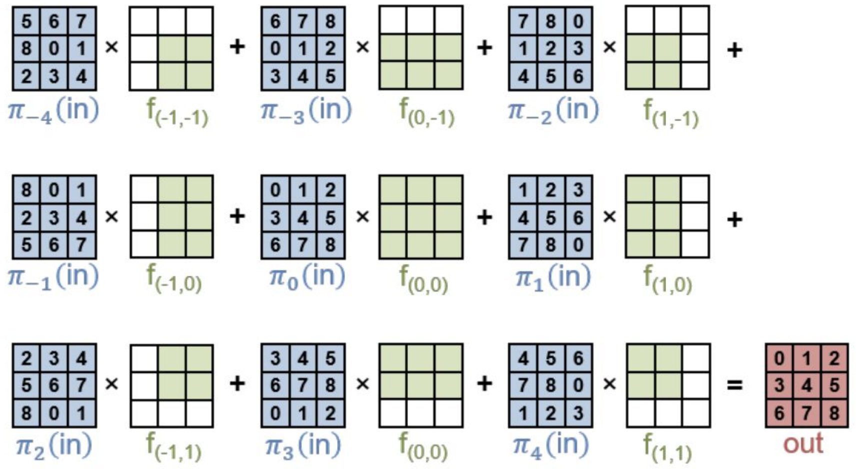


Figure 9: Packed SISO Convolution. (Zeros in the punctured plaintext shown in white.)

Naive multi-channel convolution

- Put each input channel into its own ciphertext → c_i ciphertexts
- Do this c_o times:
  - Do SISO convolution on each input channel
  - Sum over ciphertexts to get an output ciphertext
- Result: c_o ciphertexts

Channel packing for multi-channel convolution

- Similar to diagonal method with mat-vec products, but with entire SISO convs

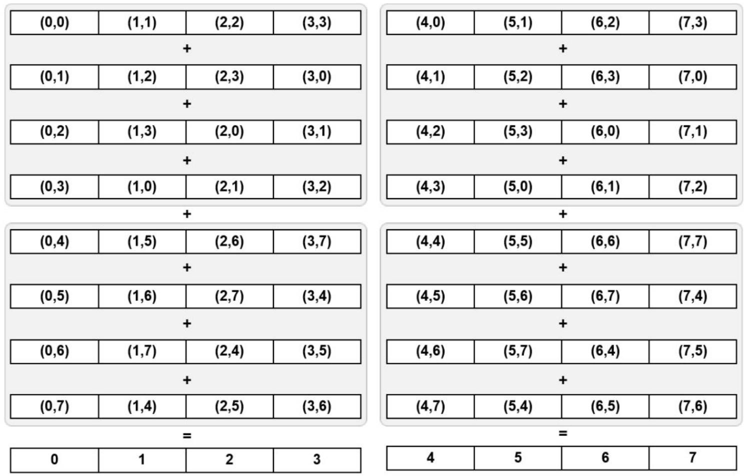


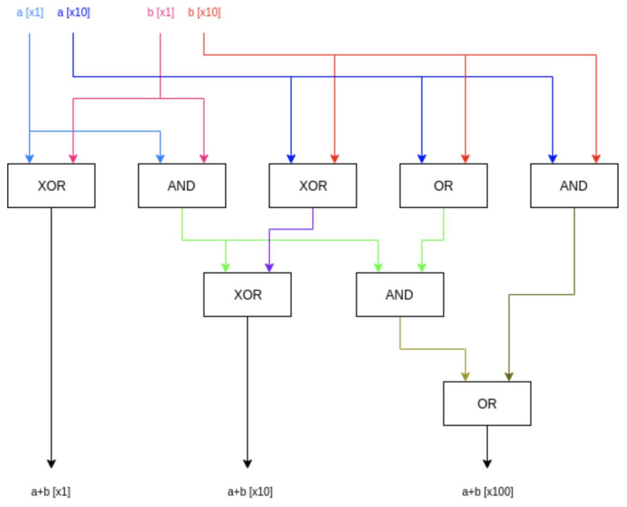
Figure 11: Diagonal Grouping for Intermediate Ciphertexts ($c_i = c_o = 8$ and $c_n = 4$)

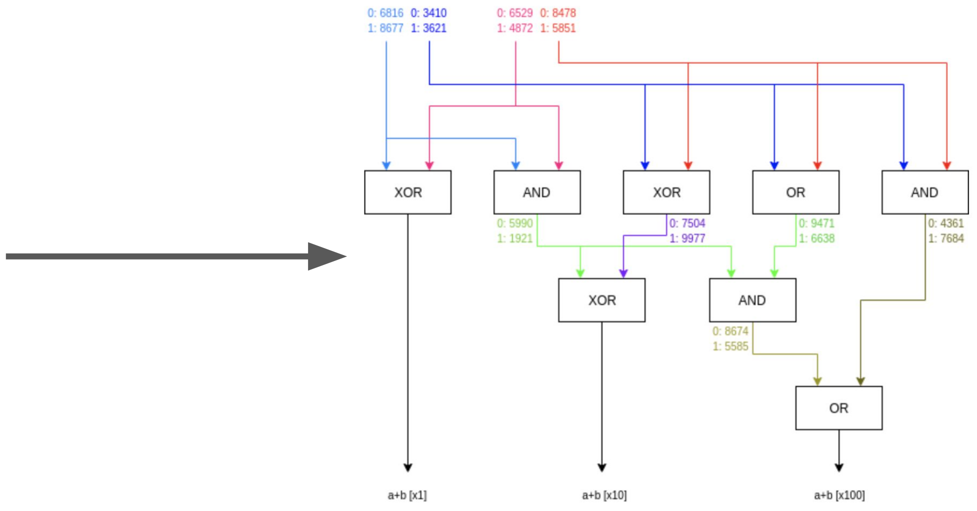
Garbled Circuits

Garbled Circuits: Basics

Example of secure multi-party computation:

- Yao's Millionaires' Problem: if two people want to find out who's richer without disclosing their assets, how can they do it?





Garbled circuits for two-party communication

Authors choose:

- Yao's Garbled Circuit
- Justgarble + half-gates
- Oblivious transfer uses IKNP

Garbling the circuit only depends on NN topology, not client input - so they can do it offline

Binary circuit of non-linear layer is large, and needs to be communicated between parties, which creates large overhead

Protocol

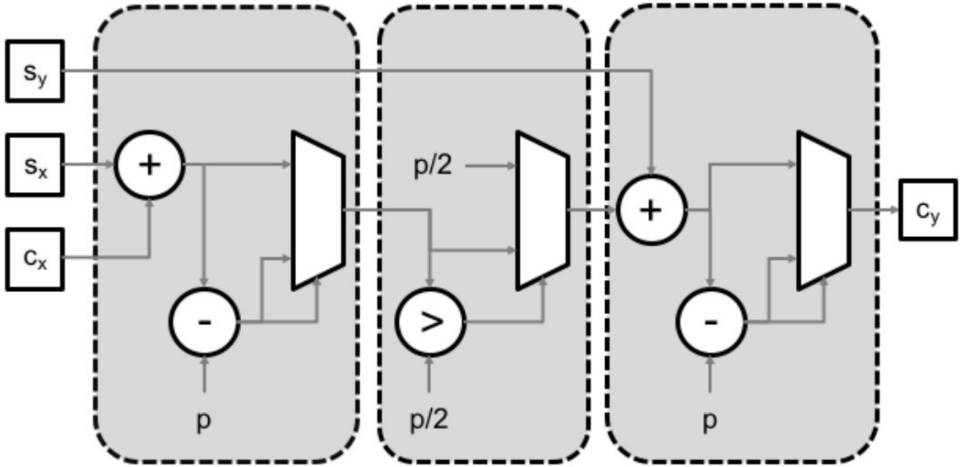
Combining HE and GC for a single conv layer

1. Start with additive shares of **y**, (c_y, s_y), initially (**y**, 0)
2. Client encrypts c_y to get $[c_y]$
3. Server adds it with s_y to get [**y**]
4. Server evaluates linear layer to get [**x**]
5. Server randomly generations **r** and sends [**x + r**] to client
6. Now, server has share s_x = **r**, client decrypts to get share c_x = **r + x** mod p
7. Compute ReLU on these shares using GC to get shares of **y,** (s_y, c_y)
8. Repeat!

Specific protocol for non-linear layers

This circuit computes the ReLU function

1. First block adds $s_x + c_x$ to get y mod p
2. Second block computes ReLU
3. Third block adds randomly generated share s_y to output c_y



Experimental Results

Piecewise experiments

Setup: CPU-focused cloud server (AWS) with LAN internet

- Arithmetic
    - Gazelle achieves faster elementary operations (adding, multiplying, permutations)

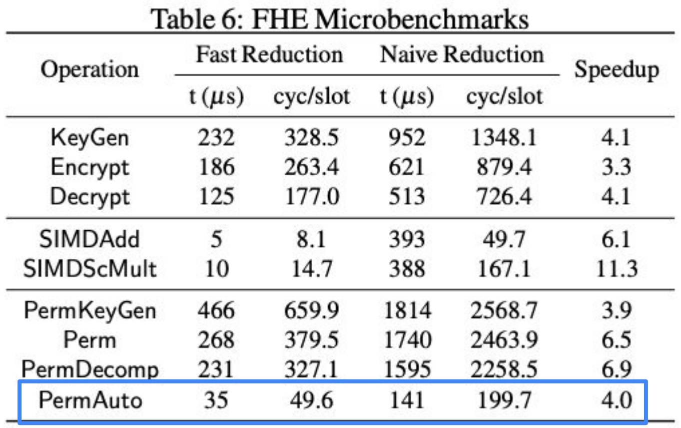


Table 6: FHE Microbenchmarks

Operation	Fast Reduction		Naive Reduction		Speedup
	t (μs)	cyc/slot	t (μs)	cyc/slot	
KeyGen	232	328.5	952	1348.1	4.1
Encrypt	186	263.4	621	879.4	3.3
Decrypt	125	177.0	513	726.4	4.1
SIMDAdd	5	8.1	393	49.7	6.1
SIMDScMult	10	14.7	388	167.1	11.3
PermKeyGen	466	659.9	1814	2568.7	3.9
Perm	268	379.5	1740	2463.9	6.5
PermDecomp	231	327.1	1595	2258.5	6.9
PermAuto	35	49.6	141	199.7	4.0

Piecewise experiments

- Linear algebra
  - Matrix-vector multiplication: Gazell's hybrid method always faster
  - Convolutions: Output rotation not much faster than input rotation

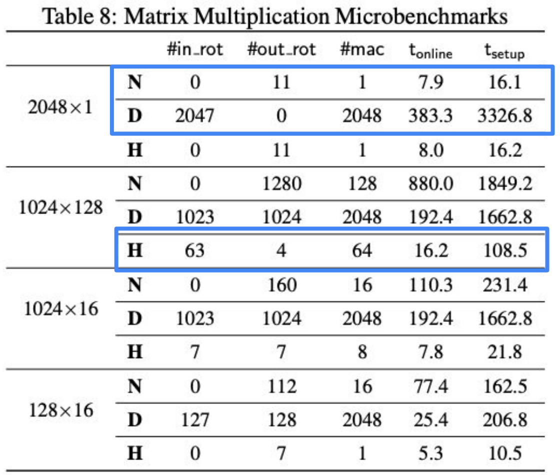


Table 8: Matrix Multiplication Microbenchmarks

		#in_rot	#out_rot	#mac	t_{online}	t_{setup}
2048×1	**N**	0	11	1	7.9	16.1
	D	2047	0	2048	383.3	3326.8
	H	0	11	1	8.0	16.2
1024×128	**N**	0	1280	128	880.0	1849.2
	D	1023	1024	2048	192.4	1662.8
	H	63	4	64	16.2	108.5
1024×16	**N**	0	160	16	110.3	231.4
	D	1023	1024	2048	192.4	1662.8
	H	7	7	8	7.8	21.8
128×16	**N**	0	112	16	77.4	162.5
	D	127	128	2048	25.4	206.8
	H	0	7	1	5.3	10.5

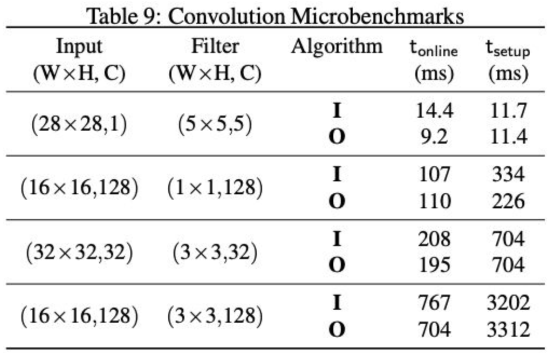


Table 9: Convolution Microbenchmarks

Input (W×H, C)	Filter (W×H, C)	Algorithm	t_{online} (ms)	t_{setup} (ms)
(28×28,1)	(5×5,5)	**I**	14.4	11.7
		O	9.2	11.4
(16×16,128)	(1×1,128)	**I**	107	334
		O	110	226
(32×32,32)	(3×3,32)	**I**	208	704
		O	195	704
(16×16,128)	(3×3,128)	**I**	767	3202
		O	704	3312

Piecewise experiments

- Non-linear layers
  - No benchmark, but it’s tractable
  - Near-linear scaling with outputs
  - Most of the time is offline for ReLU and MaxPool

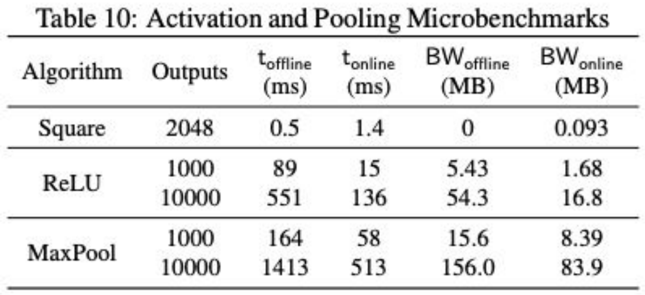


Table 10: Activation and Pooling Microbenchmarks

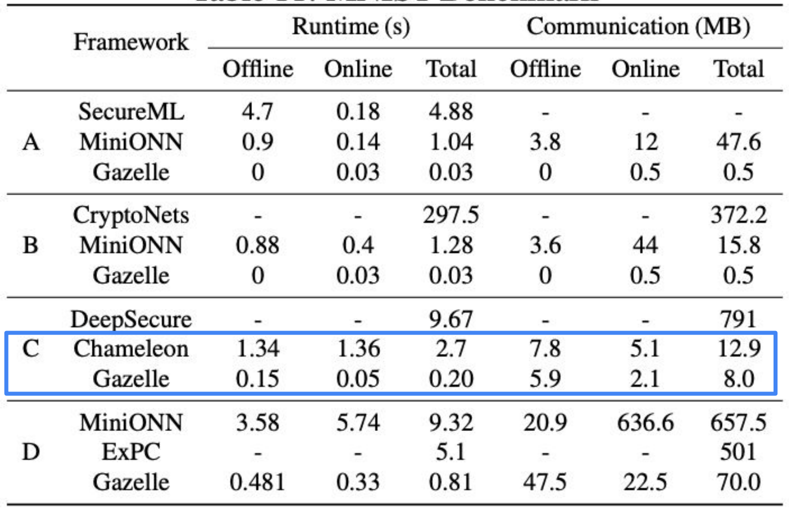
Algorithm	Outputs	$t_{offline}$ (ms)	t_{online} (ms)	$BW_{offline}$ (MB)	BW_{online} (MB)
Square	2048	0.5	1.4	0	0.093
ReLU	1000	89	15	5.43	1.68
	10000	551	136	54.3	16.8
MaxPool	1000	164	58	15.6	8.39
	10000	1413	513	156.0	83.9

Neural net experiments

- MNIST:
  - A: 3-FC with square activation
  - B: 1-Conv and 2-FC with square activation
  - C: 1-Conv and 2-FC with ReLU activation
  - D: 2-Conv and 2-FC with ReLU activation and max pool

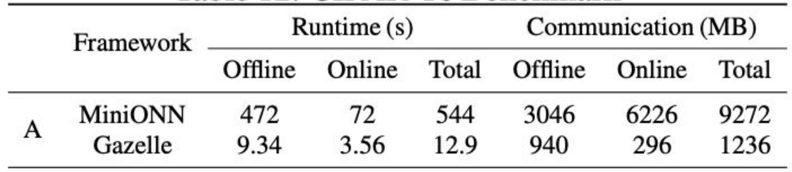
- CIFAR-10
  - A: 2-Conv and 2-FC with ReLU activation and max pool
  - 50x faster
  - 20x lower latency

Table 11: MNIST Benchmark



	Framework	Runtime (s)			Communication (MB)		
		Offline	Online	Total	Offline	Online	Total
	SecureML	4.7	0.18	4.88	-	-	-
A	MiniONN	0.9	0.14	1.04	3.8	12	47.6
	Gazelle	0	0.03	0.03	0	0.5	0.5
	CryptoNets	-	-	297.5	-	-	372.2
B	MiniONN	0.88	0.4	1.28	3.6	44	15.8
	Gazelle	0	0.03	0.03	0	0.5	0.5
	DeepSecure	-	-	9.67	-	-	791
C	Chameleon	1.34	1.36	2.7	7.8	5.1	12.9
	Gazelle	0.15	0.05	0.20	5.9	2.1	8.0
	MiniONN	3.58	5.74	9.32	20.9	636.6	657.5
D	ExPC	-	-	5.1	-	-	501
	Gazelle	0.481	0.33	0.81	47.5	22.5	70.0

Table 12: CIFAR-10 Benchmark



	Framework	Runtime (s)			Communication (MB)		
		Offline	Online	Total	Offline	Online	Total
A	MiniONN	472	72	544	3046	6226	9272
	Gazelle	9.34	3.56	12.9	940	296	1236

Discussion

Takeaways

- Gazelle is able to make secure neural network inference much more efficient
- Homomorphic encryption and clever arrangement can efficiently implement linear algebra ops to replace slower garbled circuits
- Gazelle demonstrates a protocol to alternate between HE and GC

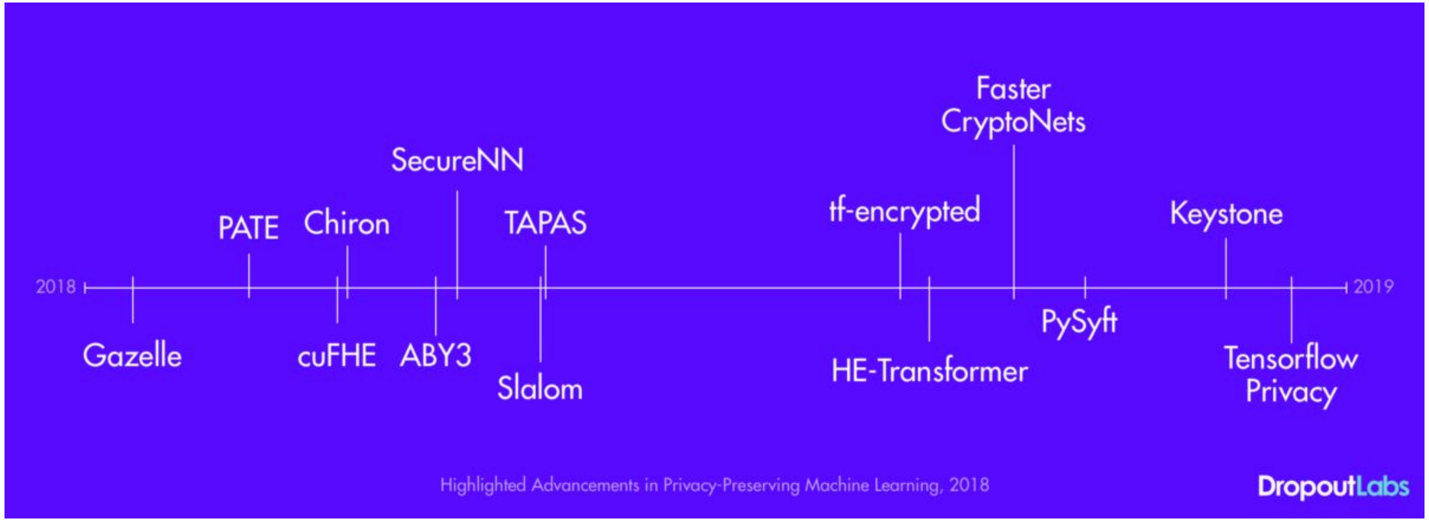
Weaknesses

1. Lacking comparison with non-secure CPU baseline
2. Mentioned model extraction, but does not defend against it
3. What is the performance of the network with square (faster) vs. ReLU activation?
4. Optimizations rely on inputs having smaller dimension than “n” - will it scale?
5. Only tested on very small image datasets
6. Relies on heavily engineered, specialized protocols for each new operation

Bonus: Capitalization of “GAZELLE” in the title implies it is an acronym, when it is not - causing a sponge attack on human brain

Future work

- Handle neural networks with larger input size
- Build framework to compile neural networks into secure inference protocols



Proof of Learning: Definitions and Practice

Jia et al. 2021

https://arxiv.org/abs/2103.05633

Motivation: Threats to ML security:

- Model extraction attack/Model stealing
- Accidental incorrect model updates in distributed training
- Malicious DoS attacks

UNIVERSITY OF TORONTO

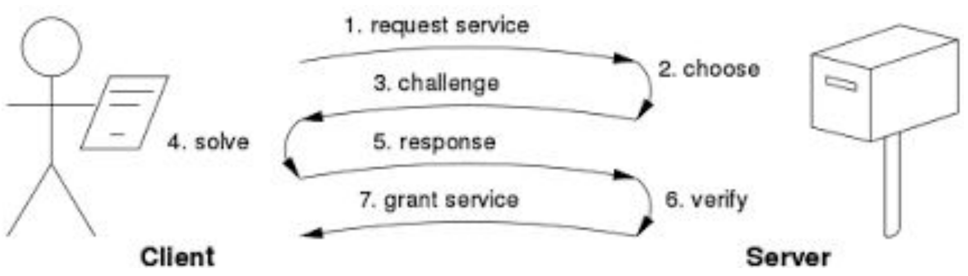
Prior Efforts: proof-of-work (PoW)

PoW: One party proves to another that it has expended computational resources towards a computation result.

- Aims to inhibit **DoS attack**
- **Significant computational resources** are required to request access to the service.
- Used by **bitcoin**!

⇩

Disadvantage: expensive computation



Prior Efforts: Against Model Extraction Attack

Model extraction attack: use substitute dataset to obtain labels from the model and retrain a surrogate model.

Defenses:

- Restricting information released by the model
- Watermarking
  - Inject watermark in the model
  - Triggering dataset

Disadvantages:

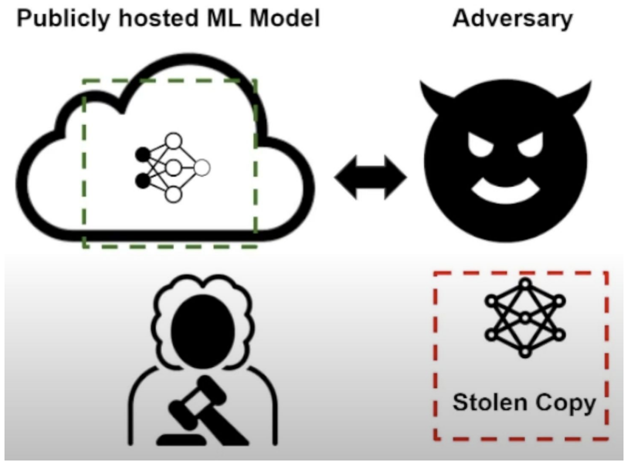
- Require modification of the training process
- Utility degradation

Proof of Learning Strategy (PoL):

Status quo: no mechanism to prove the final parameters of the model is the result of the optimization process after training.

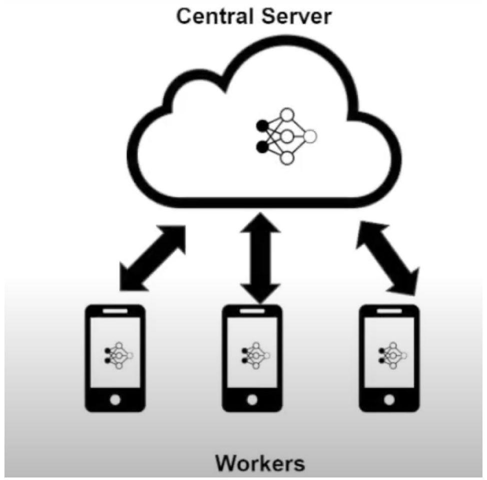
Proof of Learning Strategy (PoL): allow the **prover** to generate a proof for the **verifier** to verify the correctness of the computation performed **during training**.

UNIVERSITY OF TORONTO



Model stealing scenario: When the model is stolen, the owner is able to claim ownership to the model

- Prover: model owner
- Verifier: arbitrator



Distributed training scenario: When distributing the training across multiple workers, defend against incorrect model updates

- Prover: the worker
- Verifier: model owner

UNIVERSITY OF TORONTO

PoW Properties

The desired PoL should have the following **properties**:

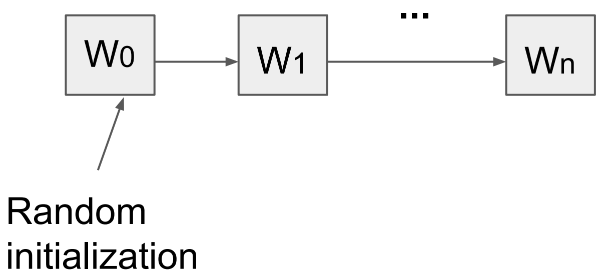
- Correctness
- Security
- Verification efficiency
- Model agnostic
- Limited overhead
- Concise proof

UNIVERSITY OF TORONTO

PoL Formalization

Definition: For a prover ***T***, a valid PoL is defined as ***P(T, fw_T) = (W, I ,H, A)***, where the elements of the tuple are ordered sets indexed by the training step ***t ∈ [T]***

- ***W****:* model weights at steps of training
- ***I***: data instance information
- ***H***: signatures of data instances in ***I***
  - When the data is private
- ***A*****:** auxiliary information
  - e.g.hyperparameters



$$W_i = W_{i-1} - \eta \cdot \nabla_{W_{i-1}}, \hat{\mathcal{L}}_{i-1}$$

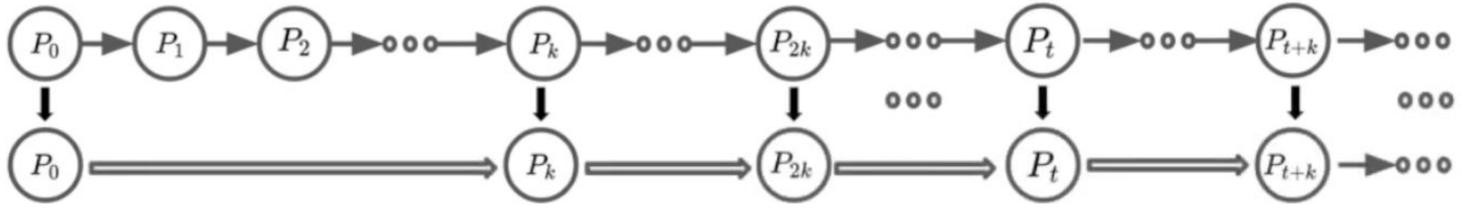
PoL Creation

During the update chain, the proof ***(W, I, H, A)*** is recorded every k steps:

k: periodic interval

Avoids the proof being excessively large

Increasing k linearly decreases storage costs, but also decreases verification accuracy(will be covered later in verification step)



PoL Verification

1. Starts from W_0 - sampled from initialization distribution -> statistical test
   Or previously trained model weights - Valid previous PoL required
2. Store the distance between each consecutive pairs of weights from ***W***
   Distance: distance measure such as p-norm
3. Sort the distances once every epoch, only the largest updates are verified !

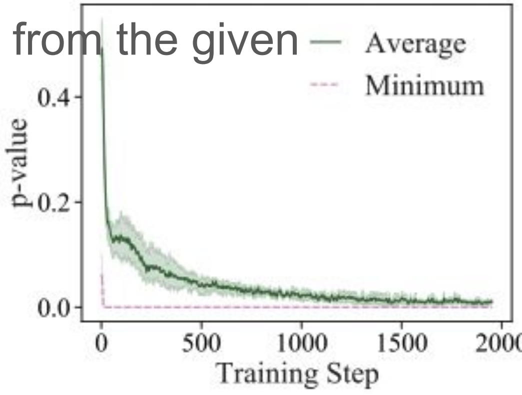
$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}.$$

4. Step-wise verification:
   a. Loads up the corresponding ***W'ₜ*** of the largest updates
   b. Perform a series of k updates to arrive at ***W'ₜ₊ₖ*** and compare it to **Wₜ₊ₖ** in the original PoL.
   c. $d_2(W'_{t+k}, W_{t+k}) \le \delta \longrightarrow$ accept
      i. d: distance measure
      ii. δ hyperparameter - accepting range

UNIVERSITY OF TORONTO

Verifying Initialization: KS test

- Most initialization strategies for model weights samples weights from a designated distribution(e.g. Normal or uniform)
    - Can be easily obtained given initialization strategy(contained in ***A***)
- KS test is a statistical test to check whether the weights are sampled from the designated distribution.
    - P-value: the likelihood that the weights of each layer are from the given distribution.
    - P-value below a chosen significant value -> fail KS-test!
    - -> PoL invalid
    - Another hyperparameter appears!



(a) CIFAR-10

Entropy Growth

Training with stochastic gradient descent is a stationary **Markov process**

I.e.Any randomness in the architecture is fixed

future progressing is independent of the history

But noise may come from hardware and lower level libraries

Training step: $$\tilde{W}_{t+1} = \tilde{W}_t - \eta \nabla_{\tilde{W}_t} \hat{\mathcal{L}}_t + \boxed{z_t}$$

Entropy: $$H(\tilde{W}_1 | \tilde{W}_0) = H(z_0)$$

Theorem 1: the entropy of the training process grows linearly in the number of training Steps. $\Longrightarrow$ **Step-wise verification!**

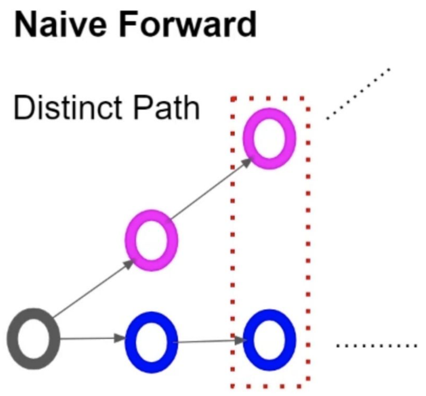
Stepwise verification

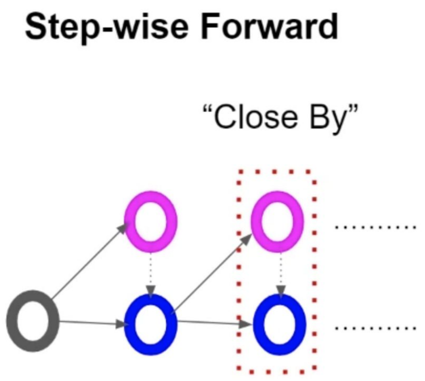
Naive forward: unbound growth of entropy

Stepwise verification:

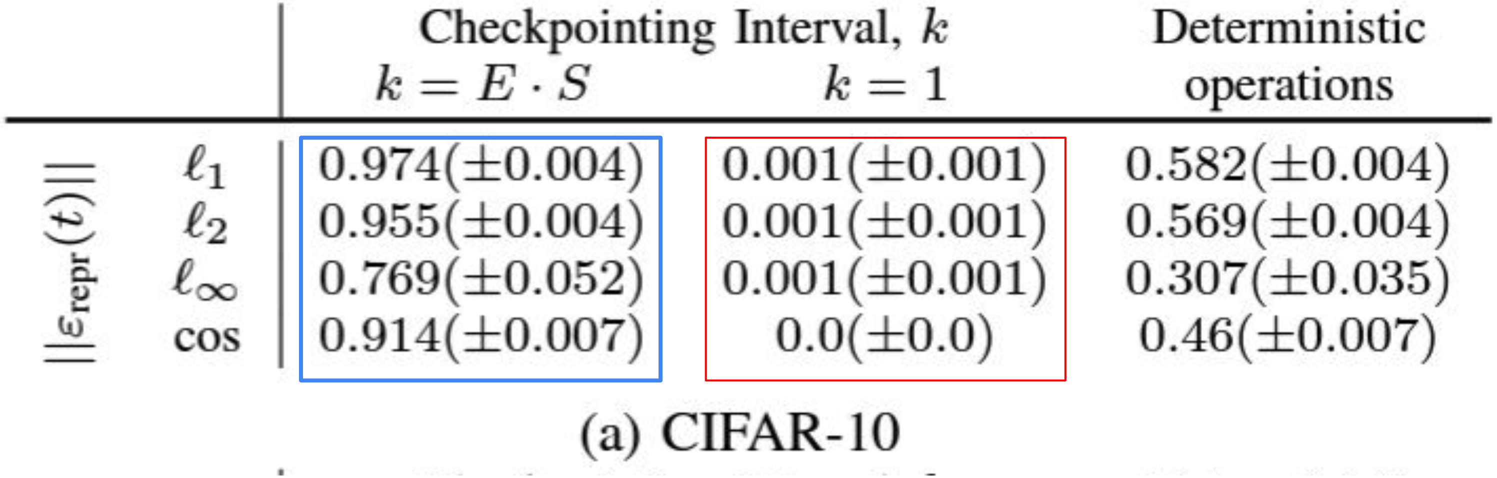
Dealing only with entropy for one-step

1. Compare the reproduced model to the original
2. If pass -> load the model from the proof and continue





Correctness Analysis: Step-wise verification



		Checkpointing Interval, k		Deterministic
		$k = E \cdot S$	$k = 1$	operations
$\|\|\varepsilon_{\text{repr}}(t)\|\|$	ℓ_1	$0.974(\pm 0.004)$	$0.001(\pm 0.001)$	$0.582(\pm 0.004)$
	ℓ_2	$0.955(\pm 0.004)$	$0.001(\pm 0.001)$	$0.569(\pm 0.004)$
	ℓ_∞	$0.769(\pm 0.052)$	$0.001(\pm 0.001)$	$0.307(\pm 0.035)$
	cos	$0.914(\pm 0.007)$	$0.0(\pm 0.0)$	$0.46(\pm 0.007)$

(a) CIFAR-10

Proves that it is impossible for a adversary to recreate a PoL!

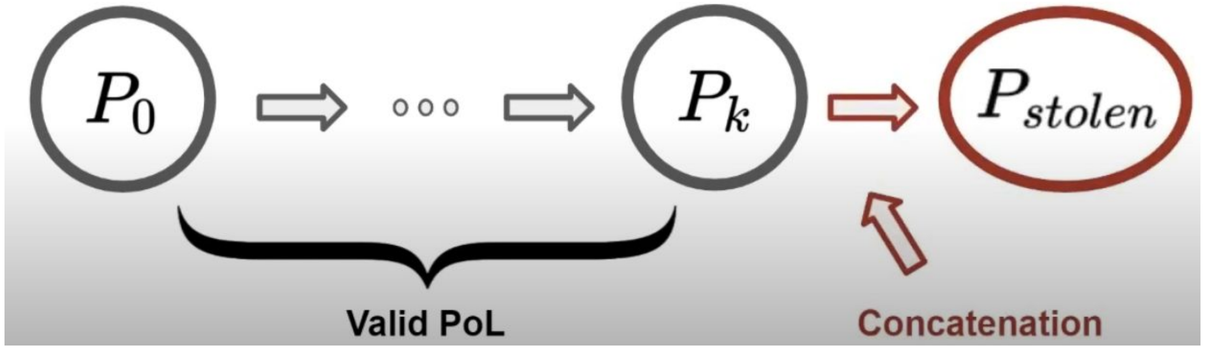
Security Analysis: Direct Retraining

Spoofing: An adversary tries to create a PoL with **less** computational cost to **pass** verification.

Possible because: only part of the PoL is verified.

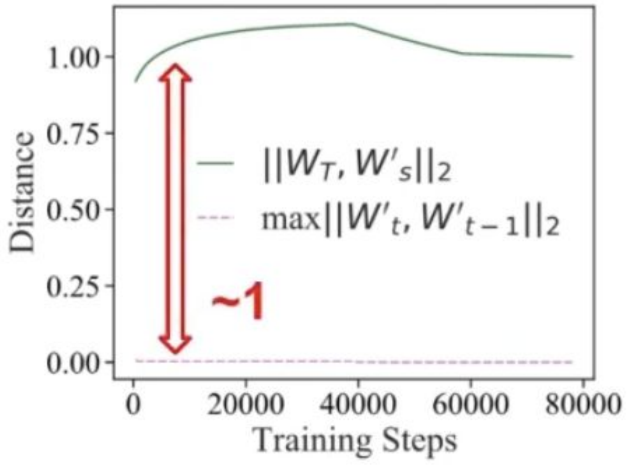
Approach: Partially valid PoL

Train fewer steps and concatenate with the stolen model

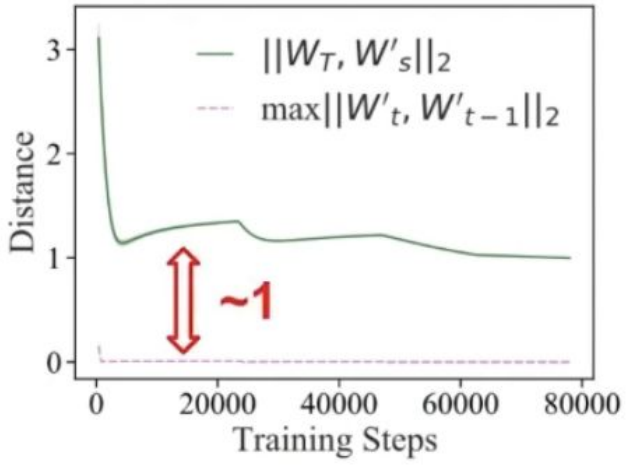


Security Analysis: Direct Retraining

Magnitude of discontinuity



(a) CIFAR-10



(b) CIFAR-100

Top n largest steps are verified, but the concatenated steps are likely to be the largest!

UNIVERSITY OF TORONTO

NOT going to work!

Security Analysis: Inverse Gradient Method

$$\begin{tikzcd} W_t \arrow[r, Rightarrow] & W_{t-1} \arrow[r, Rightarrow] & \dots \arrow[r, Rightarrow] & W_0 \end{tikzcd}$$

Given

Initial weights

$$\beta(W_{t-1}) := W_{t-1} - W_t - \eta \nabla_{W_{t-1}} \mathcal{L} = 0$$

W_0 needs to be justified to be sampled from a random distribution or accompanied with a valid PoL P_0

Security Analysis: Inverse Gradient Method

The inverse gradient method won’t work!

Entropy for forward training

$$H'(\Theta_T) = \lim_{T\to\infty} H(\tilde{W}_T|\tilde{W}_0, ..., \tilde{W}_{T-1}) = H(\tilde{W}_1|\tilde{W}_0) \\ = H(z_0)$$

Entropy for Inverse gradient method

$$\beta(W_{t-1}) := W_{t-1} - W_t - \eta\nabla_{W_{t-1}}\mathcal{L} = 0$$

$$\Downarrow$$

$$H(\tilde{W}_{t-1}|\tilde{W}_t) = H(z_0) + H(\eta\nabla_{\tilde{W}_{t-1}}L|\tilde{W}_t)$$

$\Rightarrow$ **Large verification error!**

		CIFAR-10	CIFAR-100
$\|\|\varepsilon_{\text{repr}}\|\|$	ℓ_1	0.023 ± 0.001	0.005 ± 0.001
	ℓ_2	0.048 ± 0.004	0.016 ± 0.005
	ℓ_∞	0.18 ± 0.044	0.073 ± 0.014
	cos	0.016 ± 0.002	0.0 ± 0.0

UNIVERSITY OF TORONTO

Summary

Advantages:

1. No changes to the already complex training process and model architecture
2. No trade-off between model utility and model security
3. A permanent record of the entire training process(not just the end result!)

Limitations:

1. The trade-off between verification error and storage/computational cost
2. Introduced a few hyperparameters, e.g. checkpointing interval k, p-value in KS test without providing a guidance for setting the hyperparameters.

Future work:

1. Reducing verification cost
   a. Controlling noise
   b. Exploring sources of randomness

UNIVERSITY OF TORONTO