# Implementing Efficient Language Models under Homomorphic Encryption

Donghwan Rho

Department of Mathematical Sciences
Seoul National University

November 11, 2025

# Outline

# Outline

# LLMs and Privacy Issue

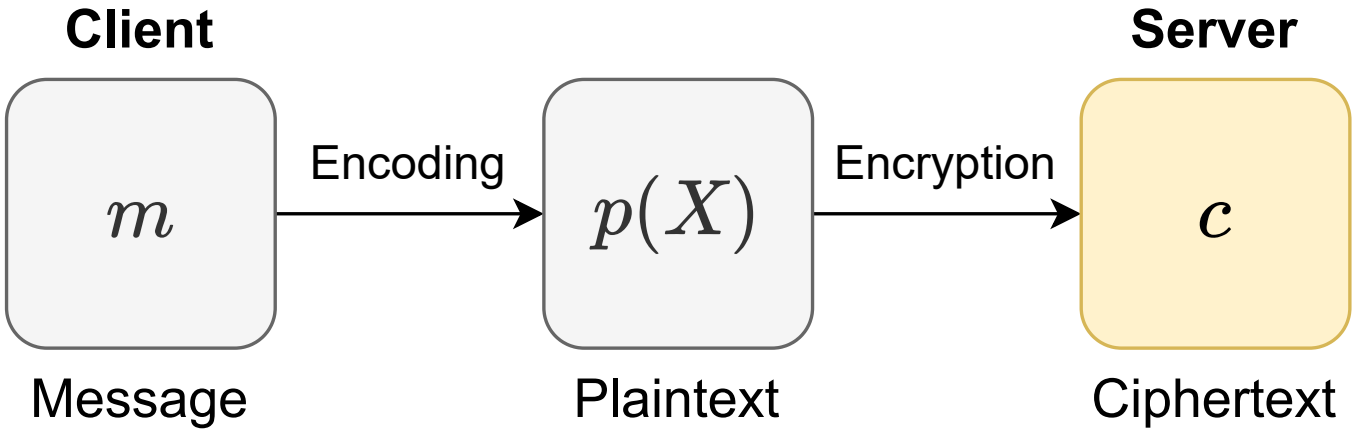
Can we use LLMs securely?

**ChatGPT banned in Italy over privacy concerns**

Samsung Bans ChatGPT Among Employees After Sensitive Code Leak

# What is CKKS?

- **CKKS** is a homomorphic encryption (HE) scheme.
- After decryption, we get an approximation of a plaintext.
- For a power of 2 $N$ and an integer $q$:
  - Message: $z \in \mathbb{C}^{N/2}$
  - Plaintext: $p(X) \in \mathbb{Z}[X]/(X^N + 1)$
  - Ciphertext: $c \in (\mathbb{Z}_q[X]/(X^N + 1))^2$



# What is CKKS? (continued)

- Possible operations: For plaintext $m_1$ and $m_2$,

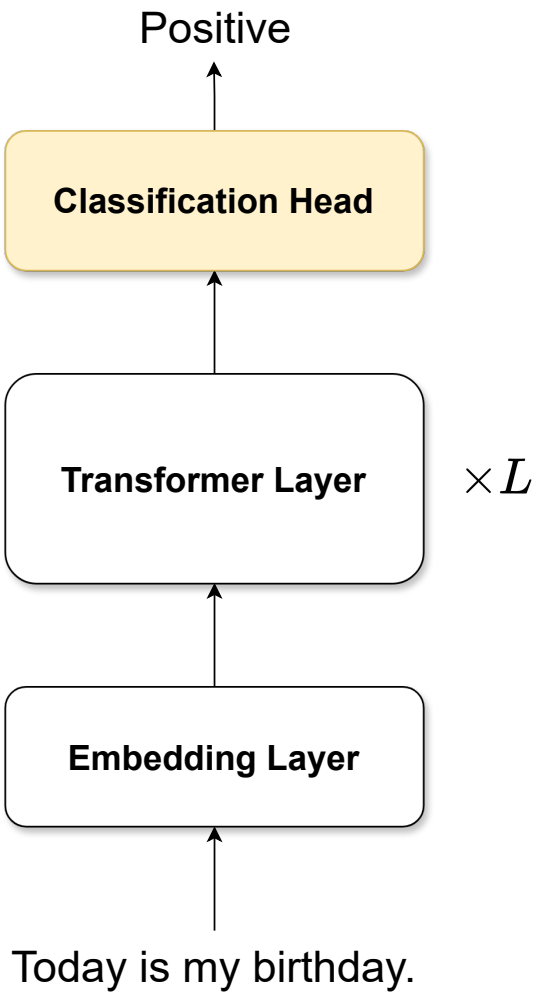
  1. Addition: $\mathrm{Dec}\left(\mathrm{Enc}\left(m_1\right) \oplus \mathrm{Enc}\left(m_2\right)\right) \approx m_1 + m_2$
  2. Multiplication: $\mathrm{Dec}\left(\mathrm{Enc}\left(m_1\right) \odot \mathrm{Enc}\left(m_2\right)\right) \approx m_1 \cdot m_2$
  3. Rotation: For $m = (z_0, \ldots, z_{s-1}) \in \mathbb{C}^s$ and $0 \leq r < s$,

     $$\mathrm{Dec}(\texttt{Lrot}(\mathrm{Enc}(m), r)) \approx (z_r, \ldots, z_{s-1}, z_0, \ldots, z_{r-1}).$$

     Similarly, `Rrot` can be defined.

- CKKS is a *leveled* HE scheme:
  - Can do operations up to $L$ times.
  - Need `Bootsrapping` to continue operations.

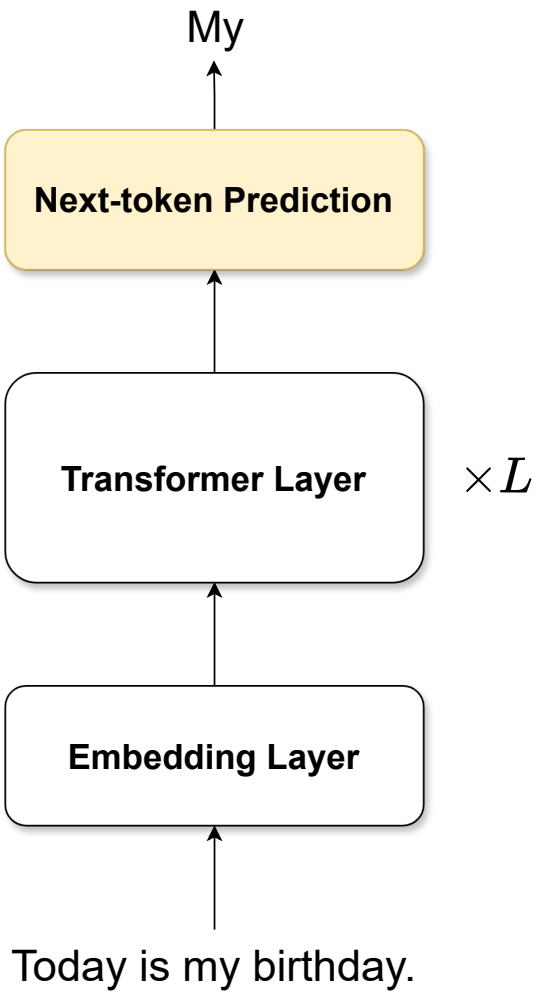
# LLM – Encoder-only Model




Encoder-only model:

- Classification:
  - Positive / Negative
  - Cat / Dog
- BERT-series
- Need fine-tuning:
  - For classification head

# LLM – Decoder-only Model

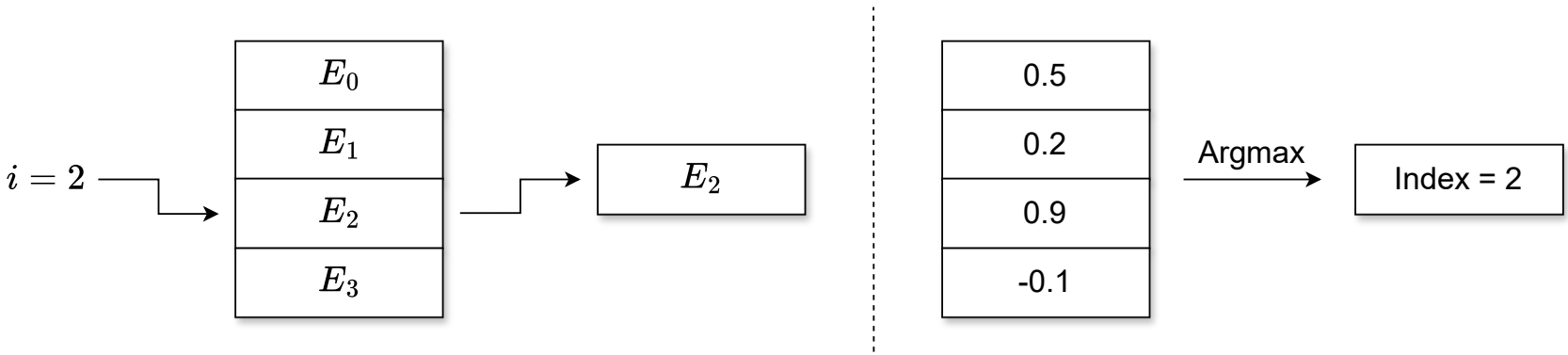



Decoder-only model:

- Text generation:
  - My
  - My friend
  - My friend gave
- GPT/Llama-series
- Need fine-tuning:
  - For domain adaptation

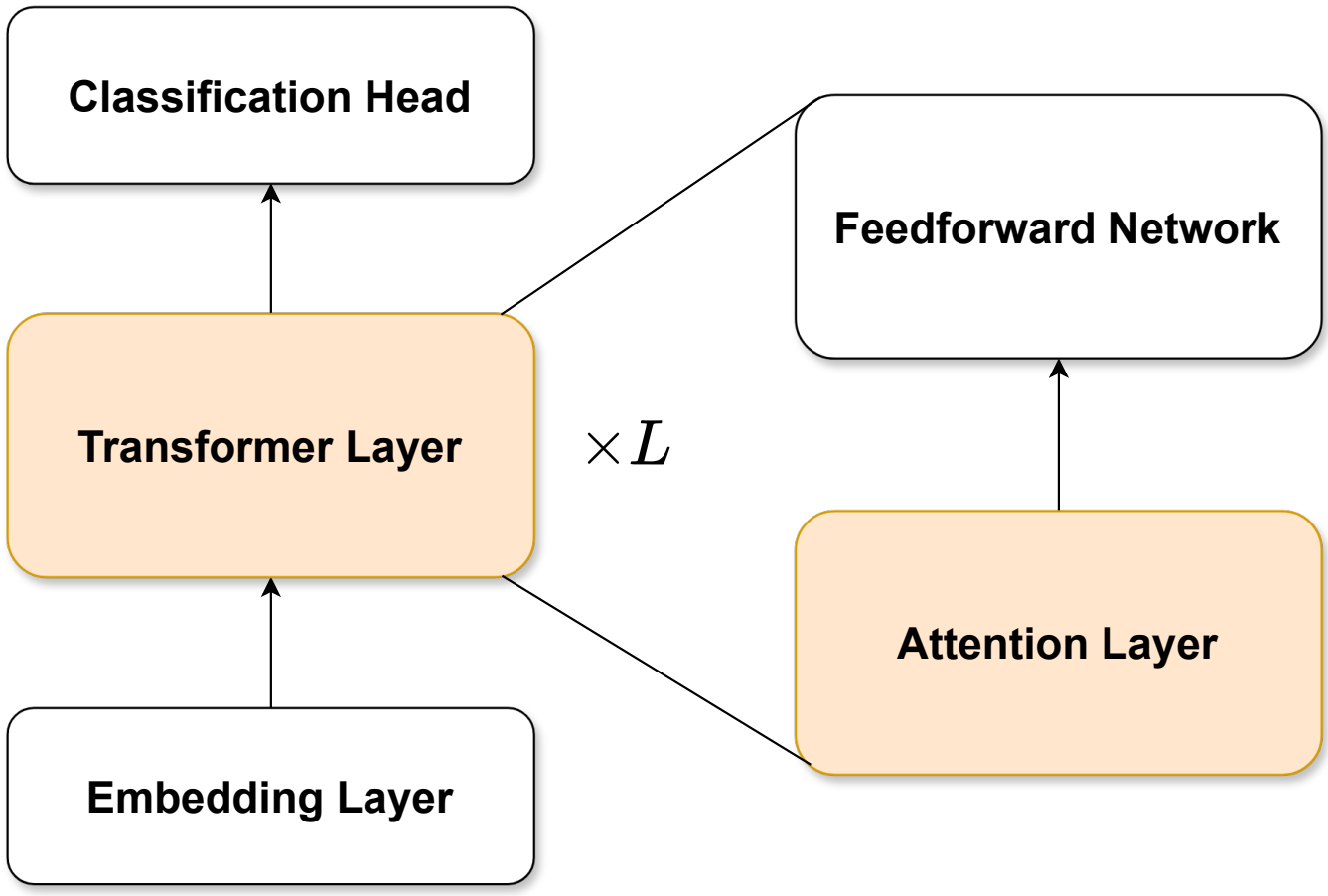
# Hard Operations under CKKS

- Non-polynomial functions: (→Approximation)
  - Division
  - $\mathrm{Softmax}(x_0, \ldots, x_n)_i = \dfrac{e^{x_i}}{\sum_{i=0}^{k} e^{x_i}}$
  - $\mathrm{ReLU}(x) = \max(0, x)$
- Matrix multiplication between ciphertexts (CCMM) (→Reduction)
- Look-up table & `If` statement (→Detour)

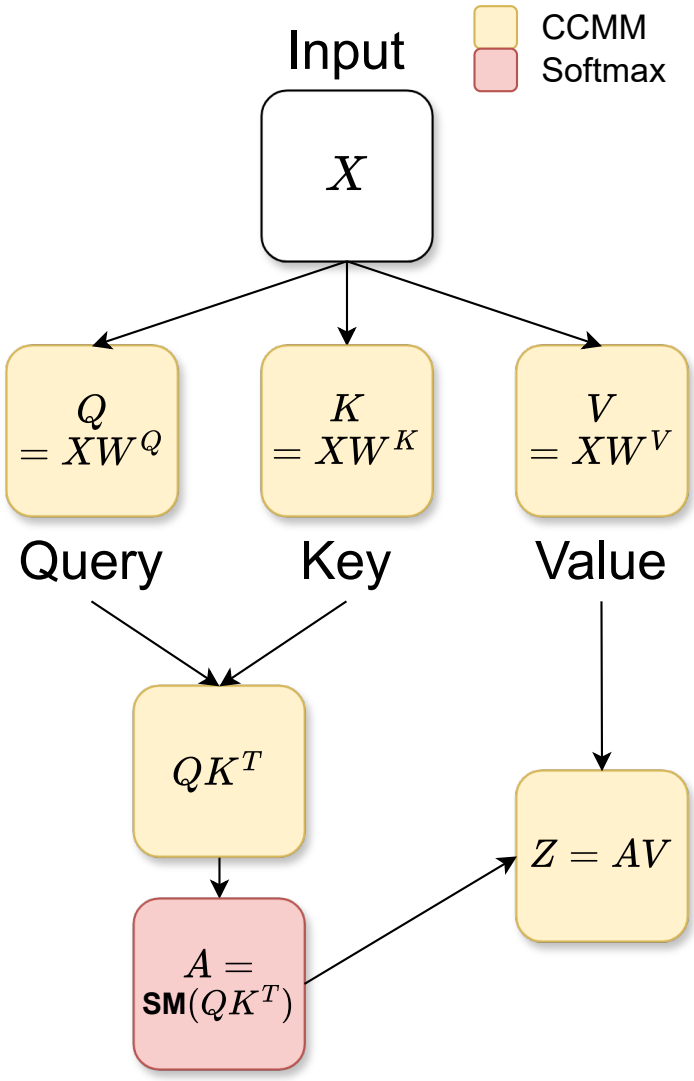


# Outline

# Scope of Part (i)



# Attention Layer




- Fine-tuning:
  - $Q, K, V$ become ciphertexts.
- Inner product ($QK^T$): Similarity between tokens
- Bottlenecks:
  - Softmax
  - Many CCMMs

# Why Hard to Implement Under HE?

Implementation / fine-tuning is too slow.

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, all tasks require $< 1$ hour.

# Why Hard to Implement Under HE? (continued)

Forward evaluation time for 1 layer of transformer block:

Table: Forward pass time by operation (1× RTX-4090).

| Operation | Time (s) | Ratio (%) |
|---|---|---|
| Softmax | 8.99 | 29.42 |
| CCMM[1] | 13.36 | 43.71 |
| BTS (Matmul.) | 4.20 | 13.74 |
| LayerNorm | 0.52 | 1.71 |
| ReLU | 1.51 | 4.94 |
| BTS | 1.51 | 4.94 |
| Save | 0.47 | 1.54 |
| Total | 30.56 | 100 |

**Softmax: 29.42%**

**Matrix Multiplication: 57.45%**

Non-polynomial Functions

Main bottlenecks: **Softmax** and **Matrix multiplication**.

[1] Ciphertext-ciphertext matrix multiplication

# Contributions

Contributions:

- Replacing Softmax with **Gaussian kernel (GK)**:
  - Deleted division and $\max$.
- Use of **LoRA** (**Lo**w-**R**ank **A**daptation) for speedup:
  - New application of LoRA under HE!
  - Large CCMMs $\longrightarrow$ Small CCMMs + Large PCMMs
- **The first fine-tuning** of a transformer under CKKS !

Speedups:

**6.94**$\times$ for fine-tuning / **2.3**$\times$ for inference!

# Softmax $\longrightarrow$ GK

- Softmax:

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

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

- Gaussian kernel (GK):

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

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

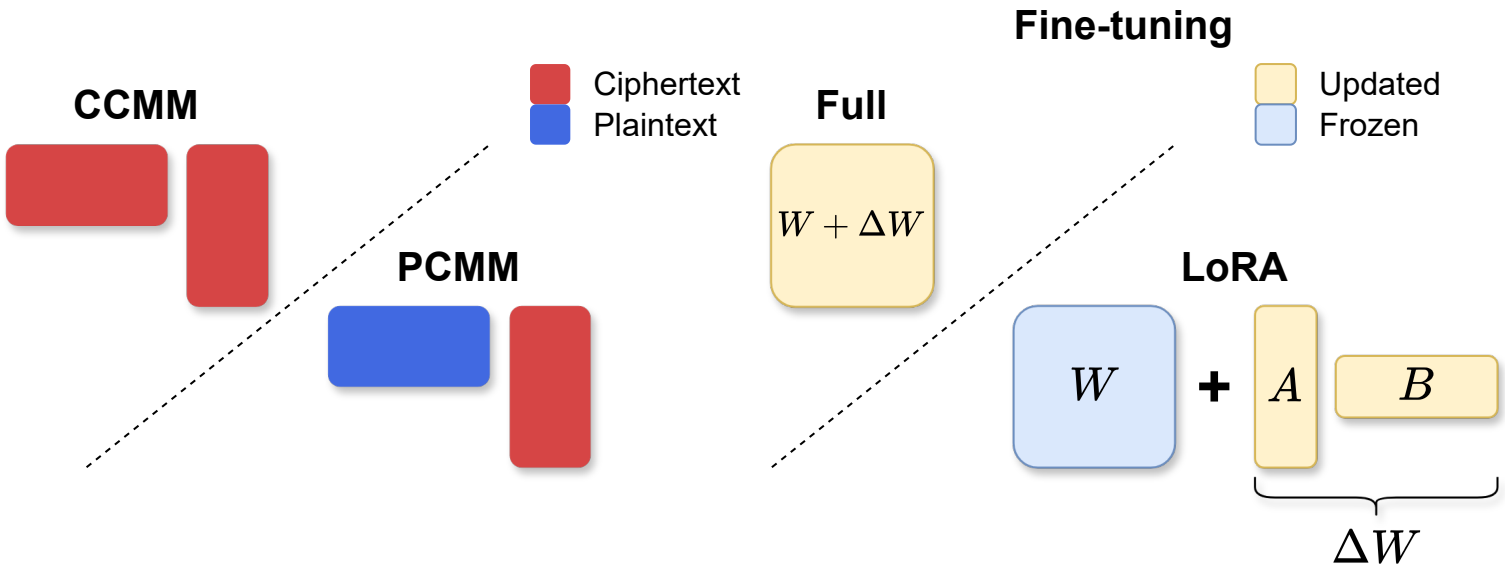
- No **division**: automatically bounded output !
- No **max**: the input of exp is always negative !

# LoRA Reduces Large CCMMs

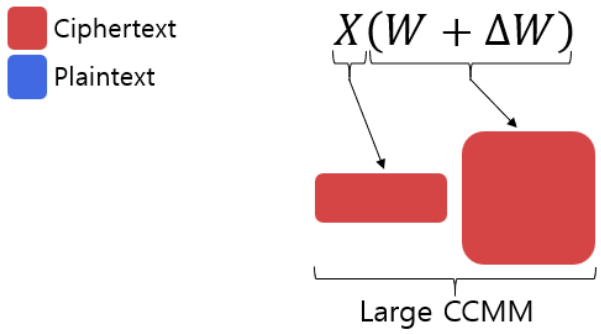
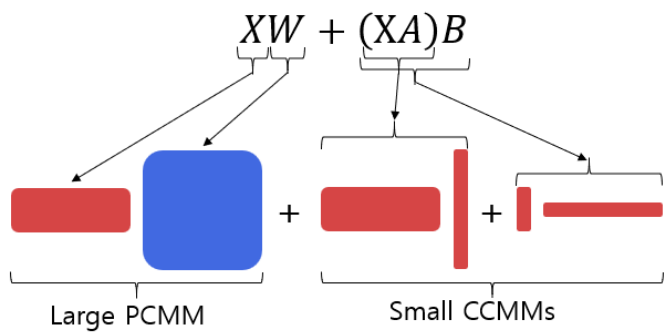
Under HE, two types of matrix multiplications:

- CCMM: ciphertext-ciphertext matrix mult
- PCMM: plaintext-ciphertext matrix mult
  - **Faster** than CCMM
  - More faster than CCMM for larger matrices

LoRA: Freeze a weight $W$ and update $W$ by $W + \Delta W = W + AB$:



# LoRA Reduces Large CCMMs (continued)

| Full Fine-tuning | LoRA Fine-tuning |
|---|---|
| Linear Layer, inference $Y = X(W + \Delta W)$ | |
|   |   |
| Large CCMM | Large PCMM + Small CCMMs |

LoRA can reduce **CCMM** size !

# Speedups

Table: Speedup results with our methods.

| | **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** |

- SM: Softmax
- Full: Full fine-tuning

# GLUE Scores

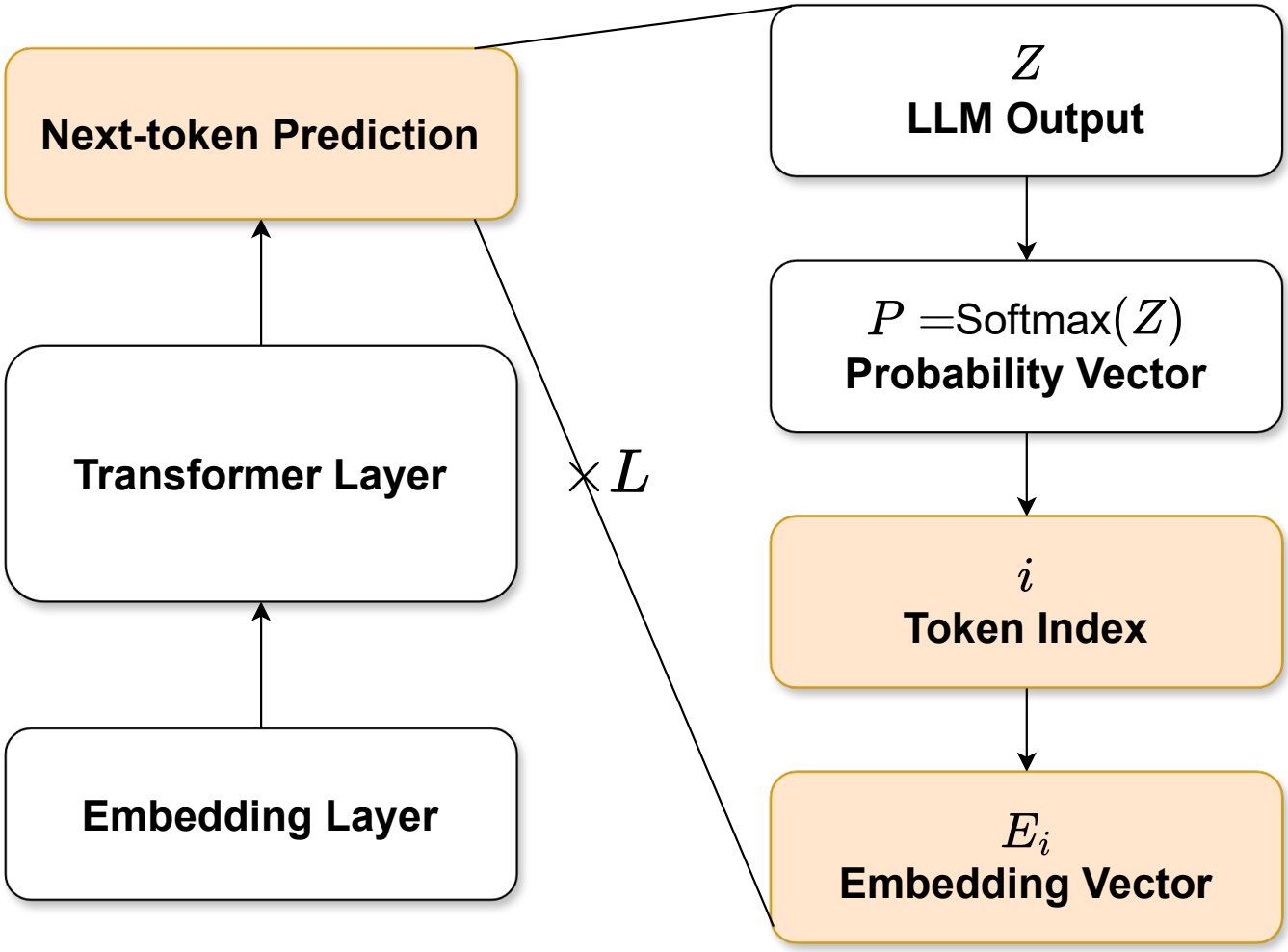
Average GLUE Scores:

| | Plaintext | | | Ciphertext |
|---|---|---|---|---|
| | Full+SM | Full+GK | LoRA+GK(Ours) | LoRA+GK(Ours) |
| GLUE Scores | 0.7068 | 0.7098 | **0.6772** | **0.6621** |

- Achieves comparable GLUE scores to the Full + SM baseline.
- Fine-tuning on ciphertext preserves performance without degradation.

# Outline

# Scope of Part (ii)



# No Random Sampling Under CKKS

No random sampling under CKKS yet.

Goal: **Text generation** with Random sampling under CKKS !

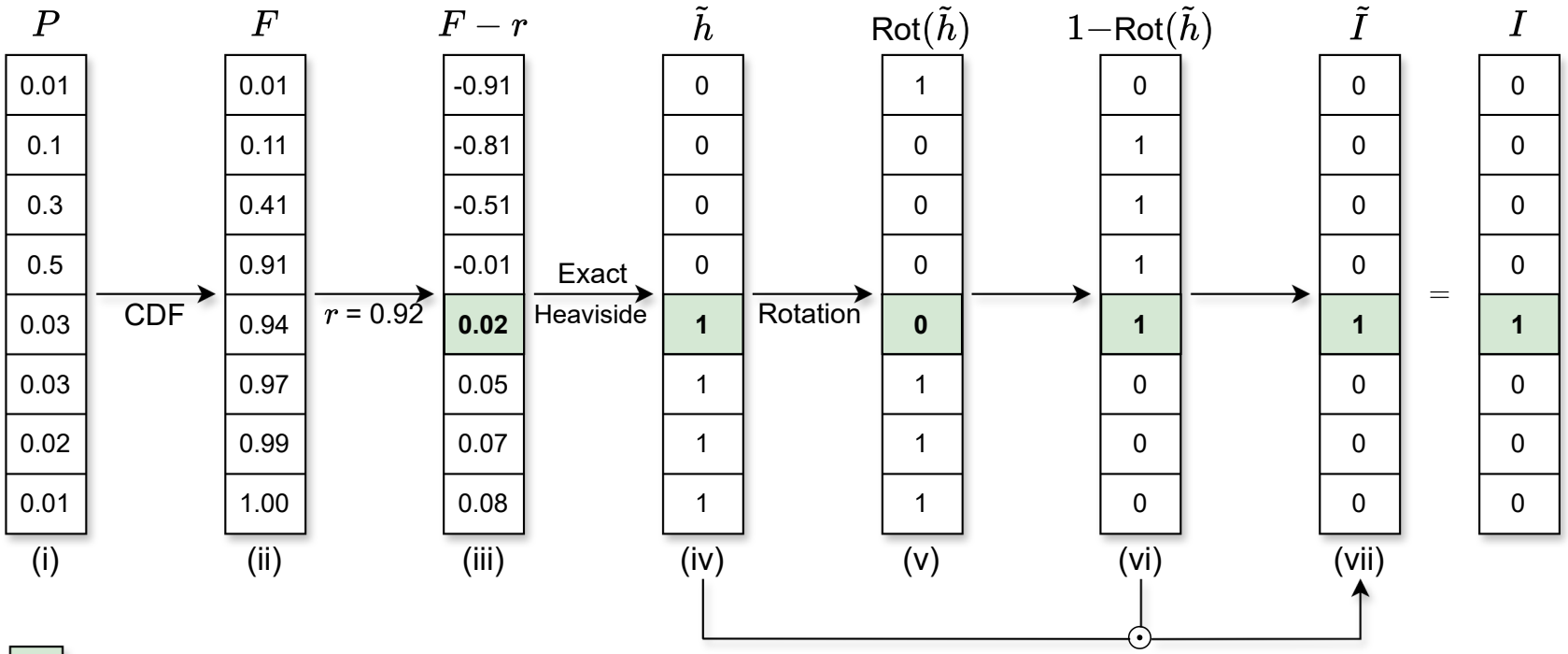
Result: An embedding vector obtained by multiplying

a **one-hot vector** $I$ with the **embedding matrix** $E$ .

$$\underbrace{\begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix}}_{\substack{I^\top \\ \text{One-hot}}} \underbrace{\begin{pmatrix} - & E_0 & - \\ - & E_1 & - \\ - & E_2 & - \\ - & E_3 & - \end{pmatrix}}_{\text{Embedding weight } E} = \underbrace{E_2}_{\text{Embedding vector}}$$

# Our Algorithm – Exact Version

- No look-up, only one rotation, and SIMD operations.




Selected next-token index

$P$: Probability vector

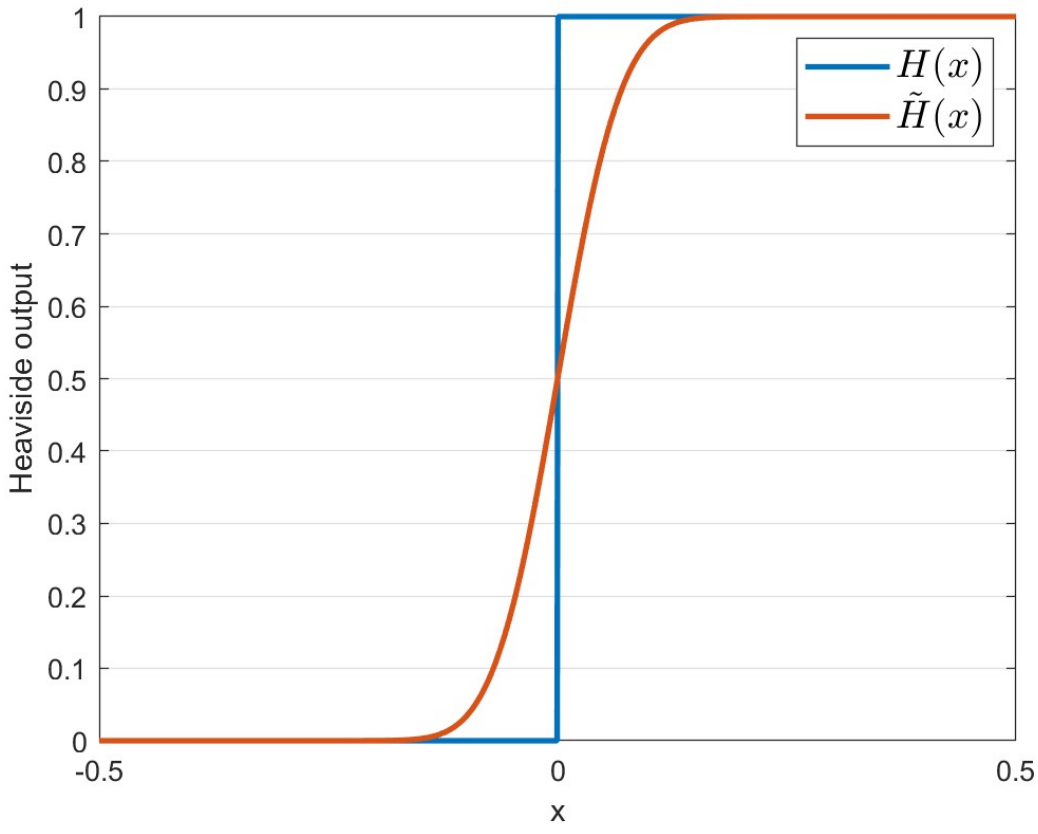
$F$: CDF vector of $P$

$r$: Random number in [0, 1]

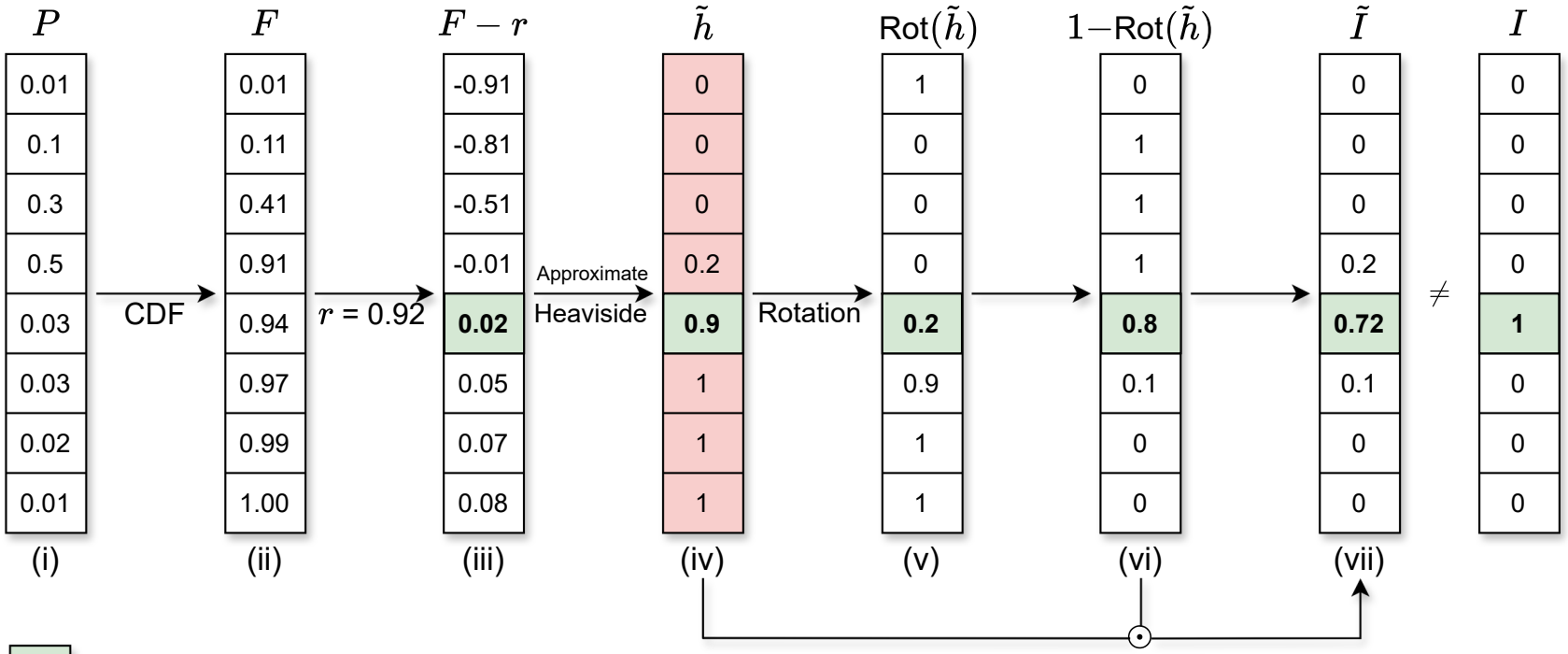
$\odot$: Element-wise multiplication

# Main Problem

We must approximate the Heaviside function $H(x) = \frac{\mathrm{sgn}(x)+1}{2}$.



# Our Algorithm – Approximate Version




Selected next-token index

Approximate value

$P$: Probability vector

$F$: CDF vector of $P$

$r$: Random number in [0, 1]

$\odot$: Element-wise multiplication

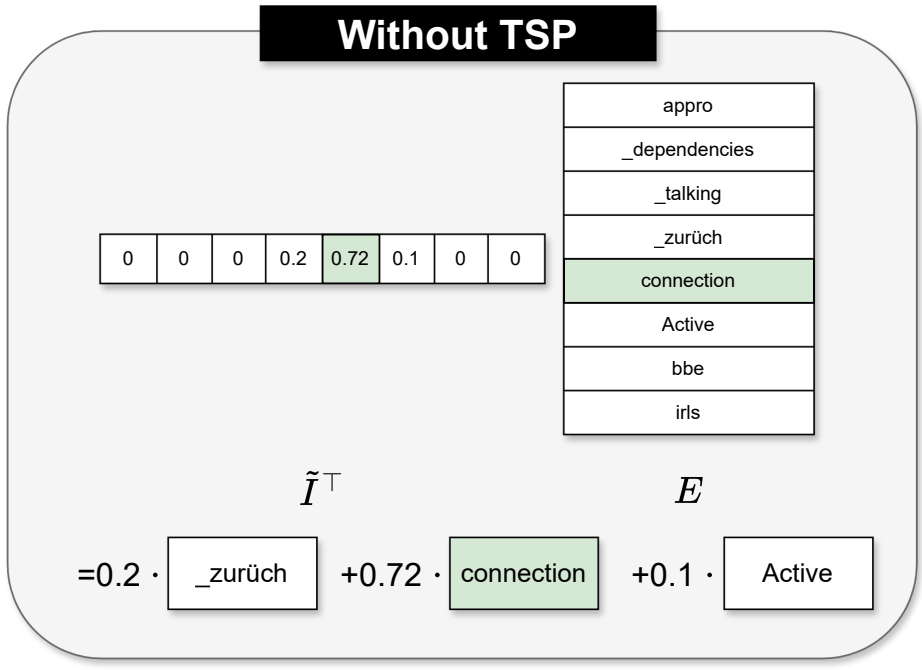
Problem : $\tilde{I}$ is not a one-hot vector !

# Problematic Case

| Prompt | Please introduce yourself. |
|---|---|
| Answer | Please introduce yourself. Not long after finishing the university in Moscow in H Physi am a friend at theTEenza stopped escape to Moscow State University where our political large a large number have metita I T a All Message When As Moscow USoni M MS To was D MSNewisMSMSH MSMSA MSMSMSMS MS M. MM SWMMMS areMSMSIIMSMS M ··· MSMMSMSMSMSMSMSMSMS STyMSMMS C IIIPMSMSMSMSMSMSMSMSMSMSMS5 MSMS GM-SLAMSMS MSMSMSMSMS Project MSMSMSMSMSMSMS MSMS MSMSMS MSMSPMSMM SKMSMSMSMSMSMSMSMMSMSMSMSMSMWMS MSMSMMSMSMSMSMSMSMSMSMSMSMSMSMSMSMS MSMS MSMSM MarMSMSMSMSWMSMSMSMSMS MSMSMSMSMSMSMSMS MSMS MSMSMSPAMSMSMSMSMSMS MSMS MSMSMSLMSMSMS MMSMSMSMS'MSMSMSMMSMMSMSIMSMSMMS MS MPA |

# Problem 1: Different Meanings for Adjacent Tokens

| Index | Token |
|---|---|
| 9962 | appro |
| 9963 | _dependencies |
| 9964 | _talking |
| 9965 | _zurüch |
| 9966 | **connection** |
| 9967 | Active |
| 9968 | bbe |
| 9969 | irls |




Problem 1:

A linear combination of semantically **not similar** embeddings.

# Solution 1: Token Index Reordering via TSP

We reorder the indices of tokens using a **traveling salesman problem (TSP)** algorithm.

We solve the following optimization problem:

$$\underset{\pi}{\operatorname{argmin}} \sum_{i=1}^{|\mathcal{V}|-1} d_{\cos}(E_{\pi(i)}, E_{\pi(i+1)}),$$

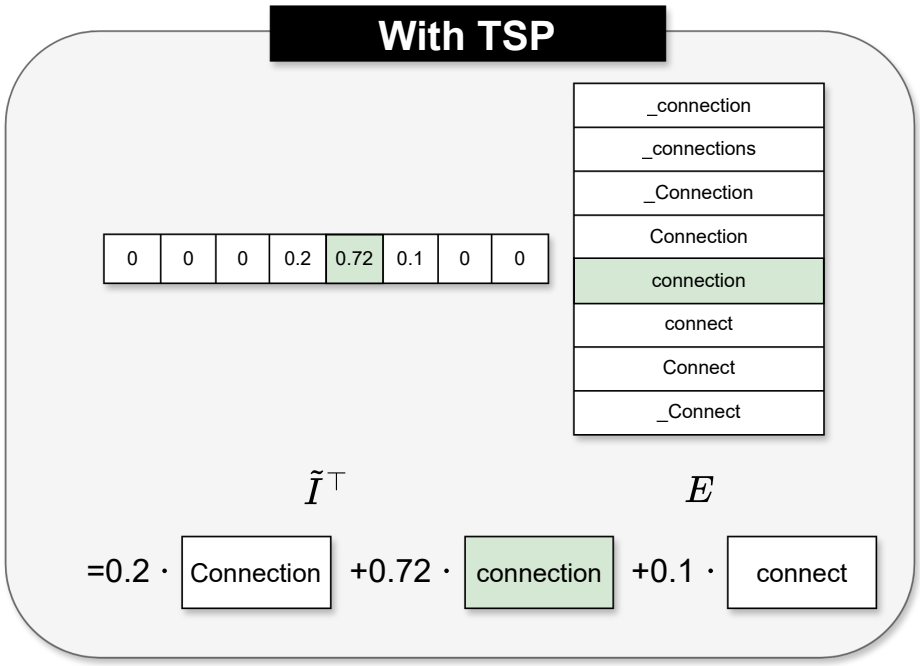
where

- $E_i$ is the $i$-th embedding vector.
- $d_{\cos}$ means the cosine distance.
- $\pi$ ranges over all permutations of the token indices.

# Solution 1: Token Index Reordering via TSP (cont'd)

Semantically similar tokens are placed adjacently.

| Index | Token |
|---|---|
| 18927 | _connection |
| 18928 | _connections |
| 18929 | _Connection |
| 18930 | Connection |
| 18931 | **connection** |
| 18932 | connect |
| 18934 | Connect |
| 18935 | _Connect |




Solution 1:
Token index reordering via a **Traveling Salesman Problem (TSP)** !

# Problem 2: The Sum of $\tilde{I}$ is Different from 1

Obtained $\tilde{I}$ can be different from a one-hot vector.

$$\tilde{I}^\top = \begin{array}{|c|c|c|c|c|c|c|c|} \hline 0 & 0 & 0 & 0.2 & \mathbf{0.72} & 0.1 & 0 & 0 \\ \hline \end{array} \neq I^\top$$

- Case 1: The sum of the weights of $\tilde{I}$ exceeds 1 significantly.
- Case 2: The sum of the weights of $\tilde{I}$ is around 1, but the dominant element is quite different from 1.

| | Top $i$-th Element of $\tilde{I}$ | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | $\cdots$ | 512 | Sum |
| Case 1 | 0.4575 | 0.4335 | 0.1816 | 0.0942 | $\cdots$ | 0.0049 | 3.9572 |
| Case 2 | 0.9292 | 0.0413 | 0.0227 | 0.0198 | $\cdots$ | 0.0000 | 1.0133 |

# Solution 2: Post-processing

To mitigate this, apply a **post-processing**: $\mathrm{PP}(x) = -2x^3 + 3x^2$.

$\tilde{I}^\top =$ | 0 | 0 | 0 | 0.2 | **0.72** | 0.1 | 0 | 0 | $\neq I^\top$

$\mathrm{PP}(\tilde{I}^\top) =$ | 0 | 0 | 0 | 0.10 | **0.81** | 0.03 | 0 | 0 |

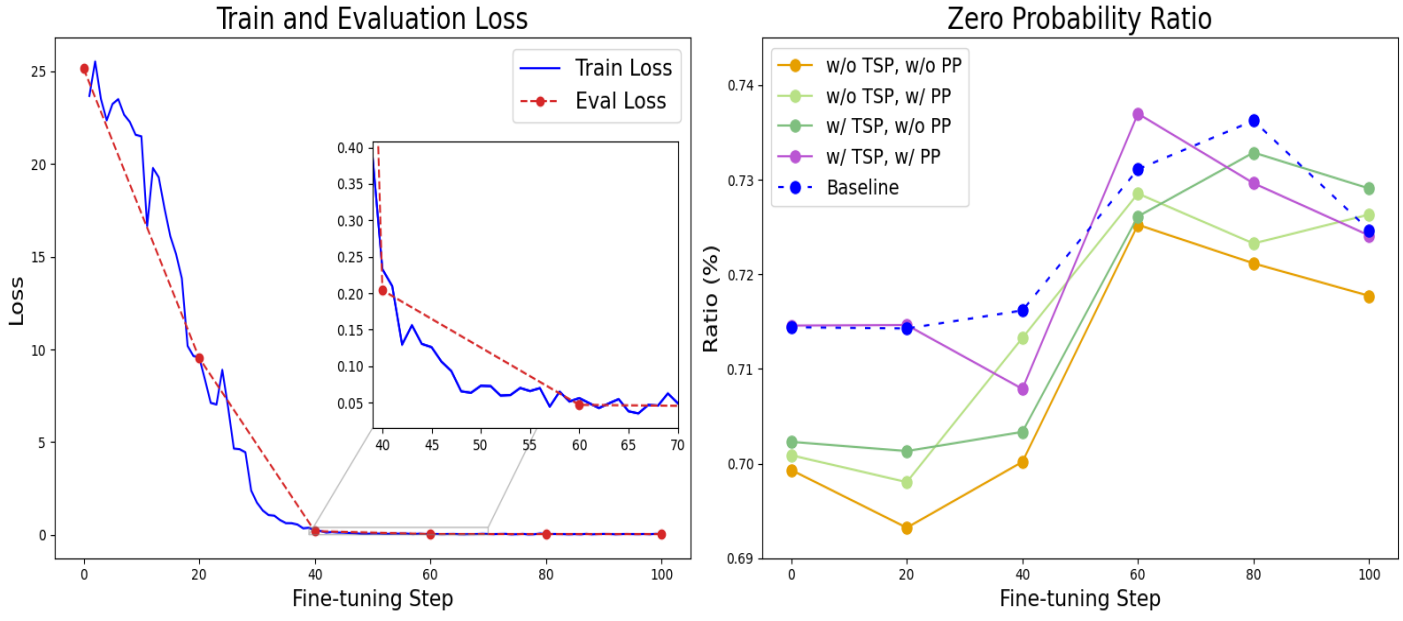
| | | Top $i$-th Element of $\tilde{I}$ | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Post-processing | 1 | 2 | 3 | 4 | · · · | 512 | Sum |
| Case 1 | False | 0.4575 | 0.4335 | 0.1816 | 0.0942 | · · · | 0.0049 | 3.9572 |
| | True | 0.4364 | 0.4008 | 0.0870 | 0.0249 | · · · | $7.18e-5$ | **1.0504** |
| Case 2 | False | 0.9292 | 0.0413 | 0.0227 | 0.0198 | · · · | 0.0000 | 1.0133 |
| | True | **0.9857** | 0.0050 | 0.0015 | 0.0012 | · · · | 0.0000 | 0.9934 |

Solution 2:
Applying **post-processing** (a polynomial) !

# Problem 3: Tokens with Low Probability Are Selected

Our algorithm is the probabilistic sampling.
$\Rightarrow$Tokens with low probability can be selected.

Fine-tuning makes the number of **zero-probability tokens** increase.




Solution 3:

**Fine-tuning** increases the ratio of zero-probability tokens !

# Evaluation Metric

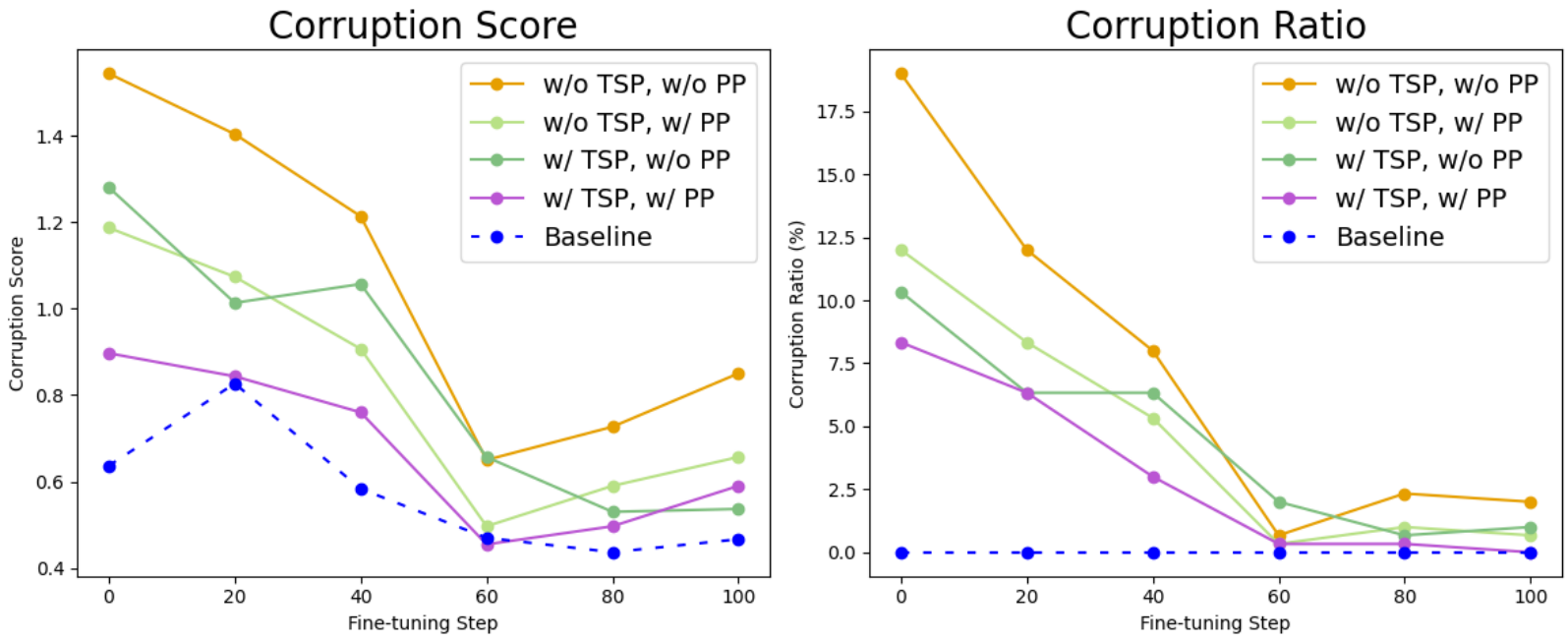
- Metric: **Corruption Score** / **Corruption Ratio**
  Corrupted: If a text contains meaningless '**MS**' tokens
- Corruption Ratio: How many texts are corrupted?
- Corruption Score: Ask GPT-4 API to evaluate generated texts.
  Score: 0 / 1 / 2 / 3 / 4
- For all metrics, lower is better.

# GPT-4 API Prompt

| Prompt | Please introduce yourself. |
|---|---|
| Criteria | I'm going to give you a piece of writing. This text was generated by an LLM using random sampling. Please determine whether or not this text is corrupted. The criteria for being considered corrupted are as follows:<br><br>When a specific character is repeated meaninglessly. For example, something like Coooooooooooooool! has meaningful repetition, so it wouldn't be considered corrupted. However, something like MSMSMSMSMS...—a meaningless sequence of repeated characters—would be considered corrupted.<br><br>When the arrangement of words is excessively random to the point where the text is completely unintelligible. Random sampling can result in some randomness in sentences, so a text with a reasonable degree of randomness wouldn't be considered corrupted. However, if the randomness is excessive to the point where the text becomes utterly unreadable, it would be considered corrupted. However, since the current text was generated to match a specific token count, please disregard any incomplete sentences at the end.<br><br>After reading the text, assign a score based on the degree of corruption in the following format:<br>**X point(s): REASON**<br><br>Here is the scoring system:<br>4 points: If 80-100% of the text is corrupted.<br>3 points: If 60-80% of the text is corrupted.<br>2 points: If 40-60% of the text is corrupted.<br>1 point: If 20-40% of the text is corrupted.<br>0 points: If 0-20% of the text is corrupted.<br><br>**Special Case:** Regardless of the above criteria, if the sequence MS is repeated meaninglessly more than two times, assign **4 points**.<br><br>Here is the text I'll show you: |

# Experimental Results

Our all methods work and are necessary !



# Future Directions

- Advances in CKKS
- HE-friendly algorithms
- HE-friendly model architectures

Thank you !

# References


1. Rho et al., Encryption-Friendly LLM Architecture. *ICLR*, 2025.
2. Rho et al., Traveling Salesman-Based Token Ordering Improves Stability in Homomorphically Encrypted Language Models. *arXiv:2510.12343*.
3. Cheon et al., Homomorphic encryption for arithmetic of approximate numbers. *ASIACRYPT*, 2017.
4. Jiang et al., Secure outsourced matrix computation and application to neural networks. *ACM CCS*, 2018.
5. Devlin et al., BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL*, 2019.
6. OpenAI, GPT-4 technical report. *arXiv:2303.08774*.
7. Dubey et al., The Llama 3 herd of models. *arXiv:2407.21783*.
8. HU et al., LoRA: Low-rank adaptation of large language models. *ICLR*, 2022.
9. Wang et al., GLUE: A multi-task benchmark and analysis platform for natural language understanding. *EMNLP Workshop*, 2018.