Encryption-friendly LLM Architecture

Donghwan Rho*[1], Taeseong Kim*[1], Minje Park[2], Jung Woo Kim[2], Hyunsik Chae[1], Ernest K. Ryu†[3], Jung Hee Cheon†[1]

[1]Department of Mathematical Sciences, Seoul National University
[2]CryptoLab Inc. [3]UCLA, Department of Mathematics

*Equal contribution †Co-senior authors

ICLR 2025, March 26, 2025

Why We Consider HE?

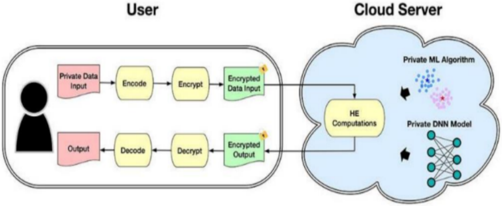
HE: Homomorphic Encryption, we use **CKKS**.

Many private data:

- Financial, medical images, $\cdots$.
- Hard to pre-process (eliminating private information, etc.).

Under HE,

- we can perform computations in an encrypted state.
- privacy is guaranteed.



Why Hard to Implement Under HE?

Implementation is too slow.

- Only support addition and multiplication.
  Hard to perform division, if statement, non-polynomial $\cdots$.
- Inference and fine-tuning **cost** more than plaintext.

Fine-tuning 2 layers of transformer blocks, 5 epochs:

Table: The times required to fine-tune for GLUE tasks with 8 RTX-4090 GPUs.

Task	**CoLA**	**MRPC**	**RTE**	**STS-B**	**SST-2**	**QNLI**
Time (h)	128.8	55.25	37.4	86.62	1016	1579

- In plaintext, RTE < 5 minutes / All tasks < 1 hour.

Why Hard to Implement Under HE?

Detailed forward evaluation time for 1 layer of transformer block:

- Hidden dimension: 768, Sequence length: 128

Table: The times required for a forward evaluation step with one RTX-4090 GPU per each operation.

Operation	Time (s)	Ratio (%)	
Softmax	8.43	43.77	Softmax: 43.77%
PCMM[1]	1.36	7.06	Matrix Multiplication: 38.47%
CCMM[2]	1.82	9.45	
BTS (Matmul.)	4.23	21.96	
LayerNorm	0.59	3.06	Non-polynomial Functions
ReLU	1.07	5.56	
BTS	1.75	9.09	
Etc	0.01	0.05	
Total	19.26	100	

Main bottlenecks: Softmax and Matrix multiplication.

[1] Plaintext-ciphertext matrix multiplication

[2] Ciphertext-ciphertext matrix multiplication

Contributions

Contributions are three-folds:

- Replacing Softmax with Gaussian kernel (GK):
  - Deleted division and $\max$.

- Use of LoRA (Low-Rank Adaptation) for speedup:
  - New application of LoRA under HE!
  - Converted Large CCMMs into Small CCMMs and Large PCMMs.
- Demonstrating **the first fine-tuning** of a transformer under HE !

Using these methods, speedups:

6.94× for fine-tuning / **2.3**× for inference!

Gaussian Kernel Replacing Softmax

- Softmax:

$$\mathrm{Softmax}\left(x_1, x_2, \cdots, x_n\right)_i = \frac{\exp\left(x_i - \alpha\right)}{\sum_j \exp\left(x_j - \alpha\right)}, \quad \text{where } \alpha = \max_{1 \leq j \leq n}\left\{x_j\right\}.$$

  - Bottlenecks: $\exp$, division, $\max$.
  - Most costly: $\max$ (about 80%)

- Gaussian kernel (GK):

$$\text{GK-Attention}(Q, K, V) = S(Q, K)V$$

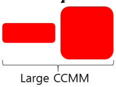
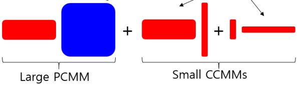
$$S(Q, K)_{ij} = \exp\left(-\frac{1}{2\sqrt{n}}\left\|Q_{i,:} - K_{j,:}\right\|_2^2\right), \; i, j = 1, \ldots, L.$$

- $\exp$ is easy to approximate: $\exp(x) \approx \left(1 + \frac{x}{2^k}\right)^{2^k}$ on $[-2^k, 0]$.
- There are no division and $\max$!

LoRA Reducing Large CCMMs

Under HE, there are two types of matrix multiplications:

- PCMM: plaintext-ciphertext matrix mult. Faster than CCMM.
- CCMM: ciphertext-ciphertext matrix mult.

Full Fine-tuning	LoRA Fine-tuning
Linear Layer	
Inference: $\underbrace{X(W+\Delta W)}$  Large CCMM	$A \in \mathbb{R}^{d\times r}, B \in \mathbb{R}^{r\times d}, r \ll d$ Inference: $\underbrace{XW} + \underbrace{(\underbrace{X}\underbrace{A})}B$  + + Large PCMM Small CCMMs
Large CCMM	Large PCMM + Small CCMMs

LoRA : Large CCMM → Large PCMM + small CCMMs !

Speedup Results

Table: Speedup results with our methods. SM means Softmax and Full means full fine-tuning.

	Fine-tuning		Inference	
	Full+SM	LoRA+GK(Ours)	Full+SM	LoRA+GK(Ours)
Time (s)	423.55	61.03	61.84	26.5
Factor	1	**6.94**	1	**2.33**

GLUE Scores

Average GLUE Scores:

	Plaintext Fine-tuning			Ciphertext Fine-tuning
	Full+SM	Full+GK	LoRA+GK(Ours)	LoRA+GK(Ours)
GLUE Score	0.7068	0.7098	**0.6772**	**0.6621**

- Our method achieves comparable GLUE scores to the Full + SM baseline.
- Fine-tuning on ciphertext preserves model performance without degradation!

Speedups Become Larger As Dimension Increase

Our speedups become larger as the hidden dimension increase!

- n: hidden dimension.

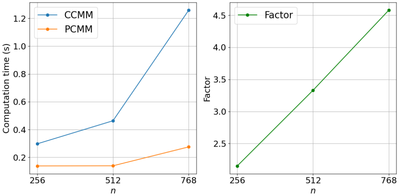


Figure: PCMM vs. CCMM

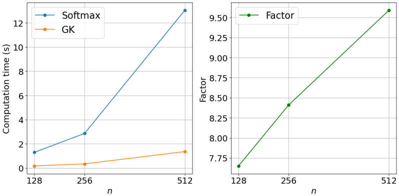


Figure: SM vs. GK

Thank you!