HETAL: Efficient Privacy-preserving Transfer Learning with Homomorphic Encryption

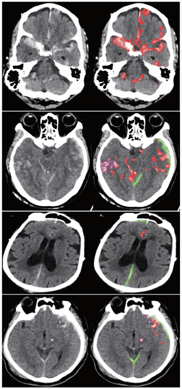
Seewoo Lee, Garam Lee, Jung Woo Kim, Junbum Shin, Mun-Kyu Lee

June 13, 2023

Center for Artificial Intelligence and Natural Sciences, KIAS

Introduction

The era of Artificial Intelligence



But...

Netflix Cancels Contest After Concerns Are Raised About Privacy

Facebook to delete users' facial-recognition data after privacy complaints

DNA facial prediction could make protecting your privacy more difficult

How to solve privacy issues in AI?

Privacy-preserving Machine Learning (PPML): prevent privacy leakage in machine learning. For example,

- Federated Learning
- Differential Privacy
- Encrypted computation
    - Secure Multi-party Computation
    - Homomorphic Encryption

Homomorphic Encryption

$$\text{Dec}(\text{Enc}(x) + \text{Enc}(y)) = x + y$$
$$\text{Dec}(\text{Enc}(x) \times \text{Enc}(y)) = x \times y$$

Homomorphic Encryption

$$\mathrm{Dec}(\mathrm{Enc}(x) + \mathrm{Enc}(y)) = x + y$$
$$\mathrm{Dec}(\mathrm{Enc}(x) \times \mathrm{Enc}(y)) = x \times y$$

We can perform computation on encrypted data *without having to decrypt it*.

Privacy-preserving machine learning and HE

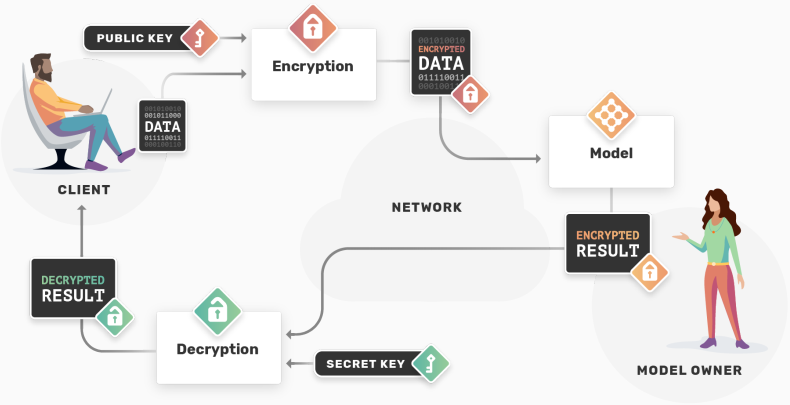


Image from Openmined blog, encrypted inference with HE

Previous works

- Several ML algorithms (logistic regression, SVM, decision tree, neural network) are implemented using HE.

Previous works

- Several ML algorithms (logistic regression, SVM, decision tree, neural network) are implemented using HE.
- Biggest issue of HE:

Previous works

- Several ML algorithms (logistic regression, SVM, decision tree, neural network) are implemented using HE.
- Biggest issue of HE: **Too Slow**

Our contributions

- We propose **HETAL**, an efficient **H**omomorphic **E**ncryption based **T**r**a**nsfer **L**earning algorithm for privacy-preserving TL. **HETAL** is the first practical scheme that strictly provides HE-based encrypted training.

Our contributions

- We propose **HETAL**, an efficient **H**omomorphic **E**ncryption based **T**r**a**nsfer **L**earning algorithm for privacy-preserving TL. **HETAL** is the first practical scheme that strictly provides HE-based encrypted training.
- We implemented and evaluated HETAL using five well-known benchmark datasets (MNIST, CIFAR-10, Face Mask Detection, DermaMNIST and SNIPS), using two pre-trained models (`ViT` and `MPNet`). HETAL took **less than an hour** for training on all datasets, with **at most 0.5% accuracy drop**.

Our contributions

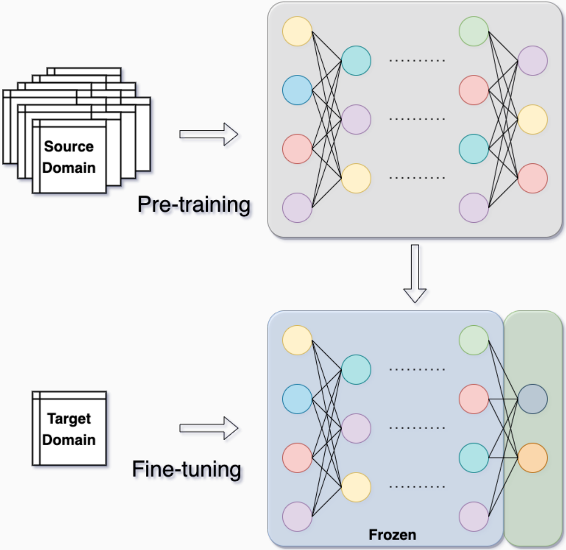
- We propose a new softmax approximation algorithm, which covers a significantly wider range than the previous works **(from** $[-8, 8]$ **to** $[-128, 128]$**)** with high precision.

Our contributions

- We propose a new softmax approximation algorithm, which covers a significantly wider range than the previous works **(from** $[-8, 8]$ **to** $[-128, 128]$**)** with high precision.
- We also propose optimized matrix multiplication algorithms, DiagABT **and** DiagATB, that compute matrix multiplications of the form $AB^\intercal$ and $A^\intercal B$ for encrypted matrices A and B. Our proposed algorithms are more efficient in both memory and computation than previous algorithms [1, 2] and show a performance improvement of **1.8 to 323 times**.

Preliminaries

Transfer learning



Transfer learning

- *Pre-train* (large) models on a large dataset (e.g. ImageNet, Wikipedia, ...) and *fine-tune* on target datasets
- Computer Vision: ResNet, ViT, ...
- NLP: GPT-(1,2,3,3.5,4), BERT, ...
- Audio & Speech: Wave2Vec, HuBERT, ...

Approaches for PPML

Approach	Pros	Cons
FL	• Enables training on decentralized data • Allows for continuous learning on live data	• Privacy can be leaked from model updates • May introduce bias due to non-i.i.d. data distribution
DP	• Provides strong mathematical guarantees on individual privacy	• May introduce significant noise, impacting model accuracy • Requires careful parameter tuning
SMPC	• Computation over encrypted data • Less computational overhead than HE	• High communication cost
HE	• Quantum-safe • No communication is needed	• High computational cost

Homomorphic Encryption (HE)

- Form of encryption that one can perform computation over ciphertexts.
- Based on the hardness of (R)LWE problems, which is *widely believed* to be unbreakable even with quantum computers.
    - Solve system of linear equations modulo large numbers, *but with errors*. $Ax = b + e$.
    - Solution for (R)LWE problems $\Rightarrow$ quantum algorithm for lattice problesm like GapSVP or SIVP (Regev [3], Lyubashevsky-Peikert-Regev [4])

Homomorphic Encryption (HE)

- Partial Homomorphic Encryption : only certain operation (addition or multiplication, but not both) is available.
- Leveled Homomorphic Encryption: evaluation of arbitrary circuits *of bounded depth*
- Fully Homomorphic Encryption (FHE): allows evaluation of *arbitrary circuits of unbounded depth*.
- Many HE schemes exist: BGV/BFV, FHEW/TFHE, **CKKS**

Cheon-Kim-Kim-Song (CKKS) scheme

(Leveled) HE scheme that supports approximate computation over complex numbers. It can be a FHE scheme with (approximate) bootstrapping [5, 6].

Plaintexts are vectors of complex numbers[1], and following operations are available in CKKS scheme:

- Addition
- Hadamard multiplication: element-wise multiplication
- Rotation / Complex conjugation
- Bootstrapping: refresh noise of ciphertexts and enable *fully* homomorphic encryption. Most costly operation among all.

[1] To be precise, *messages* are.

HETAL

Threat model

- We assume an AutoML-like service, in which a client (data owner) can outsource model training to a server.

Threat model

- We assume an AutoML-like service, in which a client (data owner) can outsource model training to a server.
- We assume that the server and client can share a pre-trained generic model as a feature extractor.

Threat model

- We assume an AutoML-like service, in which a client (data owner) can outsource model training to a server.
- We assume that the server and client can share a pre-trained generic model as a feature extractor. During the training task, the client extracts features from its private data using the feature extractor and sends the HE-encrypted features to the server.

Threat model

- We assume an AutoML-like service, in which a client (data owner) can outsource model training to a server.
- We assume that the server and client can share a pre-trained generic model as a feature extractor. During the training task, the client extracts features from its private data using the feature extractor and sends the HE-encrypted features to the server. The server performs fine-tuning for TL on the ciphertext domain and produces an encrypted model.

Threat model

- We assume that the server is honest-but-curious (HBC), where an HBC adversary is a legitimate participant in a communication protocol who will not deviate from the defined protocol but will attempt to learn all possible information from legitimately received messages.

Threat model

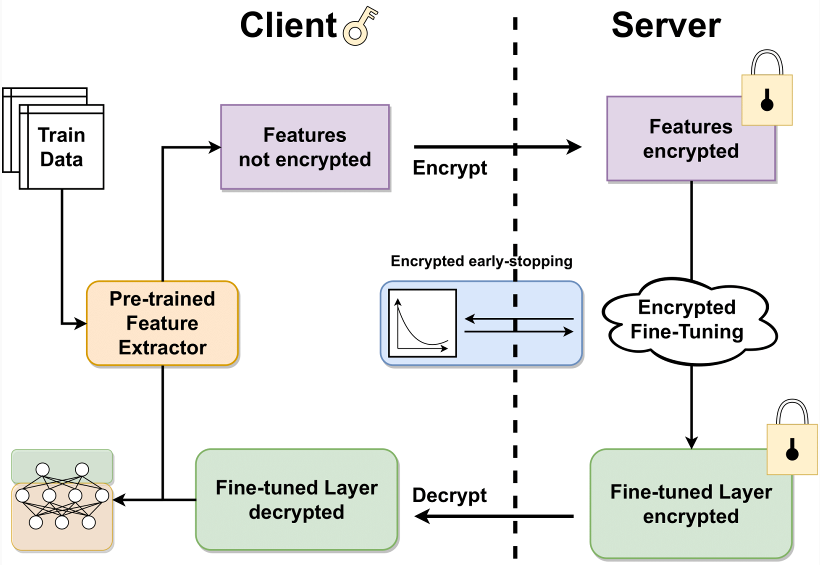
- We assume that the server is honest-but-curious (HBC), where an HBC adversary is a legitimate participant in a communication protocol who will not deviate from the defined protocol but will attempt to learn all possible information from legitimately received messages.
- Note that HE provides a good defense to protect the data against an HBC server because the server performs computation over encrypted data without knowing the decryption key.

Reconstruct data from features

Extracted features may contain significant information about the original raw data, and we can even reconstruct it from features. For example,

- Reconstruct facial image from CNN-based features [7, 8]
- Use GAN to reconstruct facial image [9]
- Reconstruct sentences from SentenceBERT embeddings [10]

HETAL


Client
Server
Train
Data
Features
not encrypted
Encrypt
Features
encrypted
Encrypted early-stopping
Pre-trained
Feature
Extractor
Encrypted
Fine-Tuning
Fine-tuned Layer
decrypted
Decrypt
Fine-tuned Layer
encrypted

HETAL

- Feature extraction is done on the client wide with pre-trained model, which is assumed to be publicly available.
- Fine-tuning is done on the server side with encrypted features.
- Encrypted early-stopping: since it is hard to know when to stop the training, we add simple client-server communication protocol to determine whether to stop the training or not.

Encrypted early-stopping

1. Client send encrypted features of validation data.
2. Server computes encrypted logit and send it back to the client.
3. Client decrypts the logit and compute loss using it.
4. Compare with previous validation losses and determine whether to stop or not, and send a signal to the server.

Encrypted early-stopping

1. Client send encrypted features of validation data.
2. Server computes encrypted logit and send it back to the client.
3. Client decrypts the logit and compute loss using it.
4. Compare with previous validation losses and determine whether to stop or not, and send a signal to the server.

- The signal does not seem to be useful for the server to recover the client's private data

Encrypted fine-tuning

Main components of the encrypted fine-tuning are

- Softmax approximation
- Matrix multiplication

Softmax approximation

We need to compute softmax for fine-tuning a classification layer for multi-class classification. Since it is not a polynomial, we need a polynomial approximation of it.

Softmax approximation

We need to compute softmax for fine-tuning a classification layer for multi-class classification. Since it is not a polynomial, we need a polynomial approximation of it.

First try: approximate exponential and inverse function separately, and combine them.

Softmax approximation - exponential

We first approximate exponential function by

$$\exp(x) \approx (g(x/2^d))^{2^d} =: \mathbf{AExp}_{g,d}(x)$$

where $g(x)$ is a minimax approximation of $\exp(x)$ on a given interval, and d is a *domain extension order*.

Softmax approximation - inverse

After that, we approximate inverse function via Goldschmidt algorithm. For $0 < x < 2$, we have

$$\frac{1}{x} = \frac{1}{1-(1-x)} = \prod_{m=0}^{\infty} (1 + (1-x)^{2^m})$$

so we can approximate $1/x$ as

$$\frac{1}{x} = \frac{1}{R}\frac{1}{x/R} \approx \frac{1}{R}\prod_{m=0}^{n}\left(1 + \left(1 - \frac{x}{R}\right)^{2^m}\right) =: \mathrm{AInv}_{R,n}(x)$$

for suitable choice of R and N, where $0 < x < 2R$. This can be computed recursively.

Softmax approximation

Combining these, we get an approximation

$$\mathrm{Softmax}(x_1, \ldots, x_c) \approx \mathbf{AInv}_{R,n}(Z) \cdot (\mathbf{AExp}_{g,d}(x_1), \cdots, \mathbf{AExp}_{g,d}(x_c))$$

where $Z = \sum_{j=1}^{c} \mathbf{AExp}_{g,d}(x_j)$. We choose

- $g(x)$ be a degree 8 approximation on $[-1, 1]$
- $d = 3$
- $R = 100$
- $n = 20$

and it gives an approximation with max error 1.4×10^{-6} on $[-8, 8]$ ($c = 10$).

We are done.

Softmax approximation

Combining these, we get an approximation

$$\mathrm{Softmax}(x_1, \ldots, x_c) \approx \mathbf{AInv}_{R,n}(Z) \cdot (\mathbf{AExp}_{g,d}(x_1), \cdots, \mathbf{AExp}_{g,d}(x_c))$$

where $Z = \sum_{j=1}^{c} \mathbf{AExp}_{g,d}(x_j)$. We choose

- $g(x)$ be a degree 8 approximation on $[-1, 1]$
- $d = 3$
- $R = 100$
- $n = 20$

and it gives an approximation with max error 1.4×10^{-6} on $[-8, 8]$ ($c = 10$).

We are done. **NO!**

Issues

The approximation range $[-8, 8]$ is not enough for training.

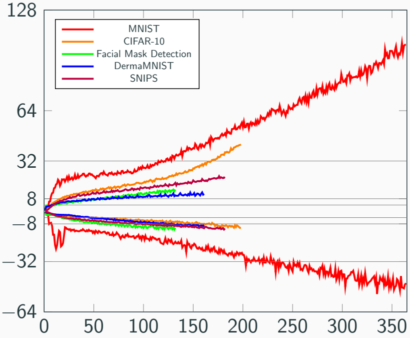


Figure 1: Maximum and minimum value of input of softmax at each step (minibatch) for each dataset.

Issues

The approximation range $[-8, 8]$ is not enough for training.

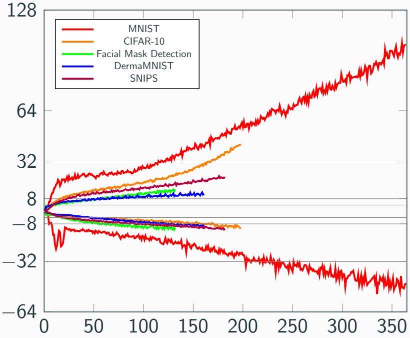


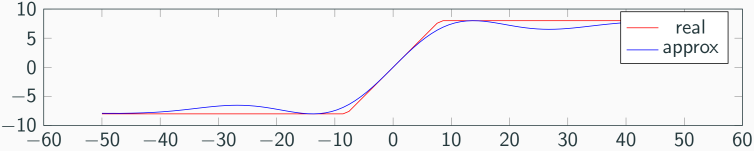
Figure 1: Maximum and minimum value of input of softmax at each step (minibatch) for each dataset.

We need to make the domain of approximation larger.

Solution: domain extension

Cheon et al. [11] proposed *domain extension*, which can enlarge the domain of approximation exponentially for sigmoid-like functions.

Idea: Approximate *clipping function* (as an iterated composition of low-degree polynomials).



Solution: domain extension

The original work is for univariate functions, and we can apply it element-wise to extend the domain of approximation.

$$\mathrm{ASoftmax}(\mathrm{DEF}(x_1), \ldots, \mathrm{DEF}(x_c))$$

We are done.

Solution: domain extension

The original work is for univariate functions, and we can apply it element-wise to extend the domain of approximation.

$$\text{ASoftmax}(\text{DEF}(x_1), \ldots, \text{DEF}(x_c))$$

We are done. **NO!!**

Solution: domain extension

This still gives large error. Here's an example:

Assume that we have an approximation of a 3-variable softmax on a 3-dimensional box $[-8, 8]^3$, and an input is given by $(8, 10, 13)$.

- True output $=$ Softmax$(8, 10, 13) = (0.006, 0.047, 0.946)$.
- This approach $\approx$ Softmax$(\mathrm{DEF}_8(8, 10, 13)) =$ Softmax$(8, 8, 8) = (0.333, 0.333, 0.333)$

Solution: domain extension + normalization

Before we apply domain extension, we first apply *normalization*: subtract maximum value of input from all. The output is the same:

$$\mathrm{Softmax}(x_1, \ldots, x_c) = \mathrm{Softmax}(x_1 - m, \ldots, x_c - m)$$

Max function is not a polynomial function. But we can still approximate it with $\log c$ homomorphic comparisons [12].

We are done.

Solution: domain extension + normalization

Before we apply domain extension, we first apply *normalization*: subtract maximum value of input from all. The output is the same:

$$\mathrm{Softmax}(x_1, \ldots, x_c) = \mathrm{Softmax}(x_1 - m, \ldots, x_c - m)$$

Max function is not a polynomial function. But we can still approximate it with $\log c$ homomorphic comparisons [12].

We are done. **Yes, in the following sense.**

Error bound

Theorem

Let $p : \mathbb{R}^c \to \mathbb{R}^c$ be an approximation of the softmax on $[-R, R]^c$ satisfying

$$\|\mathrm{Softmax}(\mathbf{x}) - p(\mathbf{x})\|_\infty < \epsilon.$$

Then for $\mathbf{x} \in [-\frac{1}{2}L^n R, \frac{1}{2}L^n R]^c$, we have

$$\|\mathrm{Softmax}(\mathbf{x}) - p(D_n(\mathrm{Norm}(\mathbf{x})))\|_\infty < \beta + \epsilon,$$

where $\beta = \beta(\delta, c, r, L, d)$ is a constant that depends only on δ, c, r, L, d.

- D_n is the domain extension polynomial that approximates clipping function $\mathrm{DEF}_R(x)$ on $[-L^n R, L^n R]$.

Matrix multiplication

While training, we encounter the following matrix multiplications:

- Inference: $\mathbf{P} = \mathrm{Softmax}(\mathbf{X}\mathbf{W}^\intercal)$,
- Backpropagation: $\nabla_W \mathcal{L} = \frac{1}{n}(\mathbf{P} - \mathbf{Y})^\intercal \mathbf{X}$.

Hence it is important to compute these efficiently.

Matrix multiplication

There are two possible approaches:

1. Implement AB and transpose
2. Implement $AB^\intercal$ and $A^\intercal B$

We choose **2** because

- Existing transpose algorithm (Jiang et al. [13]) is costly and not applicable for large matrices
- $AB^\intercal$ and $A^\intercal B$ is more HE-friendly than AB

DiagABT

Each matrix is divided into several submatrices of fixed size and those are encoded in a row-wise manner, where we use zero paddings if needed. For $A, B \in \mathbb{R}^{d \times d}$, we can compute AB^T by

$$AB^\mathsf{T} = \sum_{0 \le k < d} \mathsf{SumCols}(A \odot \mathsf{RotUp}(B, k)) \odot M^{(k)}$$

where

- $\odot$: element-wise multiplication
- $\mathsf{RotUp}(B, k)$: rotate B in upper direction by k.
- $\mathsf{SumCols}(A)$: summation of columns of A, where the results are copied along all entries
- $M^{(k)}$: mask that extracts k-th diagonal.

DiagABT

Rotation is the main bottleneck of this algorithm, and we apply several optimization techniques to reduce the number of rotations, such as

DiagABT

Rotation is the main bottleneck of this algorithm, and we apply several optimization techniques to reduce the number of rotations, such as

- Tiling for packing rectangular matrices
  - Particularly effective in our case since number of classes is (usually) much smaller than minibatch size or number of features.

DiagABT

Rotation is the main bottleneck of this algorithm, and we apply several optimization techniques to reduce the number of rotations, such as

- Tiling for packing rectangular matrices
  - Particularly effective in our case since number of classes is (usually) much smaller than minibatch size or number of features.
- Complexification
  - Based on $\Re((a+ci)(b-di)) = ab + cd$ (Hong et al. [14]).
  - Further reduces computational costs by half.

DiagABT

This gives: for $A \in \mathbb{R}^{a\times b}$ and $B \in \mathbb{R}^{c\times b}$, we have $A\overline{B}^{\intercal} = X + \mathrm{Conj}(X)$, where

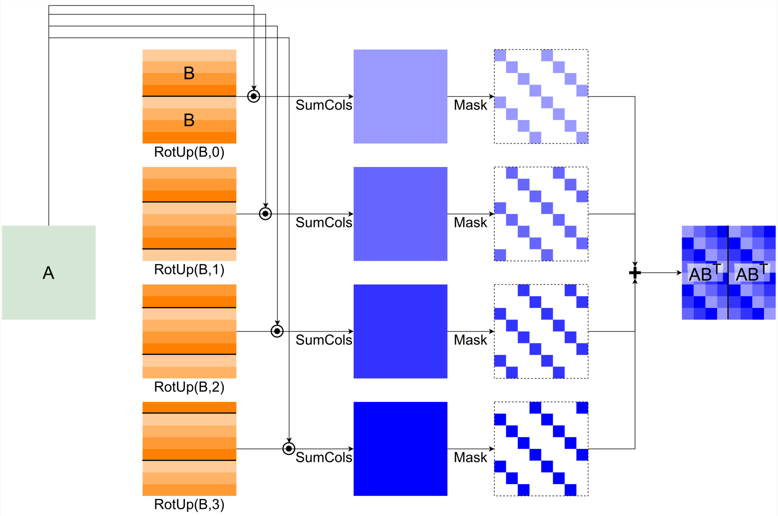
$$X = \sum_{0\le k<c/2} \mathrm{SumCols}(A \odot \mathrm{RotUp}(\overline{B}_{\mathrm{cplx}}, k)) \odot M^{(k,c)}_{\mathrm{cplx}}.$$

- $\overline{B}$ is B tiled in vertical direction
- $\overline{B}_{\mathrm{cplx}} = \overline{B} + \sqrt{-1}\,\mathrm{RotUp}(\overline{B}, c/2)$
- $M^{(k,c)}$ is an off-diagonal masking matrix with entries

$$M^{(k,c)}_{i,j} = \begin{cases} 1 & j \equiv i + k \pmod{c} \\ 0 & \text{otherwise} \end{cases}$$

- $M^{(k,c)}_{\mathrm{cplx}} = \frac{1}{2}M^{(k,c)} - \frac{\sqrt{-1}}{2}M^{(k+c/2,c)}$

DiagABT



Complexification is not included for simplicity

DiagATB

As DiagABT, we can compute $A^\intercal B$ as $\underline{A}^\intercal B = X + \mathrm{Conj}(X)$ where

$$X = \sum_{0 \le k < c/2} \mathrm{SumRows}(\mathrm{RotLeft}(\underline{A}_{\mathrm{cplx}}, k) \odot B) \odot M_{\mathrm{cplx}}^{(-k,a)}$$

where $\underline{A}$ is a tiling of A in the horizontal direction and

$$\underline{A}_{\mathrm{cplx}} = \underline{A} + \sqrt{-1}\mathrm{RotLeft}(\underline{A}, c/2).$$

This approach (DiagATB) is almost same as DiagABT, but there one notable difference - there are two **RotLeft**s, where each consumes a multiplicative depth.[2]

[2]Note that RotUp does not consume any depth.

DiagATB

To address this issue, we use the **PRotUp** that eventually consumes B's level instead of A's, so we can compute $A^\intercal B$ without any additional depth consumptions.[3]

$$X = \sum_{0 \le k < c/2} \mathrm{SumRows}(\mathrm{Lrot}(\underline{A}_{\mathrm{cplx}}, k) \odot \mathrm{PRotUp}(B, k)) \odot M_{\mathrm{cplx}}^{(-k,a)}$$

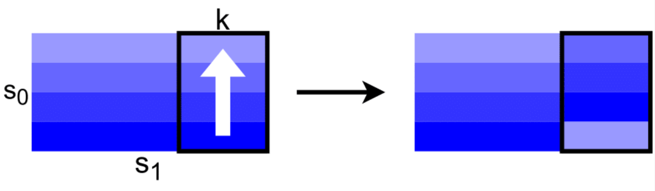


Figure 2: $\mathrm{PRotUp}(B, k)$

[3] except when $\mathrm{level}(A) = \mathrm{level}(B)$, which does not happen in our case.

Results

Experimental setup

- HEaaN library (CryptoLab)
  - CKKS scheme
  - Supports bootstrapping & GPU implementation
  - We take $N = 2^{16}$ as a cyclotomic ring dimension (single ciphertext encrypt $N/2 = 2^{15}$ complex numbers) and ciphertext modulus $q \approx 2^{1555}$ which gives 128-bit security level.
- Intel Xeon Gold 6242 CPU at 2.80GHz processor
- Single NVIDIA Ampere A40 GPU

operation	Add	Rotate	CMult	Mult	Bootstrap
time	0.0085ms	1.2ms	0.9ms	1.6ms	159ms

Experimental setup

We use the following datasets for experiments.

- **MNIST**
- **CIFAR-10**
- **Face Mask Detection** (Larxel [15])
- **DermaMNIST** (Yang et al. [16])
- **SNIPS** (Coucke et al. [17])

We use `ViT-Base` for image datasets and `MPNet-Base` for SNIPS dataset as feature extractors. Both models embed a data point into a single 768-dimensional vector.

Results

dataset	encrypted			not encrypted	
	Running time		ACC (a)	ACC (b)	ACC loss ((b) - (a))
	Total (s)	Time / Iter (s)			
MNIST	3442.29	9.46	96.73%	97.24%	0.51%
CIFAR-10	3114.30	15.72	96.57%	96.62%	0.05%
Face Mask Detection	566.72	4.29	95.46%	95.46%	0.00%
DermaMNIST	1136.99	7.06	76.06%	76.01%	-0.05%
SNIPS	1264.27	6.95	95.00%	94.43%	-0.57%

Softmax approximation

We sample 400M points uniformly on $[-128, 128]^c$, and compute maximum errors of our approximation.[4] The maximum error was 0.0037–0.0224 and the average error was 0.0022–0.0046 depending on the input dimension.

[4] with input dimensions $c \in \{3, 5, 7, 10\}$

Softmax approximation

We compare our softmax approximation algorithm with

- (Lee et al. [18]) Gumbel softmax trick: simply divides inputs by certain large numbers.
- (Hong et al. [14]) Approximate exponential as $\exp(x) \approx (1 + x/2^n)^{2^n}$.
- (Jin et al. [1]) Use one-vs-each softmax [19] as an alternative

First two works use softmax for inference, and the only last one is used for training.

Softmax approximation

The errors of previous works are fairly large and can't cover large domain:

- Lee et al.: 0.89 - 0.99, covers up to $[-32, 32]$
- Hong et al.: 0.07 - 0.41, covers up to $[-8, 8]$
- Jin et al.: 0.18 - 0.80, covers up to $[-4, 4]$

Matrix multiplication

We compare our DiagABT and DiagATB algorithm with

- (Jin et al. [1]) Use row-majored packing (RP), column-majored packing (CP), and replicated packing (REP).
- (Crockett [2]) ColMajor and RowMajor extract and replicate rows/columns and view them as matrix-vector/vector-matrix multiplications.

Matrix multiplication

(a, b, c)	$AB^\intercal$ $(A \in \mathbb{R}^{a\times b}, B \in \mathbb{R}^{c\times b})$					$A^\intercal B$ $(A \in \mathbb{R}^{a\times c}, B \in \mathbb{R}^{a\times b})$				
	[1]*	ColMajor†	DiagABT	Speedup		[1]*	RowMajor†	DiagATB	Speedup	
(128, 128, 4)	0.8192	0.1104	**0.0601**	13.63	1.84	10.0352	0.1171	**0.0415**	241.81	2.82
(256, 256, 8)	3.2768	0.3203	**0.1211**	27.06	2.64	40.1408	0.3167	**0.1239**	323.98	2.56
(512, 769, 4)	4.9216	0.7609	**0.1223**	40.24	6.22	60.2896	0.7176	**0.3343**	180.35	2.15
(1024, 769, 8)	9.8432	3.0428	**0.3710**	26.53	8.20	120.5792	2.8546	**1.2558**	96.02	2.27
(2048, 769, 16)	19.6864	12.6251	**1.2376**	15.91	10.20	241.1584	11.8220	**4.9970**	48.26	2.37

- For [1], we report estimated running times due to memory issues.
- We get performance improvementes of 1.8 to 323 times.

Matrix multiplication

(a, b, c)	$AB^\intercal$ ($A \in \mathbb{R}^{a\times b}, B \in \mathbb{R}^{c\times b}$)			$A^\intercal B$ ($A \in \mathbb{R}^{a\times c}, B \in \mathbb{R}^{a\times b}$)		
	[1]*	ColMajor	DiagABT	[1]*	RowMajor	DiagATB
$(128, 128, 4)$	0	4	0	0	4	2
	512	4	2	512	4	0
	0	63	34	7680	63	18
$(256, 256, 8)$	0	16	0	0	8	4
	2048	16	8	2048	16	0
	0	191	64	30720	191	72
$(512, 769, 4)$	0	52	0	0	4	2
	3076	52	26	3076	52	0
	0	495	50	46140	495	238
$(1024, 769, 8)$	0	200	0	0	8	4
	6152	200	100	6152	200	0
	0	2047	140	92280	2047	1008
$(2048, 769, 16)$	0	784	0	0	16	8
	12304	784	392	12304	784	0
	0	8703	456	184560	8703	4328

- Each row represents the number of CMult, Mult, and Rot.

Future works

Future works

- Apply to other domains, e.g. speech recognition or finance.

Future works

- Apply to other domains, e.g. speech recognition or finance.
- Use softmax approximation for other purposes, such as encrypted inference of transformers. Same for matrix multiplication.

Future works

- Apply to other domains, e.g. speech recognition or finance.
- Use softmax approximation for other purposes, such as encrypted inference of transformers. Same for matrix multiplication.
- Low-level optimization, especially for matrix multiplications.

Future works

- Apply to other domains, e.g. speech recognition or finance.
- Use softmax approximation for other purposes, such as encrypted inference of transformers. Same for matrix multiplication.
- Low-level optimization, especially for matrix multiplications.
- Can we fine-tune models on encrypted data for purposes other than classification? Encrypted fine-tuning of LLMs?

You can find packages and benchmarking codes in
`https://github.com/CryptoLabInc/HETAL`

You can find packages and benchmarking codes in

`https://github.com/CryptoLabInc/HETAL`

Thank you!

References i

[1] C. Jin, M. Ragab, and K. M. M. Aung, "Secure transfer learning for machine fault diagnosis under different operating conditions," in *International Conference on Provable Security*, pp. 278–297, Springer, 2020.

[2] E. Crockett, "A low-depth homomorphic circuit for logistic regression model training," *Cryptology ePrint Archive*, 2020.

[3] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM (JACM)*, vol. 56, no. 6, pp. 1–40, 2009.

[4] V. Lyubashevsky, C. Peikert, and O. Regev, “On ideal lattices and learning with errors over rings,” *Journal of the ACM (JACM)*, vol. 60, no. 6, pp. 1–35, 2013.

[5] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, “Bootstrapping for approximate homomorphic encryption,” in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 360–384, Springer, 2018.

[6] H. Chen, I. Chillotti, and Y. Song, “Improved bootstrapping for approximate homomorphic encryption,” in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 34–54, Springer, 2019.

[7] G. Mai, K. Cao, P. C. Yuen, and A. K. Jain, “On the reconstruction of face images from deep face templates,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 5, pp. 1188–1202, 2019.

[8] H. O. Shahreza, V. K. Hahn, and S. Marcel, “Face reconstruction from deep facial embeddings using a convolutional neural network,” in *2022 IEEE International Conference on Image Processing (ICIP)*, pp. 1211–1215, 2022.

[9] X. Dong, Z. Miao, L. Ma, J. Shen, Z. Jin, Z. Guo, and A. B. J. Teoh, “Reconstruct face from features based on genetic algorithm using GAN generator as a distribution constraint,” *Computers & Security*, vol. 125, p. 103026, 2023.

[10] C. Song and A. Raghunathan, "Information leakage in embedding models," in *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pp. 377–390, 2020.

[11] J. H. Cheon, W. Kim, and J. H. Park, "Efficient homomorphic evaluation on large intervals," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 2553–2568, 2022.

[12] J. H. Cheon, D. Kim, and D. Kim, "Efficient homomorphic comparison methods with optimal complexity," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 221–256, Springer, 2020.

[13] X. Jiang, M. Kim, K. Lauter, and Y. Song, "Secure outsourced matrix computation and application to neural networks," in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pp. 1209–1222, 2018.

[14] S. Hong, J. H. Park, W. Cho, H. Choe, and J. H. Cheon, "Secure tumor classification by shallow neural network using homomorphic encryption," *BMC genomics*, vol. 23, no. 1, pp. 1–19, 2022.

[15] Larxel, "Face mask detection." `https://www.kaggle.com/datasets/andrewmvd/face-mask-detection`, 2020.

[16] J. Yang, R. Shi, D. Wei, Z. Liu, L. Zhao, B. Ke, H. Pfister, and B. Ni, "Medmnist v2-a large-scale lightweight benchmark for 2d and 3d biomedical image classification," *Scientific Data*, vol. 10, no. 1, p. 41, 2023.

[17] A. Coucke, A. Saade, A. Ball, T. Bluche, A. Caulier, D. Leroy, C. Doumouro, T. Gisselbrecht, F. Caltagirone, T. Lavril, *et al.*, "Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces," *arXiv preprint arXiv:1805.10190*, 2018.

[18] J.-W. Lee, H. Kang, Y. Lee, W. Choi, J. Eom, M. Deryabin, E. Lee, J. Lee, D. Yoo, Y.-S. Kim, *et al.*, "Privacy-preserving machine learning with fully homomorphic encryption for deep neural network," *IEEE Access*, vol. 10, pp. 30039–30054, 2022.

[19] M. K. Titsias, "One-vs-each approximation to softmax for scalable estimation of probabilities," *Advances in Neural Information Processing Systems*, vol. 29, 2016.