이학박사 학위논문

Exploring Efficient Implementation of Language Models Under Homomorphic Encryption

동형암호 하에서 언어 모델의 효율적인 작동에 관한 탐구

2026년 2월

서울대학교 대학원

수 리 과 학 부

노 동 환

Exploring Efficient implementation of Language Models Under Homomorphic Encryption

동형암호 하에서 언어 모델의 효율적인 작동에 관한 탐구

지도교수 강 명 주

이 논문을 이학박사 학위논문으로 제출함

2025년 10월

서울대학교 대학원

수 리 과 학 부

노 동 환

노동환의 이학박사 학위논문을 인준함

2025년 12월

위 원 장 _____ (인)

부위원장 _____ (인)

위 원 _____ (인)

위 원 _____ (인)

위 원 _____ (인)

Exploring Efficient Implementation of Language Models Under Homomorphic Encryption

A dissertation
submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
to the faculty of the Graduate School of
Seoul National University

by

Donghwan Rho

Dissertation Director : Professor Myungjoo Kang

Department of Mathematical Sciences
Seoul National University

February 2026

Abstract

Exploring Efficient Implementation of Language Models Under Homomorphic Encryption

Donghwan Rho

Department of Mathematical Sciences
The Graduate School
Seoul National University

As the Large Language Models (LLMs) such as ChatGPT work successfully and are offered as services, language model has been at the center of the latest developments in AI. LLMs are trained on an internet-scale data, demonstrating outstanding knowledge and fluent conversational capacity. Besides, LLMs can provide customized answers to each person's questions. However, personal data often have to be sent to the LLM provider's server to receive these personalized answers. Therefore, people who do not want to share their personal information, companies with proprietary technology, and institutions such as banks holding sensitive data are hesitant to use LLMs due to privacy concerns. Because of this privacy issue, privacy-preserving machine learning (PPML) has attracted attention along with the development of LLMs.

Homomorphic Encryption (HE) offers a promising solution to PPML. It supports operations in an encrypted data. If we use it, data are sent to a server in encrypted form from the beginning. Therefore, even if the data is leaked from the server, private information is protected. Among various HE schemes, we use CKKS (Cheon-Kim-Kim-Song), which supports real number operations, so it is appropriate for implementing machine learning models. CKKS is based on RLWE (Ring Learning with Errors), and since RLWE is equivalent to an NP-hard problem, the security of CKKS is guaranteed.

Even if CKKS has many strengths, it is prohibitively hard to use it practically because of the computational overhead. LLMs are based on the Transformer, which involves many non-polynomial operations such as Softmax; however CKKS only supports addition and multiplication.

In this thesis, we study the practical implementation of LLMs under HE by addressing these problems, building on the author's earlier work [99, 100]. In Chapter 3, we replace Softmax with a Gaussian kernel and reduce matrix multiplications between ciphertexts to reduce computational overhead. In Chapter 4, we propose a novel method for stable text generation when probabilistic sampling is performed under CKKS, which has not yet been developed.

Key words: Privacy-preserving Machine Learning(PPML), Large Language Model (LLM), Homomorphic Encryption, Cheon-Kim-Kim-Song (CKKS), Random Sampling
Student Number: 2021-21713

Contents

Chapter 1

Introduction

The advent of large language models (LLMs) such as the BERT series [34, 79, 104, 67, 31, 54], the GPT series [95, 96, 109, 91], and ChatGPT [92] kick-started a new era of natural language processing (NLP) and artificial intelligence (AI). One of the many capabilities of LLMs that has received much attention is their ability to provide personalized responses based on user interactions. However, this use case raises serious concerns about user privacy, since usually it requires sensitive private data to receive these personalized responses. In response, regulations such as the GDPR [42] and CCPA [106] have been amended. In Italy, ChatGPT was even temporarily banned [84], and several major corporations, including Apple and Samsung, have restricted its use within their companies [87]. This highlights the need for techniques that ensure secure use while maintaining user data privacy.

Privacy-preserving machine learning (PPML) refers to methods that use machine learning while protecting data privacy. Techniques for PPML include secure multi-party computation (MPC) [121], differential privacy [40], and homomorphic encryption (HE) [102, 47]. Among these, only MPC and HE offer provable security based on cryptographic assumptions. MPC utilizes communications between parties, but these communications can make it challenging to accelerate and parallelize the heavy computation of neural networks. In contrast, HE supports arithmetic computations in encrypted states without requiring communications. Since the pioneering work of Gentry [47], several HE schemes have been developed [11, 13, 38, 26, 22]. Notably, the CKKS [22] scheme is particularly efficient for evaluating large-scale real-valued (as opposed to integer-valued) data in parallel and is widely used in the PPML literature [52, 69, 70, 73, 74]. Additionally, its fully SIMD mode efficiently handles the large inner dimensions of AI primitives, making

it well-suited for rapid real-number operations in Transformer inference.

In theory, homomorphic encryption (HE) presents an elegant solution to the privacy concerns associated with LLMs. However, despite the significant recent progress in the theory and implementation of HE operations, protecting LLMs with HE remains a challenge due to their computational scale. Transformer models [114] famously rely on numerous matrix multiplications and various non-polynomial operations, and directly adapting these operations to HE results in significant computation time and precision loss.

Since transformers require large-scale **matrix multiplications**, the computational cost becomes considerable, particularly as the size of the model and input data increases. Matrix multiplication is a core operation in transformer models, playing a crucial role in the attention mechanism and other key components of the model. The complexity of these operations grows with the dimensionality of the data, which can result in significant computational and memory demands. To address this issue, several studies, including [18, 123, 88], have proposed efficient methods for optimizing matrix multiplication, aiming to reduce computational overhead while maintaining the performance and accuracy of the model.

Non-linear operations, such as Softmax, GELU, and LayerNorm, present additional difficulties due to the limitations of FHE, which supports only homomorphic arithmetic operations, such as homomorphic addition and multiplication. As a result, these non-linear operations must be approximated by polynomial functions, a process that is not straightforward. In [99, 56, 70, 74, 88, 94, 128], research has focused on developing efficient implementations of Softmax, GELU, and LayerNorm or replacing these functions with simpler alternatives.

Finally, particulary in decoder-only models [95, 96, 109, 91, 110, 111, 37], **next-token prediction**, such as the use of argmax or probabilistic sampling, introduces significant computational challenges. Unlike the non-linear operations mentioned previously, which apply element-wise operations to individual values, argmax is used to compare multiple values in order to select a single output. This comparison requires several approximated versions of the sign function. In [62, 125, 123], homomorphic implementations of argmax involved a substantial number of comparison and rotation operations, limiting their efficiency. However, to the best of our knowledge, no research has yet been conducted on the implementation of probabilistic sampling using CKKS.

In this thesis, we address three parts of language models:

i) **Softmax**, ii) **Matrix Multiplication**, and iii) **Next-Token Prediction**.

In section 3, we propose a modified HE-friendly transformer architecture with an emphasis on inference following personalized (private) fine-tuning. We point out that prior work on homomorphically encrypted transformers often overlooked fine-tuning due to its complexity. Our approach has two main algorithmic components: LoRA [57] fine-tuning and the replacement of softmax in attention with Gaussian kernels (GK). We show that LoRA and GK significantly accelerate the encrypted transformer computation under HE while maintaining performance levels comparable to those of the plaintext model. Experimental results of our modified BERT model on encrypted data using the CKKS scheme demonstrate its ability to *securely* process natural language data. Our findings show promise for offering privacy-preserving LLM services in areas where data protection is crucial.

Next, in Chapter 4, we focus on proposing a homomorphic encryption algorithm specifically designed to efficiently handle next-token selection, complementing existing solutions for matrix multiplication and non-linear operations. Since we approximately implement next-token prediction, the accuracy of next token selection process can decrease. To solve this, we use the Traveling Salesman Problem (TSP) [2] algorithm. Also, since we generate texts randomly, the generated texts are less coherent. To enhance coherence of generated texts, we analyze the effect of domain-specific fine-tuning in increasing coherence. More detailed explanation can be seen in Chapter 4.

Chapter 2

Preliminaries

2.1 Language Models

Attention mechanism Standard large language models (LLMs), such as BERT [34], GPT-4 [91], and Llama-3 [37] are constructed by stacking many transformer layers. We explain the standard attention mechanism in a transformer [114] layer. We do not explain the feed-forward network (FFN), another component of the transformer, as it has not been modified in our work. Suppose that L and n represent sequence length and embedding dimension, respectively. Given $Q, K, V \in \mathbb{R}^{L\times n}$, the standard attention is calculated as

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^\top}{\sqrt{n}}\right) V \in \mathbb{R}^{L\times n}. \tag{2.1}$$

The rows of Q, K, and V are respectively referred to as query, key, and value vectors. Here, the softmax function [15] is applied row-wise, normalizing the similarity scores between each query and key vectors into probability distributions, which serve to weigh the importance of each value vector.

Low-Rank adaptation (LoRA) LoRA [57] of large language models has become the most widely used parameter-efficient fine-tuning scheme for LLMs. Given a weight matrix $W \in \mathbb{R}^{n\times n}$ of a linear layer, LoRA updates W with an update $\Delta W = AB$, where $r \ll n$, $A \in \mathbb{R}^{n\times r}$, and $B \in \mathbb{R}^{r\times n}$, so that the linear layer's action becomes $X \mapsto X(W + AB)$ for an input $X \in \mathbb{R}^{L\times n}$. In this LoRA fine-tuning, the pre-trained weight W is frozen while only A and B are trained.

Next token prediction of language models. Let us describe the autoregressive next token prediction of decoder-only language models such as GPT [95, 96, 109, 91] and Llama series [110, 111, 37]. Let $\mathcal{M}$ be a decoder-only language model with L layers, $\mathcal{V}$ denote the vocabulary of $\mathcal{M}$ and d represent the embedding dimension of $\mathcal{M}$. Given a tokenized input $x = [x_0, \ldots, x_{t-1}] \in \mathbb{R}^{t \times d}$, if the output of the last hidden layer of $\mathcal{M}$ is $h_L \in \mathbb{R}^{t \times d}$, then the probability $P(x_t = v \,|\, x_0, \ldots, x_{t-1})$ of a token $v \in \mathcal{V}$ being selected as a next token x_t is calculated as follows:

$$\begin{aligned} \mathrm{z}_{\mathrm{t}-1} = (z_0, \ldots, z_{|\mathcal{V}|-1}) &\coloneqq h_L[t-1, :]E^\top + b \\ P(x_t = v \,|\, x_0, \ldots, x_{t-1}) = \mathrm{Softmax}(z_{t-1}) &= \frac{\exp(z_v/T)}{\sum_{i=0}^{|\mathcal{V}|-1} \exp(z_i/T)} \end{aligned}$$

where $E \in \mathbb{R}^{|\mathcal{V}| \times d}$ is the weight matrix of the embedding vector, $b \in \mathbb{R}^{1 \times |\mathcal{V}|}$ is a bias and T is a temperature.

There are deterministic [50, 108, 8] and random sampling algorithms. For a deterministic algorithm, the token with highest probability is selected:

$$x_t = \underset{v \in \mathcal{V}}{\mathrm{argmax}}\, P(v \,|\, x_0, \ldots, x_{t-1}).$$

For random sampling, we usually use two methods: top-k [43] and top-p sampling [55]. For top-k sampling, the top k tokens $v_1, \ldots v_k \in \mathcal{V}$ with highest probabilities are selected, and each token is selected as the next token with a probability proportional to its likelihood of being chosen. The top-p sampling is similar to the top-k sampling except that the tokens with the highest probabilities whose cumulative sum does not exceed a threshold p are selected.

2.2 CKKS Functionalites

The CKKS [22] is a homomorphic encryption scheme that allows one to encrypt real or complex data as a message polynomial and perform approximate arithmetic on the encrypted message. More precisely, CKKS scheme supports a single-instruction-multiple-data (SIMD) property from encoding (Ecd)/decoding (Dcd) map: given a power-of-two positive integer ring degree N, encoding map is defined as $\mathsf{Ecd} : \mathbb{C}^{N/2} \to \mathcal{R}_Q = \mathbb{Z}_Q[x]/(x^N + 1)$, where Q is a positive integer. The decoding map Dcd can be interpreted as an inverse process of Ecd. $\mathsf{Ecd}/\mathsf{Dcd}$ represents a conversion between $(\mathbb{C}^{N/2}, \oplus, \odot)$ and $(\mathcal{R}_Q, +, \cdot)$, where $\oplus$ and $\odot$ are component-wise addition and multiplication, respectively. In the CKKS scheme, each of Ecd and

Dcd is processed before encryption or after decryption, respectively. Encryption ($\mathsf{Enc}_{\mathsf{pk}}$) and Decryption ($\mathsf{Dec}_{\mathsf{sk}}$) are a ring learning with errors (RLWE) [107, 83] encryption and decryption, respectively. We can operate addition/multiplication on those vectors via polynomial operations from the ring isomorphism. Here, we use $\approx$ notation to indicate approximate equality, as the CKKS scheme evaluates approximately (for more details, refer to [22]);

- **Key generation**: Given a security parameter λ and a $S \subset \{1, 2, \cdots, N/2\}$, return a public key pk, a secret key sk, a relinearization key rlk, and rotation keys $\{\mathsf{rk}_i\}_{i \in S}$.
- **Encryption**: Given pk and a message $\mathbf{m}$, return a ciphertext $\mathrm{ct} = \mathsf{Enc}_{\mathsf{pk}}(\mathbf{m}) = \mathsf{Enc}(\mathsf{Ecd}(\mathbf{m}), \mathsf{pk})$.
- **Decryption**: Given sk and a ciphertext ct, return a message $\mathbf{m}' = \mathsf{Dec}_{\mathsf{sk}}(\mathrm{ct}) = \mathsf{Dcd}(\mathsf{Dec}(\mathrm{ct}, \mathsf{sk}))$, where $\mathbf{m}' \approx \mathbf{m}$.
- **Addition**: Given two ciphertexts ct_1 and ct_2, return $\mathrm{ct}_{add} = \mathsf{Add}(\mathrm{ct}_1, \mathrm{ct}_2)$, satisfying $\mathsf{Dec}_{\mathsf{sk}}(\mathrm{ct}_{add}) \approx \mathsf{Dec}_{\mathsf{sk}}(\mathrm{ct}_1) \oplus \mathsf{Dec}_{\mathsf{sk}}(\mathrm{ct}_2)$. Addition also can be evaluated between plaintext and ciphertext.
- **Multiplication**: Given two ciphertexts ct_1 and ct_2 and a linearization key rlk, return $\mathrm{ct}_{mult} = \mathsf{Mult}(\mathrm{ct}_1, \mathrm{ct}_2)$, satisfying $\mathsf{Dec}_{\mathsf{sk}}(\mathrm{ct}_{mult}) \approx \mathsf{Dec}_{\mathsf{sk}}(\mathrm{ct}_1) \odot \mathsf{Dec}_{\mathsf{sk}}(\mathrm{ct}_2)$.
- **Multiplication by Plaintext**: Given a plaintext pt and a ciphertext ct, return $\mathrm{ct}_{pmult} = \mathsf{pMult}(\mathrm{pt}, \mathrm{ct})$, satisfying $\mathsf{Dec}_{\mathsf{sk}}(\mathrm{ct}_{pmult}) \approx \mathsf{Dcd}(\mathrm{pt}) \odot \mathsf{Dec}_{\mathsf{sk}}(\mathrm{ct}_{pmult})$.
- **Rotation**: Given a rotation key rk_i for $i \in S$ and a ciphertext ct with $\mathsf{Dec}_{\mathsf{sk}}(\mathrm{ct}) \approx (m_0, m_1, \ldots, m_{N/2-1})$, return a ciphertext $\mathrm{ct}_{rot_i} = \mathsf{Rot}(\mathrm{ct}, i)$ satisfying $\mathsf{Dec}_{\mathsf{sk}}(\mathrm{ct}_{rot_i}) \approx (m_i, \ldots, m_{N/2-1}, m_0, \ldots, m_{i-1})$. The rotation index i acts modulo $N/2$, so $\mathsf{Rot}(\cdot, -i) = \mathsf{Rot}(\cdot, N/2 - i)$. pRot is the same operation on the plaintext side.

HE parameter and its time measurement. During HE implementation, we use the FGb parameter, one of the HEaaN-providing parameters. For the detailed value of the FGb parameter, refer to Table 2.1. We also list the time measurement of HE operations under a single GPU. We know that Mult and Rot, operated in ciphertext, are slower than pMult and pRot, which are related to plaintext operations, respectively.

Table 2.1: The FGb parameter and time measurement for each HE operation under the single GPU. The terms $\log(QP), N, L, \lambda$ denote the bit lengths of the largest modulus, ring degree, multiplicative depth, and security parameter.

FGb parameter					Time measurement of each homomorphic operation (ms)						
$\log(QP)$	N	L	h	λ	Add	pRot	Rot	BTS	ExtBTS	pMult	Mult
1555	2^{16}	9	192	128	0.02	0.02	0.49	60	137	0.1	0.6

2.3 Server Assumption

Our model is based on the *honest-but-curious* (HBC) security model. The adversary adheres to the protocol but is allowed to collect all outputs from the model. Since the client's input data is encrypted with the CKKS ciphertext, the entire security model relies on the semantic security of the underlying CKKS.

Chapter 3

Accelerating Fine-tuning and Inference of LLMs under HE

This section is mainly based on the author's previous work [99]. Large language models (LLMs) offer personalized responses based on user interactions, but this use case raises serious privacy concerns. Homomorphic encryption (HE) is a cryptographic protocol supporting arithmetic computations in encrypted states and provides a potential solution for privacy-preserving machine learning (PPML). However, the computational intensity of transformers poses challenges for applying HE to LLMs. In this chapter, we propose a modified HE-friendly transformer architecture with an emphasis on inference following personalized (private) fine-tuning. Utilizing LoRA fine-tuning and Gaussian kernels, we achieve significant computational speedups—6.94× for fine-tuning and 2.3× for inference—while maintaining performance comparable to plaintext models. Our findings provide a viable proof of concept for offering privacy-preserving LLM services in areas where data protection is crucial.

3.1 Introduction

3.1.1 Prior Work

Transformer-based language models and LoRA. Since the advent of attention [114], the transformer has become the standard of language models. There are three types of transformer-based language models. Encoder-only models, including BERT series [34, 79, 104, 67, 31, 54] output embeddings for inputs that can be used

for downstream tasks. Encoder-decoder models, which use the original transformer architecture, such as MarianMT [63], T5 [97], BART [76], mBART [80], and mT5 [120], are used for translation, summarizing, etc. Decoder-only models, including GPT series [95, 96, 109, 91] and Llama series [110, 111, 37], generate sentences to the user's query. These large language models (LLMs) follow the scaling law [64], so the scale of LLMs tends to increase more and more. Hence, these models require a huge amount of memory capacity for inference and fine-tuning. To overcome this issue, LoRA [57] is mainly used to fine-tune a pre-trained LLM. Freezing all other weights, LoRA adapters are added to important layers of the model, such as attention layers during fine-tuning. Using LoRA, one can fine-tune LLMs, updating only less than 1% of all of the parameters.

Privacy-preserving transformer under HE. Many researchers have explored privacy-preserving algorithms leveraging HE for transformer models. There are two main scenarios: interactive, which combines secure MPC with HE, and non-interactive, which relies only on HE. In interactive scenarios, encrypted computation time can be reduced through communications between parties. THE-X [18] proposed the first protocol for BERT-tiny inference, introducing HE-friendly workflows and non-polynomial evaluations via communications. Subsequent research [53, 77, 1, 36, 93] enhanced computation time and reduced communication costs.

However, interactive approaches may struggle with large-scale communication as model size increases, and they also require data owners to be online during computation. To address these, non-interactive research is being conducted on the other side. [128] introduced the first HE-friendly transformer model, replacing the original transformer with a HE-friendly structure, minimizing the approximation domain, and obtaining pre-trained weight with the changed structure. Using these weights, they performed inference with BERT non-interactively. NEXUS [123] further proposed a non-interactive BERT inference method without re-training, using polynomial approximations for non-polynomial operations. More recently, [94] proposed Powerformer, which, like our method, proposed replacing softmax in the attention with their BRP-max function for homomorphic inference.

Fine-tuning plays a crucial role in improving transformer models and providing more personalized responses. However, previous works have primarily focused on secure inference, largely avoiding the challenge of fine-tuning due to the significant computational complexity involved, especially in non-interactive settings. Only a few attempts, such as those by Lee et al. [70] and HETAL [74], have explored fine-tuning, focusing exclusively on the classification head while leaving other key

components like attention layers and feed-forward networks untouched. P3EFT [78] also addresses fine-tuning of foundation models, but it concentrates more on energy consumption during fine-tuning.

3.1.2 Contributions

We propose a homomorphic encryption (HE) friendly transformer architecture with an emphasis on inference following personalized (private) fine-tuning. We resolve two computational bottlenecks of the HE transformer model:

i) We use a simpler Gaussian kernel (GK) to replace the softmax layer, which is notoriously challenging to compute under HE.

ii) Using LoRA, we avoid large ciphertext-ciphertext matrix multiplications (CCMMs).

Experiments on an HE-encrypted BERT-style transformer demonstrate a speedup of 6.94× for fine-tuning and 2.3× for inference.

3.2 Preliminaries

3.2.1 Server-client Computation Model

In this section, we first state the server-client computation model, referred to as the *threat model* in the cryptography community.

Our server-client computation model (threat model) involves a client with private data, outsourcing fine-tuning and inference to the server, an LLM service provider. See Figure 3.1. Specifically, the client sends encrypted token-embedded data to the server, and the server fine-tunes the model, generating customized encrypted LoRA weights. Subsequently, the server performs inference using encrypted inference data, plaintext pre-trained weights, and encrypted LoRA weights. The server returns the encrypted inference results to the client. The token embedding layer weights are not encrypted and are not updated throughout fine-tuning.

Our model resides in a honest-but-curious (HBC) server. For more detail of this server assumption, refer to Section 2.3.

The user data used throughout fine-tuning and inference is cryptographically protected, even from the LLM service provider. Both the pre-trained weights of the LLM and the fine-tuned LoRA weights reside on the server and are not shared

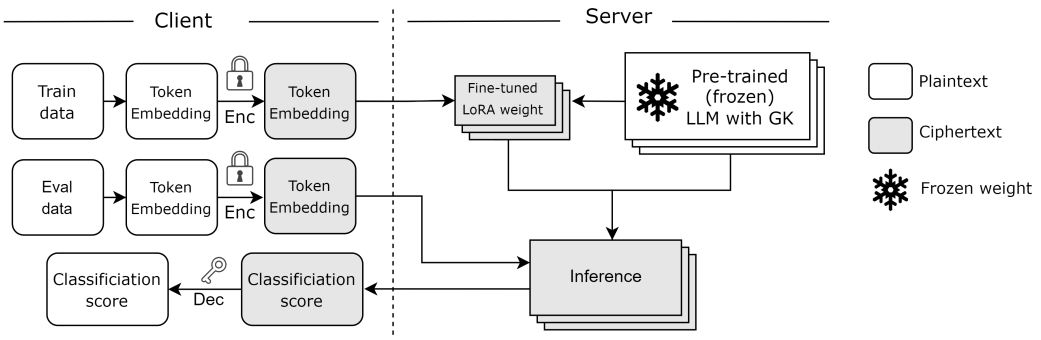


Figure 3.1: Proposed privacy-preserving LLM under homomorphic encryption (HE). HE cryptographically protects user's fine-tuning and inference data. We resolve two computational bottlenecks. First, we reduce the size of ciphertext-ciphertext matrix multiplication (CCMM) using LoRA fine-tuning. Second, we avoid the softmax computation, which is notoriously challenging to compute under HE, and replace it with a much simpler Gaussian kernel (GK).

with the user. However, we clarify that this does not mean the LLM weights are protected in the strict cryptographic sense, and our proposed approach does not address the model weight stealing problem [113, 16].

3.2.2 Matrix Multiplications: PCMM and CCMM.

While evaluating the transformer model under HE, we use two kinds of homomorphic matrix multiplications: plaintext-ciphertext matrix multiplication (PCMM) and ciphertext-ciphertext matrix multiplication (CCMM). Homomprhic matrix multiplication requires numerous HE operations, such as $(\mathsf{p})\mathsf{Mult}, (\mathsf{p})\mathsf{Rot}$ and Add, for each matrix, which are slower when operating on ciphertext compared to plaintext as shown later in Table 2.1.
As a result, *PCMM is much faster than CCMM*.

There have been several studies [59, 58, 103, 127, 7] to make homomorphic matrix multiplication more efficient. In this paper, we follow the JKLS [59] algorithm, which provides the fastest known HE computation algorithm when the matrix elements can be packed in a ciphertext. They also propose parallel homomorphic matrix multiplication. We use this parallel computation to evaluate large-size matrix multiplication, where the size is more than available ciphertext space, by repeating block-wise matrix multiplcation. Thus, the computation time gap between PCMM

and CCMM will be proportional to the number of block matrices. For more details, refer to Appendix A.1.

3.3 Speedup with LoRA: Avoiding Large CCMM

Bottleneck 1: Full fine-tuning incurs large CCMM. Personalized fine-tuning data is transmitted to the server encrypted, and the fine-tuning updates, which depend on the users' private data, must also be stored encrypted. Subsequently, encrypted inference data is transmitted, and the transformer performs CCMM computations. Full fine-tuning updates all weights and, therefore, lead to many large CCMM calculations.

Specifically, consider a pre-trained linear layer with weight matrix $W \in \mathbb{R}^{n\times n}$ stored in plaintext. If we update W to $W+\Delta W$ where $\Delta W \in \mathbb{R}^{n\times n}$, then evaluating the linear layer

$$X(W + \Delta W) = XW + X\Delta W, \qquad \text{for } X \in \mathbb{R}^{L\times n}$$

costs $\mathcal{O}(n^2)$ HE operations, where $\mathcal{O}(\cdot)$ only considers the dependence on n. However, while XW can be evaluated with PCMM, $X\Delta W$ must be evaluated with the costly CCMM. This is because W is not encrypted, but ΔW must be encrypted.

Accelerating homomorphic matrix-multiplication with LoRA. LoRA [57] fine-tuning alleviates the cost of CCMM by reducing the size of CCMMs. We clarify that the primary benefit of LoRA in the usual plaintext application is the reduced memory footprint due to having fewer optimizer memory states. Under HE, however, the main benefit is also converting large CCMMs into large PCMMs and reducing the computational cost.

Again, consider a pre-trained linear layer with weight matrix $W \in \mathbb{R}^{n\times n}$ stored in plaintext. However, consider LoRA fine-tuning updating W to $W + AB$, where $r \ll n$, $A \in \mathbb{R}^{n\times r}$, and $B \in \mathbb{R}^{r\times n}$. Then, evaluating the linear layer

$$X(W + AB) = XW + (XA)B, \qquad \text{for } X \in \mathbb{R}^{L\times n}$$

requires an $\mathcal{O}(n^2)$-cost PCMM to evaluate XW and two $\mathcal{O}(nr)$-cost CCMMs to evaluate $(XA)B$, where $\mathcal{O}(\cdot)$ only considers the dependence on n and r. See Appendix A.1 for further details. Since A and B are encrypted, evaluating $(XA)B$ still requires CCMMs, but the CCMMs are smaller than $X\Delta W$ by a factor of n/r.

The difference in computational cost is significant, as shown in Table 3.1a and Appendix 3.5.5.

Finally, we mention, but without describing the details, that an analogous consideration can be applied to backpropagation. (We remind the readers that we also perform fine-tuning under HE.)

Reducing optimizer states and inverse square root with LoRA. We fine-tune the transformer under HE using a variant of AdamW optimizer. The precise version of AdamW that we implement under HE is described in Appendix 3.5.3. For training and fine-tuning transformers, it is well known that adaptive optimizers like AdamW significantly outperform non-adaptive optimizers such as SGD [126].

However, the inverse square root function mapping is $x \mapsto 1/\sqrt{x}$. This is also a non-polynomial function that tends to be difficult to use under HE. In general, due to wide input ranges for division, BTS is required for each ciphertext containing weights during evaluation. Given the large number of parameters in the transformer model, the optimizer step can create a computation time bottleneck. In fact, the optimizer's computation time exceeds that of the transformer block evaluation in our target model under full fine-tuning, as presented in Section 3.5.2.

LoRA alleviates this computational cost by substantially reducing the number of parameters being fine-tuned, thereby reducing the number of inverse square root operations needed in the AdamW optimizer. In our target 2-layer BERT model, full fine-tuning requires 368 ciphertexts containing weights, whereas LoRA only needs 15, offering a clear advantage in optimizer computation time.

3.4 Speedup With Gaussian Kernel: Poly-apx-friendly Design

Bottleneck 2: Softmax is hard to evaluate under HE. Since HE only supports polynomial operations, non-polynomial functions must be approximated as polynomials. Traditionally, the softmax function is evaluated with

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

Subtracting α is crucial to avoid taking the exponential of a large positive number, which would lead to numerical instabilities. Under HE, however, the exponential, division, and max functions are non-polynomial functions that must be approxi-

mated. To clarify, the max function is conventionally evaluated using comparisons and if-statements, but these are non-polynomial functions that require polynomial approximations under HE.

While there is some recent work on efficiently approximating softmax function as polynomials [5, 60, 56, 73, 74], softmax is still considered numerically challenging to evaluate under HE. In this section, we resolve this issue by avoiding the softmax function altogether and introducing the Gaussian kernel (GK) as an alternative.

GK attention. The standard attention layer (2.1) obtains the attention scores using the softmax function, which is difficult to approximate under HE. We can view the softmax as normalizing the outputs of a scaled exponential kernel, where "kernel" is used in the sense of RKHS kernel methods of classical machine learning [10]. However, fundamentally, there is no requirement to use the exponential kernel, nor is there a necessity for normalizing the scores.

We propose the alternative *Gaussian kernel attention*:

$$\begin{aligned} \text{GK-Attention}(Q,K,V) &= S(Q,K)V \\ S(Q,K)_{ij} &= \exp\Big(-\frac{1}{2\sqrt{n}}\,\|Q_{i,:}-K_{j,:}\|_2^2\Big),\ i,j=1,\dots,L, \end{aligned} \tag{3.1}$$

where $\|\cdot\|_2$ is the L_2 norm, n is the hidden dimension, and $Q_{i,:}$ and $K_{j,:}$ mean the ith and jth row of Q and K. Compared to the standard attention, the scaled exponential kernel is replaced with a Gaussian kernel and we do not perform normalization. The Gaussian kernel is much easier to evaluate than the softmax function for the following reasons. First, there is no need for division. (The factor $1/(2\sqrt{n})$ is fixed, so the reciprocal can be pre-computed and then multiplied.) Second, there is no need to compute the max function, which is difficult to approximate under HE. Third, $\exp(x)$ only needs to be approximated for $x \le 0$, the region in which $\exp(x)$ does not blow up and therefore is much easier to approximate numerically. Specifically, we use

$$\exp(x) \approx p_k(x) \coloneqq \Big(1+\frac{x}{2^k}\Big)^{2^k} \quad \text{for } k \in \mathbb{N},$$

which is very accurate for the range $\big[-2^k, 0\big]$. See Appendix A.4 for further discussions on the polynomial approximation of $\exp(x)$.

Finally, we point out that in contexts unrelated to encryption or privacy, the prior work of Richter and Wattenhofer [101] made the observation that normaliza-

tions (divisions) are not necessary for the attention layers and that the prior work of Lu et al. [82] and Chen et al. [20] used Gaussian kernel in place of the softmax. Specifically, Richter and Wattenhofer [101] removed normalizations to improve the theoretical and practical characteristics of transformers, and Lu et al. [82] and Chen et al. [20] further used a low-rank approximation to the Gaussian kernel to present a linear-complexity attention layer accommodating long context lengths.

3.5 Experimental Results

3.5.1 Diagram: Full Fine-tuning vs. LoRA Fine-tuning

Figure 3.2, 3.3 illustrate the complete diagram of our target model with respect to full fine-tuning and LoRA fine-tuning, respectively. Orange boxes indicate where updates occur during fine-tuning, and c in the classification head layer denotes the number of classes. In full fine-tuning, large-size weight matrices are updated and transformed from plaintexts to ciphertexts, which means the large-size CCMMs must be processed during both fine-tuning and inference. In contrast, only small-size weight matrices are transformed into ciphertexts in LoRA fine-tuning.

3.5.2 Experimental Setup

In this subsection, we quickly describe some core aspects of the experimental setup. Some details, such as the penalty training trick and hyperparameters, are deferred to Appendix A.3.

Implementation of homomorphic encryption (HE). Our implementation is based on the C++ HEaaN library [32]. All of our experiments used 8 $\times$ Nvidia GeForce RTX 4090 24GB GPUs. We use the FGb parameter of the HEaaN library with the specific values in Table 2.1 (Appendix 2.2). Overall, we can do $L = 9$ (p)Mults before BTS, which means the minimum required level for BTS is 3. The input range of BTS is $[-1, 1]$. If the input range exceeds it, we need another BTS algorithm; ExtBTS is an extended bootstrapping for having a wide input range $[-2^{20}, 2^{20}]$, which require at least 4 levels to operate. We usually use ExtBTS in our HE model except when the input range is guaranteed. For the detailed parameter information and time measurement of HE operations, refer to Table 2.1 in Appendix 2.2. Note that Mult is $6\times$ slower than pMult and Rot is $24\times$ slower than pRot. These are the main reasons for CCMM is much slower than PCMM.

Architecture. We use an encoder-only transformer in the style of BERT [34] and largely follow the setup of [46]. We remove all bias terms in the linear layers of transformer blocks and use the same tokenization as in [46]. The hidden dimensions of embedding, attention layers, and FFNs are 768. The number of attention heads is 12. In FFNs, we enlarge the hidden dimension into 3072 with a dense layer and reduce that to 768 using a gated linear unit (GLU) [33] and ReLU. In contrast to [46], we modify the dense layer of the shape $(768, 1024)$ in the classification head attached to the BERT into the two dense blocks of the shapes $(768, 32)$ and $(32, 1024)$, which allows us to use more optimized CCMMs (see Appendix A.2.2). We set the number of transformer layers as 2 for the practical computation time. We apply LoRA only to the query, value, and key layers as applying LoRA to other layers (e.g., FFN) did not give a noticeable performance gain in our experiments. LoRA rank is 2 for all LoRA layers.

Non-polynomial approximation. Our transformer implementation under HE requires the polynomial approximation of several non-polynomial functions. We use the following methods for the approximation: Remez approximation [98], Newton's method [90], iterative method, and composition of minimax functions. Remez algorithm is computed using the Sollya tool [25]. For inverse square root functions, we combine Remez approximation with a few Newton's steps. For each non-polynomial approximation method and its precision based on input range and degree, refer to Table A.8 in Appendix A.4. We also classify polynomials used for each downstream task through the HE experiment in the same section.

3.5.3 Time Measurments

We now present the speedups provided by our proposed methods. We use an input token length of 128. For further details on the optimizations of homomorphic matrix multiplication and LoRA-friendly CCMM, refer to Appendix A.2.1 and A.2.2.

CCMM vs. PCMM. As explained in Appendix A.1 and shown by the time differences between ciphertext and plaintext operations in Table 2.1 (Appendix A.3), CCMM is approximately $5\times$ slower than PCMM for the same matrix sizes evaluation (Table 3.1a).

Table 3.1: Computation time comparison for homomorphic matrix multiplication and Softmax with GK. For matrix multiplication, we assume the evaluation of matrices for 128×768 by 768×768 where 128×768 is in ciphertext, and 768×768 is either in ciphertext or plaintext. PCMM and GK are much faster than their respective comparison group.

(a) CCMM vs. PCMM.

	Time (s)	Factor
CCMM	1.259	1
PCMM	0.275	**4.58**

(b) Softmax vs. GK

	Time (s)	Factor
Softmax	8.99	1
GK	1.44	**6.24**

Softmax vs. GK. In Table 3.1b, we compare the computation time of Softmax and GK under HE, where the input range is $[-2^{10}, 0]$ and we evaluate 12 pairs of 128×128 matrices row-wise for softmax and on matrices of the same size for GK. For the softmax function, we utilize the method of HETAL [74], which approximates the maximum function, the exponential function, and the inverse function homomorphically. GK is much faster than the softmax function evaluation using the HETAL [74] approach under the same experimental setting.

Optimizer. In the experiments, we use AdamW-HE, a modified version of AdamW [81] is stated in the following as Algorithm 1.

Algorithm 1 AdamW-HE

1: **Initialize:** $m_0 \leftarrow 0, v_0 \leftarrow 0$
2: **procedure** AdamW-HE$(\gamma, \beta_1, \beta_2, \varepsilon > 0, \lambda, \theta_{t-1}, m_{t-1}, v_{t-1}, g_t)$
3: $\quad \theta_t \leftarrow \theta_{t-1} - \gamma\lambda\theta_{t-1}$
4: $\quad m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1)\, g_t$
5: $\quad v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2)\, g_t^2$
6: $\quad \widehat{m_t} \leftarrow m_t / \left(1-\beta_1^t\right)$
7: $\quad \widehat{v_t} \leftarrow v_t / \left(1-\beta_2^t\right)$
8: $\quad \theta_t \leftarrow \theta_t - \gamma\widehat{m_t}/\sqrt{\widehat{v_t}+\varepsilon}$ $\quad$ $\left(\text{In AdamW, } \theta_t \leftarrow \theta_t - \gamma\widehat{m_t}/\left(\sqrt{\widehat{v_t}}+\varepsilon\right)\right)$
9: $\quad$ **Return** θ_t
10: **end procedure**

In the original version of AdamW, the Step 8 has the form $\theta_t \leftarrow \theta_t - \gamma\widehat{m_t}/(\sqrt{\widehat{v_t}}+\varepsilon)$, and this is numerically challenging to evaluate for the following two reasons:

- To calculate $1/\left(\sqrt{\widehat{v_t}}+\varepsilon\right)$, we have to approximate $\sqrt{x}$ and $1/x$, which are

delicate.

- Conventionally, $\varepsilon > 0$ is set to a sufficiently small value, such as 10^{-12}. However, approximating $1/x$ on the large range $[\varepsilon, 1]$ is challenging if ε is small.

Therefore, we change Step 8 in the Algorithm 1 into

$$\theta_t \leftarrow \theta_t - \gamma \widehat{m_t} / \sqrt{\widehat{v_t} + \varepsilon},$$

and we choose a value of $\varepsilon > 0$ that is not too small by considering both the performance and the approximation accuracy. Also, we use the maximum-dividing trick as in (A.1). These modifications allow AdamW-HE to be run stably under HE. Refer to Table A.6 in Appendix A.3 for hyperparameters used in AdamW-HE for each task.

Computation time for our model and downstream tasks. Table 3.2 presents the execution times of each setup. We observe a speedup of 6.94× for fine-tuning and 2.3× for inference. Table 3.3 compares the execution time for each downstream task with the Full+SM model. Our model consistently achieves a 4× improvement across each downstream task. For the detailed end-to-end time measurement for each block operation of our target model, see Table A.7 in Appendix A.3.

Table 3.2: Comparison of fine-tuning and inference times for full fine-tuning with Softmax (Full+SM), LoRA fine-tuning with Softmax (LoRA+SM), and LoRA fine-tuning with GK (LoRA+GK, Ours) approaches under a single GPU. Our model achieves speedup in both fine-tuning and inference.

	Fine-tuning			Inference		
Time (s)	Full+SM	LoRA+SM	LoRA+GK (Ours)	Full+SM	LoRA+SM	LoRA+GK (Ours)
2 layers	169.2	65.16	49.22	61.12	41.72	25.78
Class. head	1.52	1.5	1.5	0.72	0.72	0.72
Optimizer	252.83	10.31	10.31	-	-	-
Total	423.55	76.97	61.03	61.84	42.44	26.5
Factor	1	5.5	**6.94**	1	1.46	**2.33**

Table 3.3: Execution time analysis of fine-tuning for each downstream task, comparing our method with Full+SM. Our model shows the consistent 4× improvement for each downstream task.

	Task	RTE	MRPC	STS-B	COLA	SST-2	QNLI
Time per epoch (h)	Ours	4.8	7.1	11.1	16.5	130	202
	Full+SM	19.5	28.8	45.1	67.1	529	822

3.5.4 Downstream Task Performance on Encrypted Language Models

We evaluate our model on the GLUE benchmark [116]. However, we exclude WMNI [34] as in Geiping and Goldstein [46], and MNLI [118] and QQP [116] due to their size and our computational constraints. We fine-tune using the cross-entropy loss for tasks including CoLA [117], MRPC [35], RTE [48], QNLI [116], and SST-2 [105], and MSE loss for STSB [17]. We fine-tune SST-2 and QNLI only for 1 epoch and 5 epochs for others since the former are more computationally heavy. One can see the scores for Full+SM, LoRA+GK under plaintext, and LoRA+GK under HE (ours) in Table 3.4. We set $\varepsilon = 10^{-12}$ in the optimizer for all plaintext settings. For plaintext experiments, we repeat 10 times for each task and choose the best score. For experiments under HE, different ε values are chosen for each task. The results in Table 3.4 indicate that using LoRA, Gaussian kernel, and homomorphic encryption results in only a minimal reduction in model accuracy compared to using full fine-tuning, softmax, and no encryption.

Table 3.4: GLUE results on our homomorphically encrypted language model. "Matthews corr." means Matthews correlation. "Full+GK" denotes full fine-tuning with GK. For all metrics, higher values mean better performance. Our model performs comparably to plaintext models.

Task	Plaintext Fine-tuning			LoRA+GK under HE (Ours)	
	Full+SM	Full+GK	LoRA+GK	Eval under Plaintext	Eval under HE
CoLA (Matthews corr. ↑)	0.2688	0.2640	0.1883	0.1512	0.1575
MRPC (F1 ↑)	0.8304	0.8431	0.8258	0.8147	0.8147
RTE (Accuracy ↑)	0.5884	0.6101	0.5776	0.5957	0.5993
STSB (Pearson ↑)	0.8164	0.8215	0.8107	0.8002	0.7997
SST-2 (Accuracy ↑)	0.8991	0.8911	0.8567	0.8188	0.8188
QNLI (Accuracy ↑)	0.8375	0.8287	0.8040	0.7827	0.7827
Average	0.7068	0.7098	0.6772	0.6606	0.6621

3.5.5 Computational Advantage of LoRA and Gaussian Kernel for Larger Models

In section 3.5.3, we saw that CCMM is approximately 4.58 × slower than PCMM when performing matrix multiplication with dimensions 128×768 and 768×768, where 768 is the embedding size. Also, calculating attention scores using softmax is approximately 6.24 × slower than using GK. However, when the embedding dimension and sequence length become larger, as in GPT-4 [91], the gain in computational time by LoRA and GK is amplified. In this section, we calculate the time required to perform PCMM and CCMM and to obtain attention score through softmax and GK as the embedding dimension and the sequence length increase. See the Table 3.5 for the results.

Table 3.5: Computation times for performing PCMM and CCMM with $128 \times n$ by $n \times n$ matrices, and evaluating Softmax and GK on a n-dimensional input. Factor in each table represents the ratio of computation times between two operations. As the dimension n increases, the performance gains become larger.

(a) Comp. time for PCMM vs. CCMM.

Dim. (n)	256	512	768
PCMM (s)	0.138	0.139	0.275
CCMM (s)	0.297	0.463	1.259
Factor	2.15	3.33	4.58

(b) Comp. time for GK vs. Softmax.

Dim. (n)	128	256	512
GK (s)	0.17	0.34	1.36
Softmax (s)	1.3	2.86	13.04
Factor	7.65	8.41	9.59

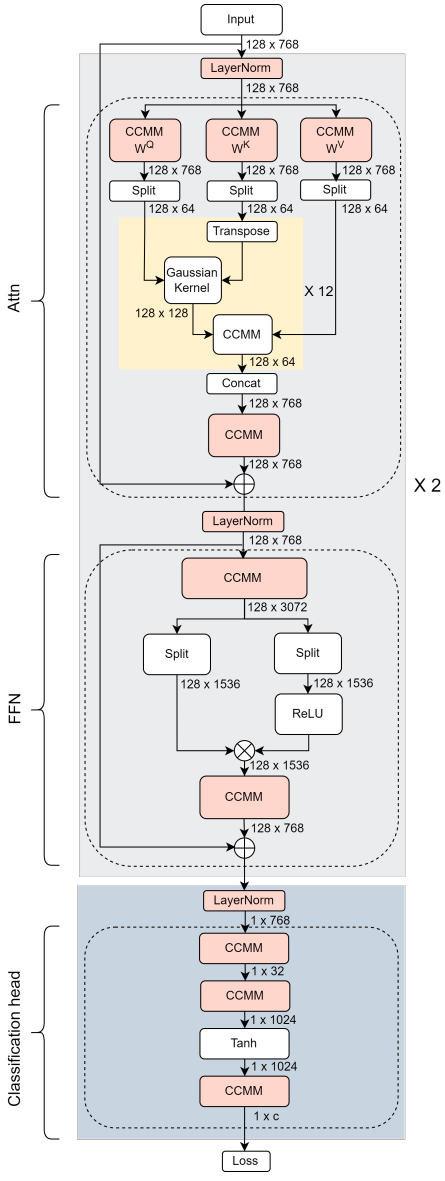


Figure 3.2: Full fine-tuning.

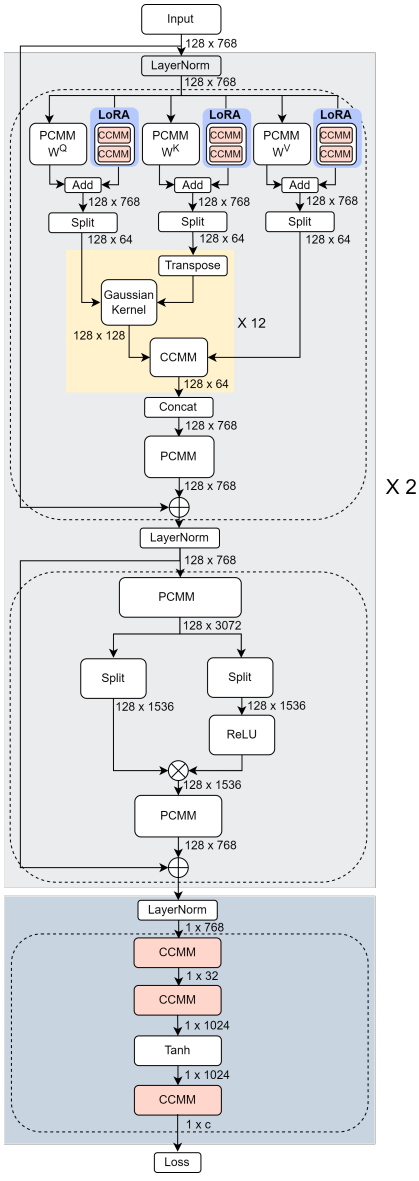


Figure 3.3: LoRA fine-tuning.

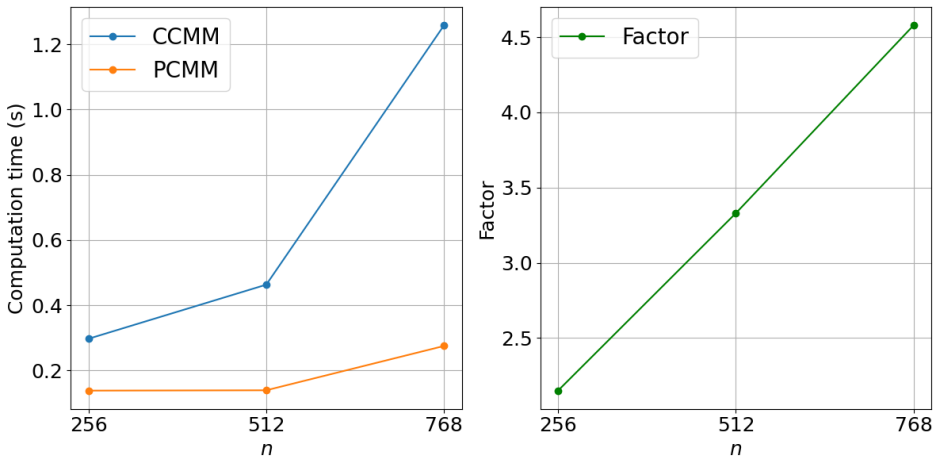


Figure 3.4: Computation time comparison between PCMM with CCMM for $128 \times n$ matrix by $n \times n$ matrix multiplications according to input dimension n. As the input dimension increases, CCMM becomes progressively slower compared to PCMM.

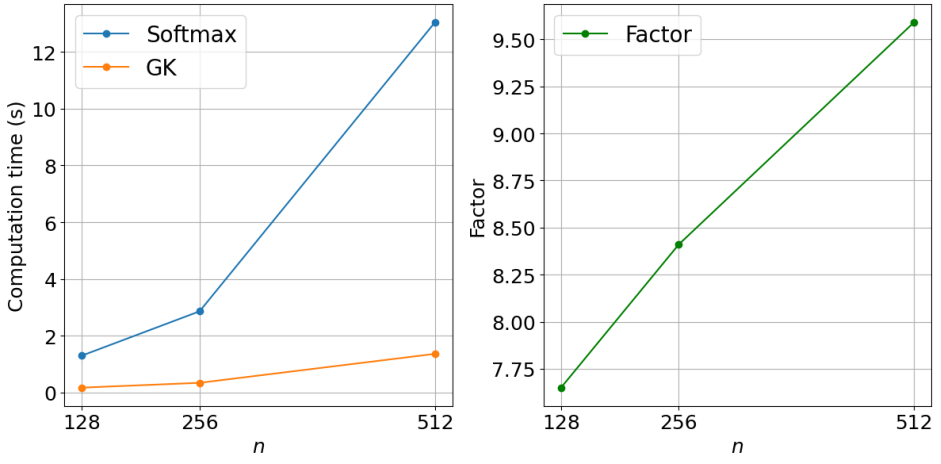


Figure 3.5: Computation time comparison between Softmax and GK according to input dimension n. As the input dimension increases, Softmax becomes progressively slower compared to GK.

Chapter 4

Stable Text Generation under HE

This section is mainly based on the author's previous work [100]. As users increasingly interact with large language models (LLMs) using private information, secure and encrypted communication becomes essential. Homomorphic encryption (HE) provides a principled solution by enabling computation directly on encrypted data. Although prior work has explored aspects of running LLMs under HE, the challenge of text generation, particularly next-token prediction, has received limited attention and remains a key obstacle to practical encrypted interaction. In this work, we propose a TSP-based token reordering strategy to address the difficulties of encrypted text generation, together with a post-processing step that further reduces approximation error. Theoretical analysis and experimental results demonstrate that our method prevents collapse, improves coherence in generated text, and preserves data privacy throughout. Overall, our contributions advance the feasibility of practical and privacy-preserving LLM inference.

4.1 Introduction

4.1.1 Prior work

Language models under FHE. A variety of approaches have been proposed to address the main challenges: non-linear operations, matrix multiplication, and fine-tuning. For non-linear operations, [68] approximates the sign function with minimax approximation, while [71] approximates ReLU and max-pooling in convo-

lutional neural networks under FHE. [99] replaces softmax with a Gaussian kernel to eliminate division and max operations, and [29] introduces a normalize-and-square method for accurate softmax approximation over a large range. [74] and [88] present algorithms for faster matrix multiplication. Finally, for fine-tuning, [99] shows that LoRA reduces ciphertext–ciphertext multiplications. Beyond this line of work, we focus on random sampling for next-token prediction, contributing to the practical implementation of LLMs under FHE.

Next-token prediction methods of language models. Language models typically generate text via next-token prediction, and a variety of decoding algorithms have been explored both prior to and following the advent of language models. The most basic approaches include greedy decoding and probabilistic sampling [49]. To generate plausible text, more sophisticated strategies have been developed, such as beam search and threshold-based methods [19, 86, 89, 115]. Among these, top-k [43] and top-p sampling [55] have become the most widely adopted methods, as they produce more natural text. However, they rely on conditional operations such as if-statements, which are inefficient to implement under CKKS.

Secure sampling methods under FHE. There are several papers for secure sampling. First, [30] address secure sampling in multi-party computation (MPC) [122] settings under FHE, where parties such as server and client communicate for computation in encrypted states. However, MPC incurs substantial communication overhead, while our non-interactive setting avoids this cost. Besides these, there are a few works implementing sampling and argmax under FHE. For example, [45] performs secure sampling under TFHE [28] and [62, 124, 125] implement argmax under CKKS [21, 23]. Concurrent to our work, [3] proposes CKKS-compatible argmax and nucleus sampling methods. While they focus on the next-token selection step itself, our work emphasizes the impact of embedding error accumulation and provides quantitative analysis of text quality.

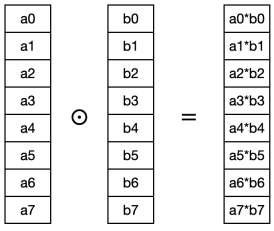
4.1.2 Contributions

In this work, we propose an efficient algorithm for next-token prediction under CKKS, along with a method that addresses the limitations of inexact one-hot vector representations. Our algorithm avoids max operations and sorting, relying solely on SIMD-friendly operations to enable practical homomorphic implementation. We show that the resulting errors are bounded by a small constant, ensuring

controlled accuracy. However, even when errors are reduced to a negligible level, subsequent matrix multiplications may still yield semantically unrelated embedding combinations that affect model stability. To mitigate this, we adapt the traveling salesman problem (TSP) to reorder embedding vectors so that semantically similar vectors are placed adjacently, thereby preserving model stability in the presence of encryption noise.

4.2 Preliminaries

SIMD capabilities in FHE system. Fully Homomorphic Encryption (FHE) is a foundational cryptographic primitive that enables computations on encrypted data without requiring decryption. FHE supports various operations, including addition, multiplication, and rotation. Both addition and multiplication can be applied to scalar and non-scalar values, and are executed in a slot-wise manner, while the rotation operation performs a cyclic shift on the data elements within the slots. Notably, the FHE system packs N data elements into a single ciphertext, allowing computations to be performed on this packed data. Consequently, a single ciphertext operation can process N data elements in parallel, enabling efficient SIMD (Single Instruction, Multiple Data) operations. The following example demonstrates a single ciphertext multiplication leading to the simultaneous computation of the products of eight data elements. This product is the Hadamard product, denoted as $\odot$.



FHE security constraints on the use of divide-and-conquer. The number of data elements packed into a polynomial, denoted as N, cannot be arbitrarily changed because N plays a critical role in the security of FHE schemes, particularly those based on the Learning With Errors (LWE) problem [27, 39] and the Ring Learning With Errors (RLWE) problem [12, 14, 21, 23, 44]. Reducing N results in a smaller modulus Q, which makes the underlying lattice problem easier to solve and increases susceptibility to attacks, such as those using the LLL (Lenstra-Lenstra-

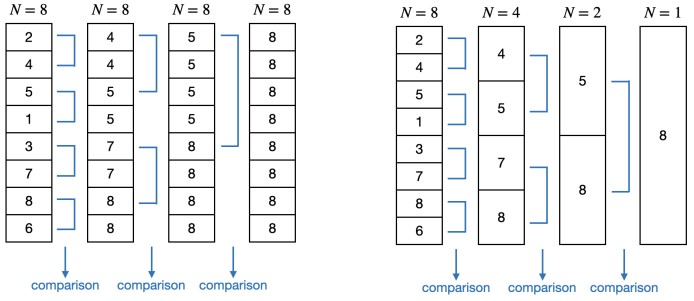


Figure 4.1: (**Left**) The process of calculating the maximum value from $N = 8$ inputs using a full sequence of secure comparisons in FHE. Each step compares adjacent pairs, halving the number of candidates until the maximum is found. (**Right**) An alternative strategy that reduces N before applying secure comparison. While this approach reduces computational cost, it compromises FHE security by exposing intermediate results, making it infeasible in secure settings.

Lovász) [75] algorithm. Therefore, the divide-and-conquer method, which reduces the number of slots N to improve efficiency, as shown on the right side of Figure 4.1, cannot be applied in FHE. While this method would enhance the efficiency of comparison operations, reducing N compromises the security of the FHE scheme and the integrity of the system.

Inverse transform sampling. Inverse transform sampling (ITS) is a standard technique that generates samples from a probability distribution using its cumulative distribution function (CDF) and uniform sampling. Let X be a discrete real-valued random variable with support $\{x_0, \ldots, x_{|\mathcal{V}|-1}\}$ with $p_k = P(X = x_k)$ and define $s_k = \sum_{i=0}^{j} p_i$ $(k = 0, \ldots, |\mathcal{V}| - 1)$ with $s_{-1} = 0$. The procedure of the ITS for X is as follows: (i) Sample $U \sim \text{Uniform}([0, 1])$; (ii) find k such that $s_{k-1} \leq U < s_k$ (which is equivalent to $s_{k-1} < U \leq s_k$, the definition of inverse transform sampling); (iii) return $X = x_k$. In this work, we adapt into a CKKS-friendly variant of the ITS, as described in Algorithm 2.

Next token prediction of language models. We quickly describe the auto-regressive next-token prediction of decoder-only language models such as GPT [91, 95, 96, 109] and Llama series [37, 110, 112]. Let $\mathcal{M}$ be a decoder-only language model with L layers, $\mathcal{V}$ and d denote the vocabulary and the embedding dimension of $\mathcal{M}$. Given a tokenized input $X = [x_0, \ldots, x_{t-1}] \in \mathbb{R}^{d \times t}$, if the output of the last

hidden layer of $\mathcal{M}$ is $h_L \in \mathbb{R}^{d\times t}$, then the probability $P(x_t = v \mid x_0, \ldots, x_{t-1})$ of a token $v \in \mathcal{V}$ being selected as a next token x_t is calculated as follows:

$$Z_{t-1} \coloneqq (z_0, \ldots, z_{|\mathcal{V}|-1}) = W h_L^{(t-1)},$$
$$P(x_t = v \mid x_0, \ldots, x_{t-1}) \coloneqq \mathrm{Softmax}(Z_{t-1}) = \frac{\exp(z_v/T)}{\sum_{i=0}^{|\mathcal{V}|-1} \exp(z_i/T)}$$

where $W \in \mathbb{R}^{|\mathcal{V}|\times d}$ is the embedding weight, $h_L^{(t-1)}$ is the last column of h_L, and $T > 0$ is a temperature parameter. In practice, greedy decoding, top-k, and top-p sampling are the most widely used methods. However, since these methods rely on many comparisons that are inefficient under CKKS, we focus on the probabilistic sampling: $x_t \sim P(\cdot \mid x_0, \ldots, x_{t-1})$.

If we implement a language model using libraries, the embedding vector corresponding to the next token is retrieved as follows: if the index of the next token is i, then just get the embedding vector by directly indexing the embedding matrix, *i.e.*, the look-up table: $e_i = E[i]$. However, under HE, indexing is difficult to implement, so the embedding vector must be retrived via matrix multiplication: $e_i = \mathbf{e}_i E$ where $\mathbf{e}_i \in \mathbb{R}^{\mathcal{V}}$ is the i-th unit vector.

4.3 Reordering tokens using TSP to prevent corrupted text generation

In this section, we explain why the ordering of tokens influences generation under approximate computation, and propose to reorder the tokens using a traveling salesman problem (TSP) approach to prevent corrupted text generation.

4.3.1 With imperfect sampling, adjacent tokens affect each other

In our random sampling algorithm, which will be described in Section 4.4, we aim to construct a one-hot vector $I \in \mathbb{R}^{|\mathcal{V}|}$ corresponding to the predicted next token (see Figure 4.4). This design choice arises because, under CKKS, the predicted index remains encrypted and thus direct embedding lookup is not feasible. Instead, the input to the subsequent layer is obtained by multiplying the embedding matrix by the one-hot vector I. Formally, the embedding vector w_i for the next token i is given by $w_i = I^\top W$, where W denotes the embedding weight matrix.

However, in the homomorphic setting, sampling cannot yield an exact one-hot vector. Due to unavoidable approximation errors, the resulting vector $\tilde{I}$ is imperfect

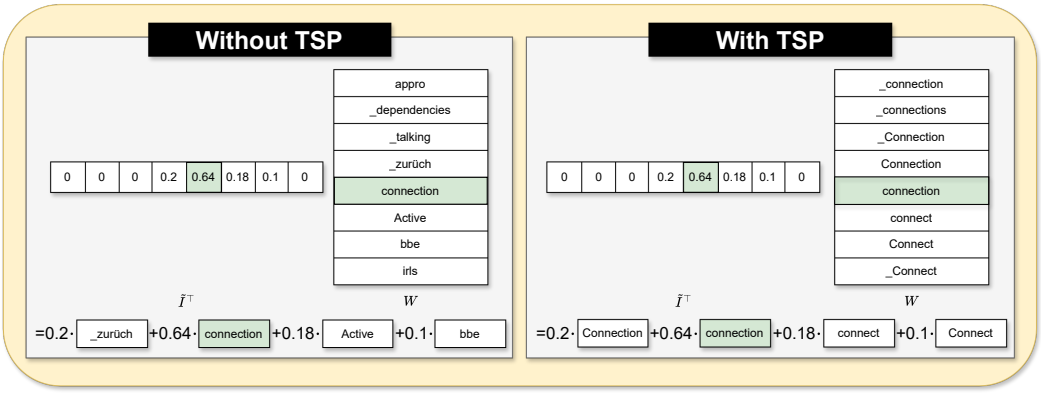


Figure 4.2: The effect of TSP. **(Left)** Without TSP, we obtain the linear combination of semantically irrelevant token embeddings. **(Right)** With TSP, semantically similar tokens can be combined.

and assigns small nonzero weights to multiple indices rather than a single one. To formalize this limitation, suppose that a probability vector p and its cumulative vector $(s_k)_{k=0}^{|\mathcal{V}|-1}$ is given as in Section 4.2. Inverse transform sampling maps a uniform random sample $r \sim U[0,1]$ into a one-hot vector $\mathbf{e}_{i(r)}$ where $s_{i(r)-1} \leq U < s_{i(r)}$. In the homomorphic setting, any algorithm can only produce an approximation $\tilde{I}$ of $\mathbf{e}_{i(r)}$, and the following theorem shows that the approximation error is necessarily nonzero.

Theorem 1. *Let $\mathbf{e}_{i(r)}$ denote the one-hot vector determined by inverse transform sampling from a random sample $r \sim U(0,1)$. For any homomorphic algorithm that computes an approximation $\tilde{I}$ of $\mathbf{e}_{i(r)}$, the following inequality holds:*

$$\mathbb{E}_r\left[\|\tilde{I} - \mathbf{e}_{i(r)}\|_\infty\right] > 0,$$

where the expectation is taken over the random sample r.

Proof. For a random variable r, $i(r)$ is a step function. Any homomorphic evaluation can be represented by a polynomial function. Let the degree of this polynomial be m.

According to the study on the uniform approximation of $\mathrm{sgn}(x)$ [41], the deviation between $\tilde{I}$ and $\mathbf{e}_{i(r)}$ satisfies

$$\|\tilde{I} - \mathbf{e}_{i(r)}\|_\infty \;\geq\; \frac{1-\delta}{\sqrt{\pi\delta}}\left(\frac{1-\delta}{1+\delta}\right)^m \frac{1}{\sqrt{m}}$$

outside the δ-neighborhood of its discontinuities, for any $\delta > 0$ and regardless of how large m is. Consequently, we have

$$\mathbb{E}_r\left[\|\tilde{I} - \mathbf{e}_{i(r)}\|_\infty\right] \;\geq\; (1-2\delta)\,\frac{1-\delta}{\sqrt{\pi\delta}}\left(\frac{1-\delta}{1+\delta}\right)^m \frac{1}{\sqrt{m}} \;>\; 0,$$

which completes the proof. □

Consequently, the computed embedding $\tilde{I}^\top W$ is not a single token embedding w_i but rather a linear combination of multiple embeddings, as illustrated in the left of Figure 4.2.

In LLMs, however, tokens at adjacent indices generally have unrelated semantics (see the left of Table 4.1). As a result, when the approximate vector $\tilde{I}$ produces a linear combination of such embeddings, the outcome is semantically meaningless. Theorem 1 further shows that this issue cannot be avoided: even with highly accurate function approximation, an exact one-hot vector is unattainable, and thus approximation alone is not a fundamental solution. Therefore, we require a different approach. Several strategies can be considered to mitigate interference between embeddings of adjacent tokens, and we propose a TSP-based reordering method that places semantically similar tokens adjacently.

Table 4.1: Adjacent tokens of `Llama-2-7b-hf` before and after applying TSP ordering. The TSP places similar tokens in adjacent indices. The symbol '_' represents a space.

Without TSP		With TSP	
Index	Token	Index	Token
⋮	⋮	⋮	⋮
9961	`appro`	18927	`_connection`
9962	`_dependencies`	18928	`_extension`
9963	`_talking`	18929	`_Connection`
9964	`_zurück`	18930	`Connection`
9965	`connection`	18931	`connection`
9966	`Active`	18932	`connect`
9967	`bbe`	18933	`Connect`
9968	`irls`	18934	`_Connect`
9969	`_Inf`	18935	`_connect`
⋮	⋮	⋮	

4.3.2 TSP order mitigates damage from imperfect sampling

As a solution to the aforementioned problem, we propose applying the TSP to reorder the tokens to mitigate the damage from imperfect one-hot vectors.

When an imperfect one-hot vector is multiplied with the embedding matrix, a weighted sum of multiple embedding vectors $\sum_{j=-m}^{n} c_j W_{i+j}$ is obtained in place of W_i where i is the selected next-token index with non-negative coefficients c_j. However, as shown in the left of Table 4.1, the surrounding tokens have semantically unrelated meanings, degrading the quality of the linear sum of obtained embedding vector. This erroneous embeddings accumulate during text generation, resulting in a collapsed text (see Table 4.2).

In contrast, if we rearrange the rows of the pretrained embedding matrix such that the embedding vectors of adjacent token indices are similar to each other, then text generation collapse can be mitigated. To resolve this, we minimize the sum of cosine distance of adjacent tokens:

$$\min_{\pi} \sum_{i=1}^{|\mathcal{V}|-1} d_{\cos}(W_{\pi(i)}, W_{\pi(i+1)}),$$

where π is a permutation and adopt a TSP as a solution. By applying the TSP, we can place semantically similar tokens adjacently, therefore mitigating the problem mentioned in the previous section, as illustrated in the right of Figure 4.2. We visualize the effect of TSP token reordering in Figure 4.3. In this figure, one can see that after reordering of tokens, the average cosine similarity between adjacent tokens substantially increases (from 0.024 to 0.271 in the case of `Llama-2-7b-hf`).

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 MMT MMSMSMSZMSMSMS MSMS MMSMSMSMSMSMS MMSMSMSMSMS MS MMSMSMSMS MMSMSMSMSMS SMS MMSMS WeMS MSMS MSMMSMMS I MOMS MS MS MJ MSMSMSMMS IMMSMSMSMSMSPA MSMSMSMMSMSMSMSMSMS MSMS MSMS MSMMSItWACMSMS MSMSMMSMMS These IlMS MSMSMSMSMSMSMSMSMSMSMSMSMS MMSMSMSMSMSMS MSMS M MMS MMSMSMSMSMSMSMSMSMSMSMSMS MS MMSMS MSMSCCMS MMS MSMSMMSMSMS MSMS MMS $MSMSMSMSNMSMSMSMSMSMSMSMSMS-MAMMSMSMSMSMSMSMSMSMS MSMSMSMMSMSMS MMSAAMS weM-CMSMSMMSMSMSMSAAMSMMSMSMSMSMSMSMSMSMSMSMSMSMSMS MSMSMSMSMS MANMSMSMSN Find MSMSMSMSMSMSMSMSMSMSMSM-SNA MSMSMSMSMSMSMSMS MSMMSMSMSMSMSMSMS STyMSMMS C IllPMSMSMSMSMSMSMSMSMSMSMS5 MSMS GMSLAMSMS MSMSMSMSMS Project MSMSMSMSMSMSMS MSMS MSMSMS MSMSPMSMM SKMSMSMSMSMSMSMMSMSMSMSMSMWMS MSMSMMSMSMSMSMSMSMSMSMSMSMSMSMS MSMS MSMSM MarMSMSMSMSWMSMSMSMSMS MSMSMSMSMSMSMSMS MSMS MSMSMSPAMSMSMSMSMS MSMS MSMSMSLMSMSMS MMSMSMSMS'MSMSMSMMSMMSMSlMSMSMMS MS MPA
Perplexity	12.6717
Score	4

Table 4.2: An example of corrupted text generated by our homomorphic inverse transform sampling. The token '`MS`' is repeated meaninglessly.

However, since the TSP is NP-hard [65], naively applying it to a large-size LLM vocabulary is prohibitively expensive. To circumvent this, we adopt a nearest neighbor heuristic [61], which heuristically constructs an approximate solution with time complexity $\mathcal{O}(|\mathcal{V}|^2)$.

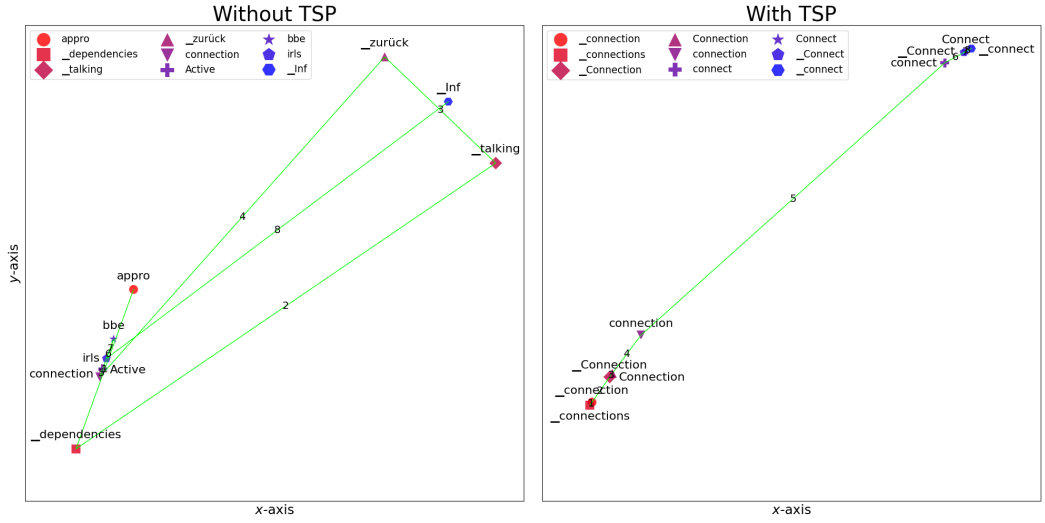


Figure 4.3: Visualization of adjacent token embeddings in a 2D plane using UMAP [85], with vectors normalized to the unit circle for cosine similarity comparison. In both figures, embedding vectors of tokens adjacent to the token '`connection`' (▼) are plotted. Each green line connects two adjacent tokens, and longer lines indicate lower similarity. In the left figure, there are four long green lines, indicating the low cosine similarity between adjacent tokens. In contrast, with TSP (right), only one long line remains, showing that adjacent tokens are now semantically similar.

4.4 CKKS-friendly and efficient SIMD-based inverse transform sampling

In this section, we present an algorithm for next-token prediction suitable for FHE using inverse transform sampling. Our algorithm starts from the softmax probability vector in next-token prediction stage, and the goal is to obtain the one-hot vector corresponding to the selected token. As explained in the previous section, methods such as argmax, top-k, and top-p cannot utilize SIMD processing or divide-and-conquer technique (see Section 4.2), making them inefficient for FHE. In contrast, our algorithm leverages efficient SIMD processing. Our algorithm is described in Algorithm 2 and Figure 4.4. $\widetilde{H}(x)$ in the algorithm refers to the approximation (B.1) of the Heaviside function $H(x) = \frac{1}{2}(\mathrm{sgn}(x) + 1)$ where sgn is the sign function.

Algorithm 2 CKKS-friendly inverse transform sampling

Input: Probability vector P.
Output: Approximate one-hot vector $\tilde{I}'$
procedure HOMOMORPHIC INVERSE TRANSFORM SAMPLING(P)
    Sample $r \sim U(0,1)$.
    Compute the cumulative distribution vector F of P.
    Apply the approximate Heaviside function: $h \leftarrow \widetilde{H}(F - r)$.
    Rotate the slots: $\mathrm{Rot}(h)$.
    Calculate $\tilde{I} \leftarrow h \odot (1 - \mathrm{Rot}(h))$.
    Apply post-processing: $\tilde{I}' \leftarrow \mathrm{PP}(\tilde{I}) := 3\tilde{I}^2 - 2\tilde{I}^3$.
    Return $\tilde{I}'$
end procedure

4.4.1 The sampling procedure

We explain Algorithm 2 and refer to Figure 4.4 for intuitive understanding. In step (i) of Figure 4.4, a softmax probability vector $P = (p_i) \in \mathbb{R}^{|\mathcal{V}|}$ where p_i denotes the probability of selecting token i is given. In step (ii) we compute the cumulative distribution vector $F = (s_i)_{i=0}^{|\mathcal{V}|-1}$ as $s_i = \sum_{j=0}^{i} p_j$ with $s_{-1} = 0$. In step (iii), sample $r \sim \mathrm{Uniform}([0,1])$ and compute $F - r$. In step (iv), apply the Heaviside function. The goal of this step is making each element of $F - r$ 0 or 1 (see the top of Figure 4.4). However, we have to approximate the non-polynomial Heaviside function, therefore the result of step (iv) is not 0 or 1 (see the bottom of Figure 4.4). In step (v)$\sim$(vii), we apply a rotation, subtraction, and element-wise multiplication. In step (viii), we apply a polynomial to enhance one-hotness of the resulting vector. Since we approximate the Heaviside function, $\tilde{I}$ is different from a one-hot vector, as shown in the bottom of Figure 4.4. Intuitively, we obtain the one-hot vector corresponding to the token at the first index where $F - r$ exceeds zero.

Step (ii) can be efficiently implemented as a single homomorphic matrix multiplication operation. This is because F can be computed by multiplying P with a lower triangular matrix filled with ones, as shown below:

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_{|\mathcal{V}|-1} \end{bmatrix} = \begin{bmatrix} p_0 \\ p_0 + p_1 \\ \vdots \\ p_0 + p_1 + \cdots + p_{|\mathcal{V}|-1} \end{bmatrix}.$$

This operation involves a plaintext matrix multiplication with an encrypted vector.

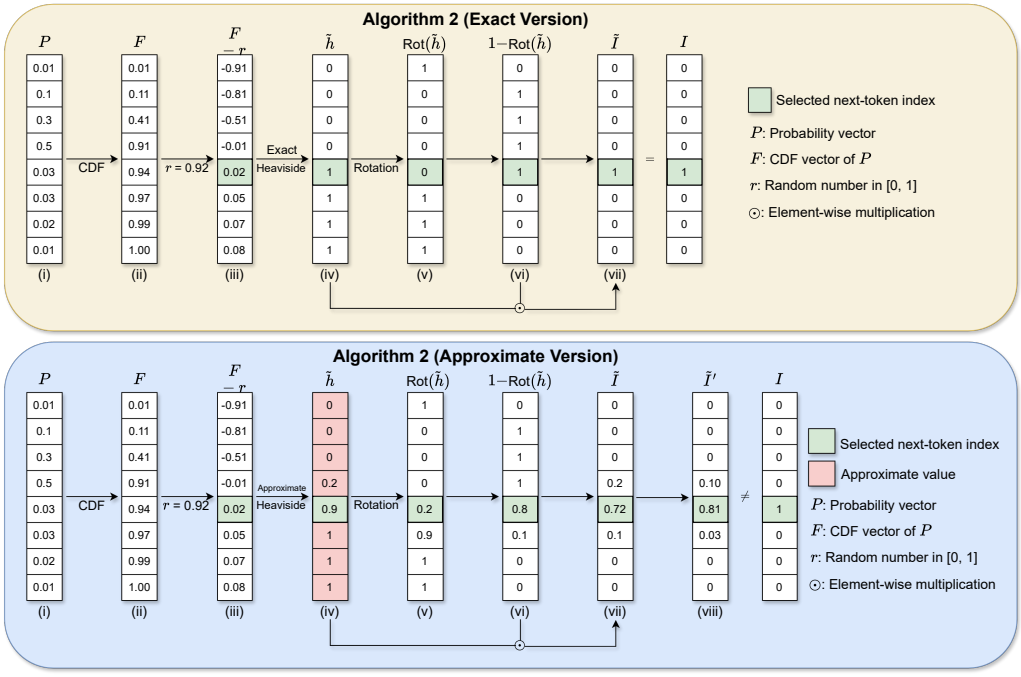


Figure 4.4: An illustrative example of Algorithm 2. **(Top)** With the exact Heaviside function, we obtain the exact one-hot vector. **(Bottom)** With an approximate Heaviside function, the resulting vector is different from a one-hot vector. Note that the steps are labeled (i)–(vii) for the exact version, with an additional step (viii) in the approximate version to enhance the one-hotness of $\tilde{I}$.

Given that the plaintext matrix is fully known, such plaintext-ciphertext operations can be performed with high efficiency under homomorphic encryption, particularly when leveraging optimized linear algebra libraries, as described in [6]. Step (iv) and (viii) is implemented efficiently, utilizing SIME processing. Also, no homomorphic operation is required to sample r. And we need only one homomorphic rotation in step (v) and subtraction and element-wise multiplication of vectors are also cheap.

Finally, we explain step (viii) in the bottom of Figure 4.4, which we call *post-processing*. As stated in the previous section, although we approximate the Heaviside function with high accuracy, sometimes the resulting vector can be significantly different from a one-hot vector as shown in Table 4.3. For example, the sum of the resulting vector can significantly exceed 1. Further, although the sum of the resulting vector is approximately 1, the dominant value can be much less than 1. These cases are problematic since they result in 'big' or 'mixed' embedding vectors. To

Table 4.3: The effects of post-processing. **(Case 1)** An example of a weight vector obtained using our algorithm where the sum of all weights exceedingly 1. After applying post-processing, the sum of all weights becomes closer to 1. **(Case 2)** An example of a weight whose sum is near 1 but the dominant value is far less than 1. After post-processing, the sum of all weights is still approximately 1, but the dominant weight increases, and the other weights significantly decrease.

		Top i-th Element								
	Post-processing	1	2	3	4	5	6	$\cdots$	512	Sum
Case 1	False	0.4575	0.4335	0.1816	0.0942	0.0863	0.0863	$\cdots$	0.0049	3.9572
	True	0.4364	0.4008	0.0870	0.0249	0.0211	0.0211	$\cdots$	$7.18e-5$	**1.0504**
Case 2	False	0.9292	0.0413	0.0227	0.0198	0.0003	0.0000	$\cdots$	0.0000	1.0133
	True	**0.9857**	0.0050	0.0015	0.0012	0.0000	0.0000	$\cdots$	0.0000	**0.9934**

remedy these, we propose a function PP defined as $\mathrm{PP}(x) = -2x^3 + 3x^2$, which is also used in [66]. This function makes an element close to 0 closer to 0 and an element close to 1 closer to 1. It is FHE-friendly since the degree of f is low, consuming low depth. Post-processing is applied in step 9 of Algorithm 2. By applying post-processing, the sum of the resulting vector of our algorithm becomes closer to a one-hot vector as in Table 4.3. See Appendix B.2 for the case analysis of the need for post-processing. Section 4.5 presents the results of the ablation study that evaluate the effectiveness of post-processing in improving text generation quality.

4.4.2 Imperfect one-hot vector generation in homomorphic inverse transform sampling

The problematic part of Algorithm 2 is step (iv). If we apply the Heaviside function in step (iv), it is straightforward to see that a one-hot vector can be obtained in step (vii). However, since we have to approximate the discontinuous Heaviside function, in step (iv), the resulting vector $\tilde{I}$ obtained in step (vii) is different from one-hot. This error implies that several tokens may be chosen simultaneously. The error from the Heaviside is particularly considerable when the input is near zero. Also, this error compounds during text generation as explained in Section 4.3.2. This motivates the use of TSP-based token reordering described in Section 4.3.2.

4.4.3 Theoretical analysis of our sampling method

In this section, we establish error bounds for the approximate one-hot vectors and embedding representations, and analyze the effects of TSP token reordering and

post-processing. To this end, we introduce the notion of a *good event* and impose two natural assumptions, motivated by the approximation of the Heaviside function and [55], respectively.

Assumption 2. The approximation $\widetilde{H}$ of the Heaviside function satisfies the following properties:

a) $0 \leq \widetilde{H}(x) \leq 1$ and $\widetilde{H}(x) = 1 - \widetilde{H}(1-x)\ \forall x \in [-1, 1]$.

b) Given $\varepsilon > 0$, $\exists\, \delta > 0$ s.t. $x \in [-1, -\delta] \Rightarrow \widetilde{H}(x) \in [0, \varepsilon]$ and $x \in [\delta, 1] \Rightarrow \widetilde{H}(x) \in [1-\varepsilon, 1]$.

The parameters ε and δ capture the accuracy–margin trade-off of $\widetilde{H}$ and both of them are very small as in figure B.2. In what follows, we fix ε together with its corresponding δ.

Definition 3. Given a probability vector p and its cumulative probability $(s_i)_{i=0}^{|\mathcal{V}|-1}$ where $s_{-1} = 0$ and $s_i = \sum_{j=0}^{i} p_j$, we define a *good event* $\mathcal{G}_p$ as follows:

$$\mathcal{G}_p = \{r \in [0,1] \,|\, \exists\, k \text{ s.t. } s_{k-1} + \delta \leq r < s_k - \delta\}$$

and a bad event as $\mathcal{G}_p^c$.

Finally, following [55], we assume that a small number of tokens account for the majority of the probability mass in next-token prediction.

Assumption 4. The output distribution of the LLM is peaked, meaning the number of tokens that capture most of the probability mass is much smaller than the total vocabulary size. That is, $k_{\text{eff}} \ll |\mathcal{V}|$ tokens together capture a probability mass of at least $1 - \varepsilon_{\text{tail}}$, for some small $\varepsilon_{\text{tail}} > 0$.

We are now ready to state our theoretical results, which provide upper bounds on the expected errors of our methods.

Theorem 5. *Let P be a given probability vector and let $r \sim \text{Uniform}([0,1])$. For the outputs $\tilde{I}$ and $\tilde{I}'$ of Algorithm 2, their deviations from the one-hot vector $\mathbf{e}_{i(r)}$ satisfy the following inequalities:*

$$\begin{aligned} \mathbb{E}_r\big[\|\tilde{I} - \mathbf{e}_{i(r)}\|_\infty\big] &\leq 2\varepsilon + 2k_{\text{eff}}\delta + \varepsilon_{\text{tail}}, \\ \mathbb{E}_r\big[\|\tilde{I}' - \mathbf{e}_{i(r)}\|_\infty\big] &\leq 12\varepsilon^2 + 2k_{\text{eff}}\delta + \varepsilon_{\text{tail}}. \end{aligned}$$

Since $12\varepsilon^2$ is much smaller than 2ε, the post-processing step effectively reduces the expected error.

Proof. According to Assumption 2, $\widetilde{H}(x)$ satisfies the error bound ε outside the δ-neighborhood of its discontinuity. Let $\mathcal{V}$ be the vocabulary with distribution $p = (p_j)_{j=0}^{|\mathcal{V}|-1}$, and define the cumulative sums $s_i = \sum_{j=0}^{i} p_j$ for $i = 0, \ldots, |\mathcal{V}| - 1$ and $s_{-1} = 0$. The inverse-transform output $\mathbf{e}_{i(r)}$ is discontinuous at each threshold $r = s_i$. For $j = i(r)$,

$$\begin{aligned}|\tilde{I}_j - 1| &= |\widetilde{H}(s_j - r)(1 - \widetilde{H}(s_{j-1} - r) - 1| = |\widetilde{H}(s_j - r)\widetilde{H}(r - s_{j-1}) - 1| \\ &= 1 - \widetilde{H}(s_j - r)\widetilde{H}(r - s_{j-1}) \leq 1 - (1 - \varepsilon)^2 = 2\varepsilon - \varepsilon^2 \\ &< 2\varepsilon.\end{aligned}$$

Also, if $j \neq i(r)$, then $s_{j-1} - r \geq \delta$ or $s_j - r \leq -\delta$. Therefore,

$$|\tilde{I}_j - 0| = \widetilde{H}(s_j - r)\widetilde{H}(r - s_{j-1}) \leq 1 \cdot \varepsilon = \varepsilon.$$

Hence we have $\|\tilde{I} - \mathbf{e}_{i(r)}\|_\infty \leq 2\varepsilon$.
From Assumption 4, the next-token distribution p is concentrated on a small head of size k_{eff} that carries at least $1 - \varepsilon_{\text{tail}}$ of the total mass. Let H denote the set of indices of the top-k_{eff} tokens by probability (the *Head*), and let $T := H^{\text{c}}$ denote the remaining indices (the *Tail*). Decomposing $\mathcal{G}_p^c$ into contributions from Head and Tail gives

$$\begin{aligned}\mathbb{P}[\mathcal{G}_p^c] &= \mathbb{P}\left[r \in \bigcup_{i=0}^{|V|-1} ([s_{i-1}, s_i] - [s_{i-1} + \delta, s_i - \delta))\right] \\ &= \mathbb{P}\Big[r \in \bigcup_{i \in H} ([s_{i-1}, s_i] - [s_{i-1} + \delta, s_i - \delta])\Big] + \mathbb{P}\Big[r \in \bigcup_{i \in T} ([s_{i-1}, s_i] - [s_{i-1} + \delta, s_i - \delta])\Big] \\ &\leq \sum_{i \in H} 2\delta + \sum_{i \in T} p_i = 2k_{\text{eff}}\delta + \varepsilon_{\text{tail}}.\end{aligned}$$

Combining the fact that $\|\tilde{I} - \mathbf{e}_{i(r)}\|_\infty \leq 1$ under $\mathcal{G}_p^c$ with the previous results, we obtain the following upper bound on the approximation error of $\tilde{I}$:

$$\mathbb{E}_r[\|\tilde{I} - \mathbf{e}_{i(r)}\|_\infty] \;\leq\; \mathbb{P}[\mathcal{G}_p] \cdot 2\varepsilon + \mathbb{P}[\mathcal{G}_p^c] \cdot 1 \;\leq\; 2\varepsilon + 2k_{\text{eff}}\delta + \varepsilon_{\text{tail}}.$$

On the interval $[0, 1]$, the post-processing function $2x^3 - 3x^2$ satisfies the fol-

lowing inequalities:

$$\left|(2x^3 - 3x^2) - 0\right| \le 3x^2, \qquad \left|(2x^3 - 3x^2) - 1\right| \le 3(x-1)^2.$$

Therefore, for the post-processed $\tilde{I}'$ if $j = i(r)$, then

$$|\tilde{I}'_j - 1| = |\mathrm{PP}(\tilde{I}_j) - 1| \le 3(\tilde{I}_j - 1)^2 \le 12\varepsilon^2.$$

If $j \neq i(r)$, then

$$|\tilde{I}'_j - 0| = |\mathrm{PP}(\tilde{I}_j)| \le 3\tilde{I}_j^2 \le 3\varepsilon^2.$$

Hence, the upper bound for $\|\tilde{I}' - \mathbf{e}_{i(r)}\|_\infty$ can be derived in exactly the same manner as for $\|\tilde{I} - \mathbf{e}_{i(r)}\|_\infty$, and is therefore omitted. □

Finally, we establish a theorem about the effect of TSP token reordering.

Theorem 6. ***(Effect of TSP with compact support)*** *Suppose that a probability vector p is given, $r \sim \mathrm{Uniform}([0,1])$, and its corresponding index is $i(r)$. Let $\tilde{I}$ and $\tilde{I}'$ be the outputs of Algorithm 2, and assume there exists $R \in \mathbb{N}$ such that $\tilde{I}_j = 0$ and $\tilde{I}'_j = 0$ for $|j - i(r)| > R$. For the embedding matrix $W \in \mathbb{R}^{|\mathcal{V}| \times d}$ and its normalized row vectors $\bar{W}_i := W_i / \|W_i\|_2$, define*

$$\kappa_R := \sup_k \max_{|t| \le R} \frac{\|W_{k+t}\|_2}{\|W_k\|_2}, \;\; \bar{d}_t^{(p)} := \sum_{i=0}^{|\mathcal{V}|-1} p_i \, d_{\cos}(\bar{W}_{i+t}, \bar{W}_{i+t+1}), \;\; \text{and } D_R^{(p)} := \sum_{j=1}^{R} j \sum_{t=-j}^{j-1} \bar{d}_t^{(p)}.$$

Then for $p_b := \mathbb{P}(\mathcal{G}_p^c)$ and the target embedding vector $\bar{W}_{i(r)}$, the followings hold:

$$\text{If } (1-\varepsilon)^2 \ge 2R\varepsilon\kappa_R, \; \mathbb{E}_r\left[d_{\cos}(\tilde{I}^\top W, \bar{W}_{i(r)})\right] \le (1-p_b)\frac{\varepsilon\kappa_R}{(1-\varepsilon)^2} D_R^{(p)} + 2p_b.$$

$$\text{If } 1 - 3(2\varepsilon - \varepsilon^2)^2 \ge 6R\,\varepsilon^2\,\kappa_R, \; \mathbb{E}_r\left[d_{\cos}(\tilde{I}'^\top W, \bar{W}_{i(r)})\right] \le (1-p_b)\frac{3\,\varepsilon^2\,\kappa_R}{1 - 3(2\varepsilon - \varepsilon^2)^2} D_R^{(p)} + 2p_b.$$

Intuitively, $\bar{d}_t^{(p)}$ is the weighted sum of the cosine distances between the adjacent embedding vectors. After applying TSP, the upper bound for the expected distance between the obtained and the target embedding vectors decreases.

Proof. We only prove the formal inequality; the proof for for the latter is similar. Decompose $\mathbb{E}_r\left[d_{\cos}(\tilde{I}^\top W, \bar{W}_{i(r)})\right]$ as

$$\mathbb{E}_r\left[d_{\cos}(\tilde{I}^\top W, \bar{W}_{i(r)})\right] = (1-p_b)\,\mathbb{E}_r\left[d_{\cos}(\tilde{I}^\top W, \bar{W}_{i(r)}) \,\middle|\, \mathcal{G}_p\right] + p_b\,\mathbb{E}_r\left[d_{\cos}(\tilde{I}^\top W, \bar{W}_{i(r)}) \,\middle|\, \mathcal{G}_p^c\right].$$

Since $d_{\cos} \in [0,2]$, the second term is bounded by $2p_b$. We work on $\mathcal{G}_p$.
For $r \in \mathcal{G}_p$, Assumption 1 on $\widetilde{H}$ gives

$$\tilde{I}_{i(r)} \;\geq\; (1-\varepsilon)^2, \qquad \tilde{I}_j \;\leq\; \varepsilon \;\; (j \neq i(r)), \qquad \tilde{I}_j = 0 \;\; (|j - i(r)| > R). \tag{4.1}$$

Rewrite $\tilde{I}^\top W \;=\; \sum_j \tilde{I}_j W_j \;=\; \sum_j \tilde{I}_j \|W_j\|_2 \bar{W}_j$. Let $S := \sum_j \tilde{I}_j \|W_j\|_2$ and $w_j := \tilde{I}_j \|W_j\|_2 / S$. Then

$$\sum_j w_j = 1, \qquad w_j \geq 0, \qquad \text{and} \qquad \tilde{I}^\top W \;= Sm$$

where $m = \sum_j w_j \bar{W}_j$. Therefore, $d_{\cos}(\tilde{I}^\top W, \bar{W}_{i(r)}) = d_{\cos}(m, \bar{W}_{i(r)})$ by scale-invariance. Also $\|m\|_2 \leq 1$. By (4.1) and the definition of κ_R,

$$S \;\geq\; \tilde{I}_{i(r)} \|W_{i(r)}\|_2 \;\geq\; (1-\varepsilon)^2 \|W_{i(r)}\|_2, \qquad \|W_{i(r)\pm s}\|_2 \;\leq\; \kappa_R \, \|W_{i(r)}\|_2 \;\; (1 \leq s \leq R).$$

Hence, for $1 \leq s \leq R$,

$$w_{i(r)\pm s} \;=\; \frac{\tilde{I}_{i(r)\pm s} \|W_{i(r)\pm s}\|_2}{S} \;\leq\; \frac{\varepsilon \, \kappa_R}{(1-\varepsilon)^2}, \tag{4.2}$$

and by

$$S = \sum_{j=-R}^{R} \tilde{I}_j \|W_{i(r)}\|^2 \leq \tilde{I}_{i(r)} \|W_{i(r)}\|_2 + \sum_{0<|j|\leq R} \varepsilon \kappa_R \|W_{i(r)}\|_2 = (\tilde{I}_{i(r)} + 2R\varepsilon\kappa_R) \|W_{i(r)}\|_2,$$

we have

$$w_{i(r)} \;=\; \frac{\tilde{I}_{i(r)} \|W_{i(r)}\|_2}{S} \;\geq\; \frac{(1-\varepsilon)^2}{(1-\varepsilon)^2 + 2R\,\varepsilon\,\kappa_R}. \tag{4.3}$$

The assumed condition $(1-\varepsilon)^2 \geq 2R\,\varepsilon\,\kappa_R$ implies $w_{i(r)} \geq \frac{1}{2}$ by (4.3). Therefore $\langle m, \bar{W}_{i(r)} \rangle \geq w_{i(r)} - \sum_{i \neq k} w_i = 2w_{i(r)} - 1 \geq 0$, and with $\|m\|_2 \leq 1$,

$$d_{\cos}(m, \bar{W}_{i(r)}) \;=\; 1 - \frac{\langle m, \bar{W}_{i(r)} \rangle}{\|m\|_2} \;\leq\; 1 - \langle m, \bar{W}_{i(r)} \rangle \;=\; \sum_j w_j \, d_{\cos}(\bar{W}_j, \bar{W}_{i(r)}). \tag{4.4}$$

As $d_{\cos}(\bar{W}_{i(r)}, \bar{W}_{i(r)}) = 0$ and $w_j = 0$ for $|j - i(r)| > R$, (4.2) yields

$$d_{\cos}(m, \bar{W}_{i(r)}) \;\le\; \frac{\varepsilon\,\kappa_R}{(1-\varepsilon)^2} \sum_{s=1}^{R} \Big\{ d_{\cos}(\bar{W}_{i(r)+s}, \bar{W}_{i(r)}) + d_{\cos}(\bar{W}_{i(r)-s}, \bar{W}_{i(r)}) \Big\}.$$

For unit vectors $x_0, \dots, x_j$, the chord-triangle inequality gives

$$d_{\cos}(x_j, x_0) \;\le\; j \sum_{t=0}^{j-1} d_{\cos}(x_{t+1}, x_t).$$

Applying this in both directions,

$$d_{\cos}(\bar{W}_{i(r)\pm s}, \bar{W}_{i(r)}) \;\le\; s \sum_{t=0}^{s-1} d_{\cos}(\bar{W}_{i(r)\pm t\pm 1}, \bar{W}_{i(r)\pm t}).$$

Therefore, for fixed $r \in \mathcal{G}_p$,

$$d_{\cos}(\tilde{I}^\top W, \bar{W}_{i(r)}) \;\le\; \frac{\varepsilon\,\kappa_R}{(1-\varepsilon)^2} \sum_{s=1}^{R} s \sum_{t=0}^{s-1} \Big\{ d_{\cos}(\bar{W}_{i(r)+t+1}, \bar{W}_{i(r)+t}) + d_{\cos}(\bar{W}_{i(r)-t-1}, \bar{W}_{i(r)-t}) \Big\}.$$

On the good event, taking expectation over r (so k is distributed according to p), we get

$$\mathbb{E}_r\big[d_{\cos}(\bar{W}_{i(r)+t+1}, \bar{W}_{i(r)+t}) \mid \mathcal{G}_p \big] = \sum_i p_i\, d_{\cos}(\bar{W}_{i+t+1}, \bar{W}_{i+t}) = \bar{d}_t^{(p)},$$

and similarly for $-t-1$. Hence we have the following:

$$\mathbb{E}_r[d_{\cos}(\tilde{I}^\top W, \bar{W}_{i(r)}) \mid \mathcal{G}_p] \;\le\; \frac{\varepsilon\,\kappa_R}{(1-\varepsilon)^2} \sum_{s=1}^{R} s \sum_{t=0}^{s-1} \big(\bar{d}_t^{(p)} + \bar{d}_{-t-1}^{(p)} \big) \;=\; \frac{\varepsilon\,\kappa_R}{(1-\varepsilon)^2} \sum_{s=1}^{R} s \sum_{t=-s}^{s-1} \bar{d}_t^{(p)},$$

which completes the proof. □

4.5 Experiments

In this section, we explain our HE-based text generation scheme and experimental results.

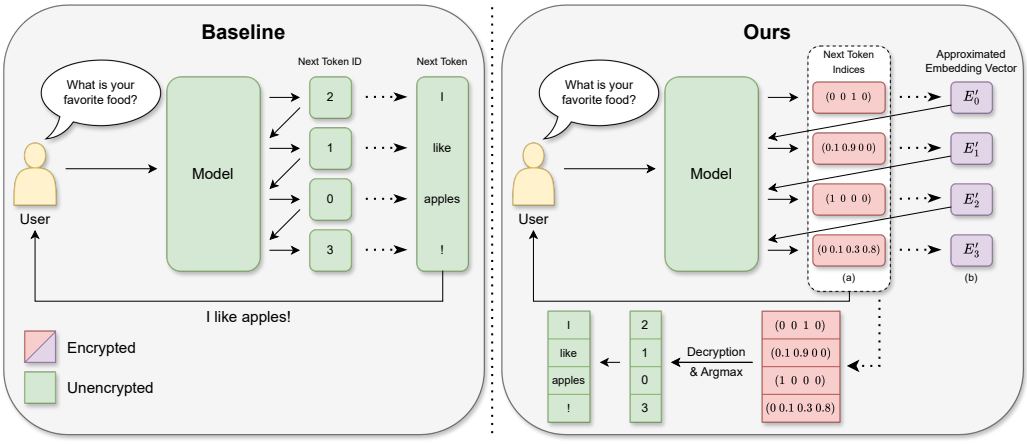


Figure 4.5: The schematic illustration of our work. **(Left)** Standard next token prediction. **(Right)** Our HE-based text generation. First, a user encrypts the embedding vectors of the prompt and the model performs the inference. However, since the output is a ciphertext, the model cannot select the index of the next-token. Instead, (a) the model approximates the one-hot vector of the next token (red), and (b) computes the approximated embedding vector (purple) and concatenates it into the model input. After generation, the model sends the concatenation of (a) to the user, who decrypts and applies argmax to recover to token. See Figure 4.2 and Section 4.3.2 for details on how (a) and (b) affect text generation.

4.5.1 HE-based text generation

We now explain text sampling under HE, illustrated in Figure 4.5. In standard next token prediction, a model predicts the indices of the next tokens. The user receives and decodes them into tokens. In contrast, under HE, we cannot use operations such as max or an if-statement to predict a particular token index. Therefore a model saves weighted indices in next token predictions and sends them to the user. Then the user decrypts the encrypted weighted indices and apply argmax decode to get the text.

We conduct our experiments under plaintext, not under HE, due to the limitations on computational budget. However, we anticipate that the result would not be significantly different under HE as can be found in the prior work [74, 99].

4.5.2 Criteria for text evaluation

We experimentally show that our TSP-based token reordering helps an LLM generate higher-quality texts, and post-processing and domain-specific fine-tuning (refer to Section 4.5.3) further provide auxiliary improvements in generation quality. In our experiments, we use `Llama2-7b-hf` [112] to generate texts. First, we define what is a corrupted text and two metrics for evaluating the ratio of the corrupted text and how a generated text is coherent. In experiments, we observed that when the generation by the model breaks, the model generates the token '`MS`' repeatedly and meaninglessly and we say the text is *corrupted*. The *corruption ratio* is defined as the percentage of the corrupted texts among all generated texts. The *corruption score* measures how many parts of a generated text is incoherent. We let GPT-4 [91] API (`gpt-4-0613`) grade the score on each text according to the degree of the corruptness of the text. If 0$\sim$20% of a text is incoherent (e.g., hard to understand or inconsistent), then the text is assigned a corruption score of 0. On the other hand, if 80$\sim$100% of a text is incoherent, then its corruption score is 4. We use these scores as the metrics to evaluate the quality of generated texts. Exceptionally, if the model generates repeated '`MS`' tokens, we automatically give this text a score of 4. The prompt to grade the score can be seen in Table 4.4. Appendix B.5 presents examples of well and poorly written texts with the corruption score 0 and 4.

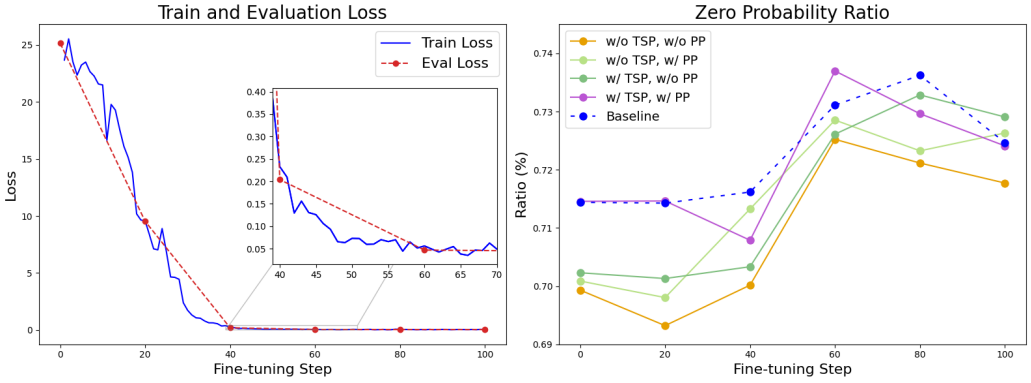


Figure 4.6: **(Left)** The train and evaluation loss and **(Right)** the ratios of zero-probability tokens during fine-tuning. After 60 steps, both the train and evaluation loss converge to zero, and the ratios of zero-probability tokens increase. In the right graph, we can see that the ratios of zero-probability tokens increase when TSP or post-processing (PP) is applied. Baseline means the case when the naive probabilistic sampling (not our algorithm) was used.

4.5.3 Making generations more coherent via domain-specific fine-tuning

In this section, we explore how domain-specific fine-tuning improves the coherence of generated text under CKKS. Because we use probabilistic sampling, the generated text has high diversity, but can lack coherence. We find that domain-specific fine-tuning can mitigate this diversity-coherence trade-off. As fine-tuning progresses, we observe that the number of zero-probability tokens increases—while Softmax is strictly positive, half or single-precision in LLM inference causes underflow to zero. The right panel of Figure 4.6 shows this observation.

This makes our sampling algorithm behave similarly to top-p/k because fine-tuning narrows the number of candidate tokens for sampling. This improves the balance between creativity and coherence, resulting in more coherent text generation. Moreover, this is especially beneficial in our setting, since sampling methods using thresholds usually do not consider zero-probability tokens and their thresholds are not practical under CKKS. Further explanation can be found in Section 4.5.4.

4.5.4 Experimental results

We use `Llama2-7b-hf` to generate texts using our proposed TSP-based token reordering. We conduct experiments on four settings, depending on whether TSP-based token reordering and post-processing (PP) are applied. Also, we fine-tune the model using LoRA [57] with lora rank 2. During fine-tuning, the batch size is 1, the gradient accumulation step is 32, the max sequence length is 4096, and the learning rate is $5 \cdot 10^{-5}$. Decreasing the temperature T increases the dominant probability of a softmax output so that one obtain a near one-hot vector for the highest-probability token. However, it widens the approximation domain of the exponential and division operations, making implementing softmax under HE significantly more costly. Therefore, in this work, we fix the temperature T as 1.

For each generation, the model generates 1500 tokens, so even if the text is does not end with a complete sentence, this does not affect the evaluation of the text. The prompt for the model is "Please introduce yourself."

Figure 4.6 shows (left) the train and evaluation loss and (right) the ratio of zero-probability tokens during fine-tuning. As both the train and evaluation loss converge to zero after 60 steps, it is found that the ratio of zero-probability tokens increases in the right graph. One can see that TSP-based token reordering and post-processing also increases of zero-probability token ratio.

Table 4.5 and Figure 4.7 present the corruption scores and the corruption ratios (%) for each case during fine-tuning. Baseline means when the naive probabilistic sampling was used instead of our algorithm. We record the average of the results of three different seeds. In this table, one can see that TSP-based token reordering, post-processing (PP), and fine-tuning all contribute to reducing the corruption scores and the corruption ratios. First, PP helps the model generate a vector closer to one-hot vectors, and TSP makes adjacent embedding vectors semantically similar. In addition, if fine-tuning is done, the corruption score and ratio also decrease, suggesting that the model generate higher-quality texts. The corruption score can approaches to the baseline score when fine-tuning step is 60 and both TSP and PP are applied. And the corruption ratio mostly converges to zero when fine-tuning step is 60 or more. Also, the corruption score and ratio of the generated texts using our algorithm approach to the baseline results when all of our methods are applied. Note that two greenish graphs are not directly comparable as the applied methods for each case are different. Results for other prompts are provided in Appendix B.4, and the effect of our methods in each next-token prediction step can be seen in Appendix B.3.

Table 4.4: The criteria to measure the quality of the generated text using Algorithm 2.

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: 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. 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. After reading the text, assign a score based on the degree of corruption in the following format: **X point(s): REASON** Here is the scoring system: 4 points: If 80-100% of the text is corrupted. 3 points: If 60-80% of the text is corrupted. 2 points: If 40-60% of the text is corrupted. 1 point: If 20-40% of the text is corrupted. 0 points: If 0-20% of the text is corrupted. **Special Case:** Regardless of the above criteria, if the sequence MS is repeated meaninglessly more than two times, assign **4 points**. Here is the text I'll show you:

Table 4.5: The results of text generation using Algorithm 2, with the TSP-based token reordering and post-processing (PP). Here *score* and *ratio* refer to the corruption score and ratio (%), respectively. For each case, we record the average of the results of different three seeds. *Baseline* means the probabilistic sampling, not our methods. Among the results using our methods, the best results are marked as **bold** and the second are underlined for both metrics. For baseline, the lowest corruption score is marked as **bold**. We can observe that when the fine-tuning step reaches 60, the corruption score is lowest when both the TSP and PP are applied. Furthermore, as fine-tuning progresses, the corruption ratio mostly converges to zero.

Prompt	Please introduce yourself.									
TSP Reordering	False				True				Baseline	
Post-processing	False		True		False		True			
Fine-tuning Step	Score	Ratio	Score	Ratio	Score	Ratio	Score	Ratio	Score	Ratio
0	1.5433	19.00	1.1867	12.00	1.2800	10.33	0.8967	8.33	0.6367	0.00
20	1.4033	12.00	1.0733	8.33	1.0133	6.33	0.8433	6.33	0.8267	0.00
40	1.2133	8.00	0.9067	5.33	1.0567	6.33	0.7600	3.00	0.5833	0.00
60	0.6500	0.67	0.4967	0.33	0.6567	2.00	**0.4548**	0.33	0.4700	0.00
80	0.7267	2.33	0.5900	1.00	0.5300	0.67	0.4967	0.33	**0.4367**	0.00
100	0.8500	2.00	0.6567	0.67	0.5367	1.00	0.5900	**0.00**	0.4667	0.00

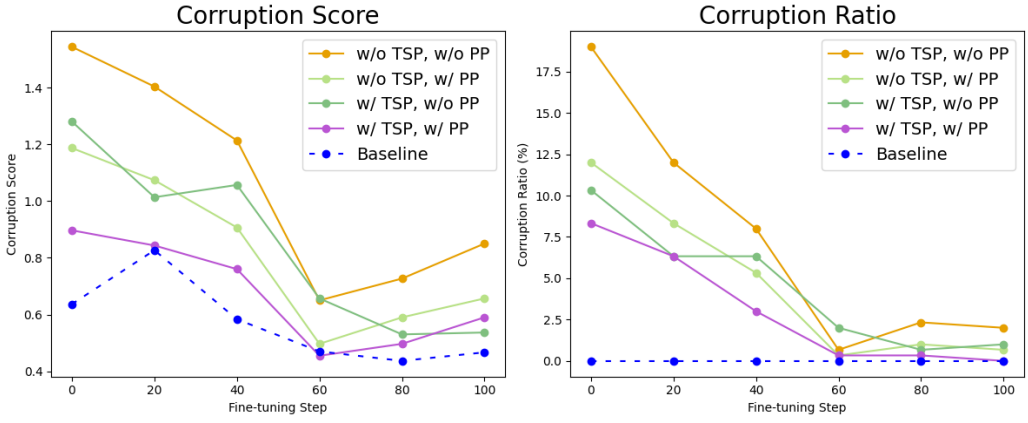


Figure 4.7: **(Left)** The corruption scores and **(Right)** the ratios over 100 generated texts for each case. The corruption score is minimized when the fine-tuning step is 60 and both TSP and PP are applied and the corruption ratio mostly converges to zero when all of our methods are combined. We can observe that the quality of generated texts by our algorithm is comparable to that of the baseline.

Chapter 5

Conclusion and Future Directions

In this thesis, we present methods for the efficient implementation of LLMs under HE, particularly in the CKKS. First, we modify the attention mechanism and explore the previously unexamined effect of LoRA speedups for fine-tuning and inference of LLMs under HE. Second, we propose a novel and efficient method for stable text generation under CKKS which has not yet been developed.

Although we have contributed to the efficient implementation of language models under HE, there remain many directions for improvement. First, including ours, the existing works focus on the implementing the same of similar architecture as those in the plaintext. However, these architectures incur an inevitable high computational cost due to non-polynomial operations. Therefore, we can devise an architecture based solely on operations which are supported by CKKS. Next, one can develop approximate algorithm for operations not yet implemented under CKKS. For example, we proposed a random sampling algorithm for next-token prediction. Similarly, if one develops a homomorphic thresholding algorithm, we could implement top-p or top-k sampling under CKKS. With these advances, we will be able to use our private data more freely and securely.

Bibliography

[1] Y. Akimoto, K. Fukuchi, Y. Akimoto, and J. Sakuma. Privformer: Privacy-preserving transformer with MPC. *IEEE European Symposium on Security and Privacy*, pages 392–410, 2023.

[2] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. The traveling salesman problem: A computational study. *Princeton Series in Applied Mathematics*, 2007.

[3] M. Avitan, M. Baruch, N. Drucker, I. Zimerman, and Y. Goldberg. Efficient decoding methods for language models on encrypted data. *arXiv:2509.08383*, 2025.

[4] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv:1607.06450*, 2016.

[5] A. A. Badawi, L. Hoang, C. F. Mun, K. Laine, and K. M. M. Aung. PrivFT: Private and fast text classification with homomorphic encryption. *IEEE Access*, 8:226544–226556, 2020.

[6] Y. Bae, J. H. Cheon, G. Hanrot, J. H. Park, and D. Stehlé. Plaintext-ciphertext matrix multiplication and fhe bootstrapping: Fast and fused. *CRYPTO*, 2024.

[7] Y. Bae, J. H. Cheon, G. Hanrot, J. H. Park, and D. Stehlé. Plaintext-ciphertext matrix multiplication and fhe bootstrapping: Fast and fused. *CRYPTO*, 2024.

[8] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[9] M. Baruch, N. Drucker, G. Ezov, E. Kushnir, J. Lerner, O. Soceanu, and I. Zimerman. Training large scale polynomial CNNs for E2E inference over homomorphic encryption. *arXiv:2304.14836*, 2023.

[10] A. Berlinet and C. Thomas-Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Springer Science & Business Media, 2011.

[11] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. *CRYPTO*, 2012.

[12] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Annual cryptology conference*, pages 868–886. Springer, 2012.

[13] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory*, 6:1–36, 2014.

[14] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.

[15] J. Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. *Neural Information Processing Systems*, 1989.

[16] N. Carlini, D. Paleka, K. D. Dvijotham, T. Steinke, J. Hayase, A. F. Cooper, K. Lee, M. Jagielski, M. Nasr, A. Conmy, E. Wallace, D. Rolnick, and F. Tramèr. Stealing part of a production language model. *International Conference on Machine Learning*, 2024.

[17] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. *International Workshop on Semantic Evaluation*, 2017.

[18] T. Chen, H. Bao, S. Huang, L. Dong, B. Jiao, D. Jiang, H. Zhou, J. Li, and F. Wei. THE-X: Privacy-preserving transformer inference with homomorphic encryption. *Findings of the Association for Computational Linguistics*, pages 3510–3520, 2022.

[19] W.-L. Chen, C.-K. Wu, H.-H. Chen, and C.-C. Chen. Fidelity-enriched contrastive search: reconciling the faithfulness-diversity trade-off in text generation. *arXiv preprint arXiv:2310.14981*, 2023.

[20] Y. Chen, Q. Zeng, H. Ji, and Y. Yang. Skyformer: Remodel self-attention with gaussian kernel and nyström method. *Neural Information Processing Systems*, 2021.

[21] J. H. Cheon, A. Kim, M. Kim, and Y. Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*, pages 409–437. Springer, 2017.

[22] J. H. Cheon, A. Kim, M. Kim, and Y. S. Song. Homomorphic encryption for arithmetic of approximate numbers. *ASIACRYPT*, 2017.

[23] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song. A full rns variant of approximate homomorphic encryption. In *Selected Areas in Cryptography–SAC 2018: 25th International Conference, Calgary, AB, Canada, August 15–17, 2018, Revised Selected Papers 25*, pages 347–368. Springer, 2019.

[24] J. H. Cheon, D. Kim, and D. Kim. Efficient homomorphic comparison methods with optimal complexity. *ASIACRYPT*, 2020.

[25] S. Chevillard, M. Joldeş, and C. Lauter. Sollya: An environment for the development of numerical codes. *International Congress on Mathematical Software*, 2010.

[26] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachene. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. *ASIACRYPT*, 2016.

[27] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachene. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. *ASIACRYPT*, 2016.

[28] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Tfhe: fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, 2020.

[29] W. Cho, G. Hanrot, T. Kim, M. Park, and D. Stehlé. Fast and accurate homomorphic softmax evaluation. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 4391–4404, 2024.

[30] S. G. Choi, D. Dachman-Soled, S. D. Gordon, L. Liu, and A. Yerukhimovich. Secure sampling with sublinear communication. In *Theory of Cryptography Conference*, pages 348–377. Springer, 2022.

[31] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. *International Conference on Learning Representations*, 2020.

[32] H. CryptoLab. HEaaN library. `https://www.cryptolab.co.kr/en/products-en/heaan-he/`, 2022.

[33] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. Language modeling with gated convolutional networks. *International Conference on Machine Learning*, 2017.

[34] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.

[35] W. B. Dolan and C. Brockett. Automatically constructing a corpus of sentential paraphrases. *International Workshop on Paraphrasing*, 2005.

[36] Y. Dong, W. jie Lu, Y. Zheng, H. Wu, D. Zhao, J. Tan, Z. Huang, C. Hong, T. Wei, and W. Chen. PUMA: Secure inference of Llama-7b in five minutes. *arXiv:2307.12533*, 2023.

[37] A. Dubey, A. Jauhri, A. Kadian, et al. The Llama 3 herd of models. *arXiv:2407.21783*, 2024.

[38] L. Ducas and D. Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. *EUROCRYPT*, 2015.

[39] L. Ducas and D. Micciancio. Fhew: bootstrapping homomorphic encryption in less than a second. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 617–640. Springer, 2015.

[40] C. Dwork. Differential privacy. *International Colloquium on Automata, Languages, and Programming*, 2006.

[41] A. Eremenko and P. Yuditskii. Uniform approximation of sgn(x) by polynomials and entire functions. *arXiv preprint math/0604324*, 2006.

[42] European Union. Regulation (EU) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation). `https://eur-lex.europa.eu/eli/reg/2016/679/oj`, 2016.

[43] A. Fan, M. Lewis, and Y. Dauphin. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*, 2018.

[44] J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive*, 2012.

[45] L. Folkerts and N. G. Tsoutsos. Tyche: Probabilistic selection over encrypted data for generative language models. *Cryptology ePrint Archive*, 2024.

[46] J. Geiping and T. Goldstein. Cramming: Training a language model on a single gpu in one day. *International Conference on Machine Learning*, 2023.

[47] C. Gentry. Fully homomorphic encryption using ideal lattices. *ACM Symposium on Theory of Computing*, 2009.

[48] D. Giampiccolo, B. Magnini, I. Dagan, and B. Dolan. The third pascal recognizing textual entailment challenge. *ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, 2007.

[49] A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[50] J. Gu, K. Cho, and V. O. Li. Trainable greedy decoding for neural machine translation. *arXiv preprint arXiv:1702.02429*, 2017.

[51] S. Halevi and V. Shoup. Faster homomorphic linear transformations in HElib. *CRYPTO*, 2018.

[52] K. Han, S. Hong, J. H. Cheon, and D. Park. Logistic regression on homomorphic encrypted data at scale. *AAAI Conference on Artificial Intelligence*, 2019.

[53] M. Hao, H. Li, H. Chen, P. Xing, G. Xu, and T. Zhang. Iron: Private inference on transformers. *Neural Information Processing Systems*, 2022.

[54] P. He, X. Liu, J. Gao, and W. Chen. DeBERTa: Decoding-enhanced BERT with disentangled attention. *International Conference on Learning Representations*, 2021.

[55] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

[56] S. Hong, J. Park, W. Cho, H. Choe, and J. Cheon. Secure tumor classification by shallow neural network using homomorphic encryption. *BMC Genomics*, 23:284, 2022.

[57] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. *International Conference on Learning Representations*, 2022.

[58] J. Jang, Y. Lee, A. Kim, B. Na, D. Yhee, B. Lee, J. H. Cheon, and S. Yoon. Privacy-preserving deep sequential model with matrix homomorphic encryption. *ACM Asia Conference on Computer and Communications Security*, 2022.

[59] X. Jiang, M. Kim, K. Lauter, and Y. Song. Secure outsourced matrix computation and application to neural networks. *ACM SIGSAC Conference on Computer and Communications Security*, 2018.

[60] C. Jin, M. Ragab, and K. M. M. Aung. Secure transfer learning for machine fault diagnosis under different operating conditions. *International Conference on Provable Security*, 2020.

[61] D. S. Johnson and L. A. McGeoch. Implementation challenge for tsp heuristics. In *Encyclopedia of Algorithms*, pages 398–400. Springer, 2002.

[62] N. Jovanovic, M. Fischer, S. Steffen, and M. Vechev. Private and reliable neural network inference. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 1663–1677, 2022.

[63] M. Junczys-Dowmunt, R. Grundkiewicz, T. Dwojak, H. Hoang, K. Heafield, T. Neckermann, F. Seide, U. Germann, A. Fikri Aji, N. Bogoychev, et al. Marian: Fast neural machine translation in C++. *Association for Computational Linguistics: System Demonstrations*, 2019.

[64] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv:2001.08361*, 2020.

[65] R. M. Karp. *Reducibility among Combinatorial Problems*. Springer, 1972.

[66] J. Kim, S. Park, J. Lee, and J. H. Cheon. Privacy-preserving embedding via look-up table evaluation with fully homomorphic encryption. In *Forty-first International Conference on Machine Learning*, 2024.

[67] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. *International Conference on Learning Representations*, 2020.

[68] E. Lee, J.-W. Lee, J.-S. No, and Y.-S. Kim. Minimax approximation of sign function by composite polynomial for homomorphic comparison. *IEEE Transactions on Dependable and Secure Computing*, 19:3711–3727, 2021.

[69] E. Lee, J.-W. Lee, J. Lee, Y.-S. Kim, Y. Kim, J.-S. No, and W. Choi. Low-complexity deep convolutional neural networks on fully homomorphic encryption using multiplexed parallel convolutions. *International Conference on Machine Learning*, 2022.

[70] G. Lee, M. Kim, J. H. Park, S.-w. Hwang, and J. H. Cheon. Privacy-preserving text classification on BERT embeddings with homomorphic encryption. *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2022.

[71] J. Lee, E. Lee, J.-W. Lee, Y. Kim, Y.-S. Kim, and J.-S. No. Precise approximation of convolutional neural networks for homomorphically encrypted data. *IEEE Access*, 2023.

[72] J. Lee, E. Lee, J.-W. Lee, Y. Kim, Y.-S. Kim, and J.-S. No. Precise approximation of convolutional neural networks for homomorphically encrypted data. *IEEE Access*, 11:62062–62076, 2023.

[73] J.-W. Lee, H. Kang, Y. Lee, W. Choi, J. Eom, M. Deryabin, E. Lee, J. Lee, D. Yoo, Y.-S. Kim, and J.-S. No. Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. *IEEE Access*, 10: 30039–30054, 2022.

[74] S. Lee, G. Lee, J. W. Kim, J. Shin, and M.-K. Lee. HETAL: Efficient privacy-preserving transfer learning with homomorphic encryption. *International Conference on Machine Learning*, 2023.

[75] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[76] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *Association for Computational Linguistics*, 2020.

[77] D. Li, R. Shao, H. Wang, H. Guo, E. P. Xing, and H. Zhang. MPCFormer: Fast, performant and private transformer inference with MPC. *International Conference on Learning Representations*, 2023.

[78] Y. Li, W. Yu, and J. Zhao. Privtuner with homomorphic encryption and lora: A P3EFT scheme for privacy-preserving parameter-efficient fine-tuning of ai foundation models. *arXiv:2410.00433*, 2024.

[79] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv:1907.11692*, 2019.

[80] Y. Liu, J. Gu, N. Goyal, X. Li, S. Edunov, M. Ghazvininejad, M. Lewis, and L. Zettlemoyer. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742, 2020.

[81] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *International Conference on Learning Representations*, 2019.

[82] J. Lu, J. Yao, J. Zhang, X. Zhu, H. Xu, W. Gao, C. Xu, T. Xiang, and L. Zhang. SOFT: Softmax-free transformer with linear complexity. *Neural Information Processing Systems*, 2021.

[83] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. *EUROCRYPT*, 2010.

[84] S. McCallum. ChatGPT banned in Italy over privacy concerns. `https://www.bbc.com/news/technology-65139406`, 2023.

[85] L. McInnes, J. Healy, and J. Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[86] C. Meister, T. Pimentel, G. Wiher, and R. Cotterell. Locally typical sampling. *Transactions of the Association for Computational Linguistics*, 11:102–121, 2023.

[87] A. Mok. Amazon, Apple, and 12 other major companies that have restricted employees from using ChatGPT. `https://www.businessinsider.com/chatgpt-companies-issued-bans-restrictions-openai-ai-amazon-apple-2023-7`, 2023.

[88] J. Moon, D. Yoo, X. Jiang, and M. Kim. Thor: Secure transformer inference with homomorphic encryption. *Cryptology ePrint Archive*, 2024.

[89] M. Nguyen, A. Baker, C. Neo, A. Roush, A. Kirsch, and R. Shwartz-Ziv. Turning up the heat: Min-p sampling for creative and coherent llm outputs. *arXiv preprint arXiv:2407.01082*, 2024.

[90] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, 2006.

[91] OpenAI. GPT-4 technical report. *arXiv:2303.08774*, 2023.

[92] OpenAI. ChatGPT. `https://www.openai.com/chatgpt`, 2024.

[93] Q. Pang, J. Zhu, H. Möllering, W. Zheng, and T. Schneider. BOLT: Privacy-preserving, accurate and efficient inference for transformers. *IEEE Symposium on Security and Privacy*, 2024.

[94] D. Park, E. Lee, and J.-W. Lee. Powerformer: Efficient privacy-preserving transformer with batch rectifier-power max function and optimized homomorphic attention. *Cryptology ePrint Archive*, 2024.

[95] A. Radford. Improving language understanding by generative pre-training. *OpenAI blog*, 2018.

[96] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.

[97] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67, 2020.

[98] E. Y. Remez. Sur la détermination des polynômes d'approximation de degré donnée. *Communications of the Mathematical Society of Kharkov*, 10:41–63, 1934.

[99] D. Rho, T. Kim, M. Park, J. W. Kim, H. Chae, E. K. Ryu, and J. H. Cheon. Encryption-friendly llm architecture. *arXiv preprint arXiv:2410.02486*, 2024.

[100] D. Rho, S. Seo, H. Sung, C. Min, and E. K. Ryu. Traveling salesman-based token ordering improves stability in homomorphically encrypted language models. *arXiv preprint arXiv:2510.12343*, 2025.

[101] O. Richter and R. Wattenhofer. Normalized attention without probability cage. *arXiv:2005.09561*, 2020.

[102] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation*, 4:169–180, 1978.

[103] P. Rizomiliotis and A. Triakosia. On matrix multiplication with homomorphic encryption. *Cloud Computing Security Workshop*, 2022.

[104] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *NeurIPS Workshop on Energy Efficient Machine Learning and Cognitive Computing*, 2019.

[105] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. *Conference on Empirical Methods in Natural Language Processing*, 2013.

[106] State of California. California consumer privacy act of 2018. `https://oag.ca.gov/privacy/ccpa`, 2018.

[107] D. Stehlé, R. Steinfeld, K. Tanaka, and K. Xagawa. Efficient public key encryption based on ideal lattices. *ASIACRYPT*, 2009.

[108] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[109] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *Neural Information Processing Systems*, 2020.

[110] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv:2302.13971*, 2023.

[111] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv:2307.09288*, 2023.

[112] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv:2307.09288*, 2023.

[113] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart. Stealing machine learning models via prediction APIs. *USENIX Security Symposium*, 2016.

[114] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Neural Information Processing Systems*, 2017.

[115] A. K. Vijayakumar, M. Cogswell, R. R. Selvaraju, Q. Sun, S. Lee, D. Crandall, and D. Batra. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*, 2016.

[116] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 2018.

[117] A. Warstadt, A. Singh, and S. R. Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019.

[118] A. Williams, N. Nangia, and S. R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018.

[119] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Transformers: State-of-the-art natural language processing. *Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020.

[120] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel. mT5: A massively multilingual pre-trained text-to-text transformer. *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021.

[121] A. C. Yao. Protocols for secure computations. *IEEE Annual Symposium on Foundations of Computer Science*, 1982.

[122] A. C. Yao. Protocols for secure computations. *Symposium on Foundations of Computer Science*, 1982.

[123] J. Zhang, J. Liu, X. Yang, Y. Wang, K. Chen, X. Hou, K. Ren, and X. Yang. Secure transformer inference made non-interactive. *Cryptology ePrint Archive*, 2024.

[124] J. Zhang, X. Yang, L. He, K. Chen, W.-j. Lu, Y. Wang, X. Hou, J. Liu, K. Ren, and X. Yang. Secure transformer inference made non-interactive. *Network and Distributed System Security*, 2025.

[125] P. Zhang, A. Duan, and H. Lu. An efficient homomorphic argmax approximation for privacy-preserving neural networks. *Cryptography*, 8(2):18, 2024.

[126] Y. Zhang, C. Chen, T. Ding, Z. Li, R. Sun, and Z.-Q. Luo. Why transformers need Adam: A Hessian perspective. *Workshop on Efficient Systems for Foundation Models at ICML*, 2024.

[127] X. Zheng, H. Li, and D. Wang. A new framework for fast homomorphic matrix multiplication. *Cryptology ePrint Archive*, 2023.

[128] I. Zimerman, M. Baruch, N. Drucker, G. Ezov, O. Soceanu, and L. Wolf. Converting transformers to polynomial form for secure inference over homomorphic encryption. *International Conference on Machine Learning*, 2024.

Appendix A

Details For Homomorphic Encryption

A.1 Homomorphic matrix multiplication algorithm

Consider $d \times d$ homomorphic matrix multiplication. JKLS [59] suggests the fastest known HE square matrix multiplication algorithm with $\mathcal{O}(d)$ HE computational complexity (for more precise number of each operation, refer to Table A.1) when the matrix elements can be packed in a ciphertext, specifically when $d^2 \leq N/2$. The required HE operations include $\mathsf{Add}, (\mathsf{p})\mathsf{Rot}, \text{and } (\mathsf{p})\mathsf{Mult}$. The computation time of Rot and Mult operations differ between plaintext and ciphertext (see Table 2.1), with operations in ciphertext being significantly slower. Consequently, CCMM is much slower than PCMM; the number of different message space operations is exactly $d + 2\sqrt{d}$ for one homomorphic matrix multiplication. For precise time difference between two kinds of homomorphic matrix multiplication, see Section 3.5.3.

When $d^2 > N/2$, where we cannot encrypt the matrix into one ciphertext, we can also utilize the JKLS algorithm by computing block-wise matrix multiplication. For example, if we take $N = 2^{16}$ for the base ring degree, we can pack two 128×128 matrices into one ciphertext. When $d = 768$, we require 18 ciphertexts to pack $d \times d$ matrix. After packing each block matrices into ciphertexts, we repeat the corresponding block-wise matrix multiplication to all the weights. Thus, for large-size matrix multiplications, the computation time gap between CCMM and PCMM will be proportional to the number of ciphertexts required for packing.

In addition, JKLS introduced a homomorphic rectangular matrix multiplcation

of size $\ell \times d$ by $d \times d$ or vice-versa, with $\ell < d$ and $\ell d \leq N/2$. We can use this method to evaluate LoRA CCMMs, where we need to evaluate the homomorphic rectangular matrix multiplication sequentially. For more detailed HE operations numbers, see Table A.2.

Table A.1: The number of each HE operation for one $d \times d$ homomorphic matrix multiplication required by JKLS [59] algorithm.

Operations	Add	pMult	Rot	Mult	Depth
JKLS [59]	$6d$	$4d$	$3d + 5\sqrt{d}$	d	3

Table A.2: The number of each HE operation for rectangular homomorphic matrix multiplication $\ell \times d$ by $d \times d$ required by JKLS [59] rectangular matrix algorithm.

Operations	Add	pMult	Rot	Mult	Depth
JKLS [59]	$3d + 2\ell + \log(d/\ell)$	$3d + 2\ell$	$3\ell + 5\sqrt{d} + \log(d/\ell)$	ℓ	3

A.2 Optimizations for homomorphic matrix multiplication

The CKKS scheme packs all input data into a polynomial from encoding (Ecd). The packed data are assigned to fixed positions within the polynomial, and to change these positions, we have to use the (p)Rot operation. In the homomorphic matrix multiplication algorithm, several (p)Rot operations are required. Our target model has sequential matrix multiplications, so it is more efficient to fix the packing structure. Among various packing methods, we use row-wise packing (see Figure A.1a) for all packed data except for LoRA weights, which will be explained in the next subsection. Because using the FGb parameter can pack a total 2^{15} data in one message space, we consider one plain/ciphertext having 2^{15} data, which corresponds to the size either 128×256 or 256×128 in our implementation. Thus, during our model computation, we will assume one plain/ciphertext is packed 128×256 matrix row-wise.

A.2.1 Homomorphic matrix multiplication

Here, we introduce the optimization techniques used in our homomorphic matrix multiplication implementation. We adopt three optimization techniques to improve the speed of homomorphic matrix multiplication:

- **Evalation in lower level:** Because homomorphic matrix multiplication contains numerous HE operations and its speed depends on the current level, the ciphertexts containing matrix values are brought down to the minimum required level before performing the evaluation [7, 94], even if BTS may be needed for the next block evaluation. We adhere to this strategy in our homomorphic matrix multiplication by taking the JLKS [59] algorithm, which requires 3 levels. Consequently, evaluating the bottom-level (7 level) PCMM is 2.35× faster than the top-level (12 level) PCMM (refer to Table A.3) of size 128×256 by 256×128. This difference is even more pronounced in CCMM.

- **Pre-computation and reuse for left-matrix:** When we perform homomorphic matrix multiplication of size 128×768 by 768×768, we need to split the matrices as three and eighteen block matrices (see Figure A.1c), respectively. And we go through each block matrix multiplication to get a corresponding result. From the property of block matrix multiplication, we know the same left matrices are used several times to multiply by the right block matrices. Thus, we do pre-computation using JKLS [59] and save these objects for left-matrices. Using this strategy, we can reduce the total number of JKLS algorithm calls, which require numerous HE operations (as shown in Table A.1), from 36 to 21. This is a trade-off between memory and computation time.

- **Lazy key-switching/rescaling:** When a ciphertext is evaluated by an automorphism, we need to do a key-switching operation. This is an expensive process. If the result ciphertext is made from several ciphertexts resulting in the same automorphism, we are used to take *lazy* key-switching (hoisting) [51]; that is, we go through only one key-switching after collecting all results instead of doing key-switching for each automorphism. By taking lazy key-switching, we can speed up the required computation time. Lazy rescaling is also a similar strategy for reducing computation time and added errors when doing a rescaling operation, which is an operation contained in Mult.

Table A.3: Computation time comparison between PCMMs according to the different levels. The lower-level homomorphic matrix multiplcation is faster than the top-level evaluation.

PCMM	Time (s)	Factor
12 level	0.242	1
7 level	0.103	**2.35**

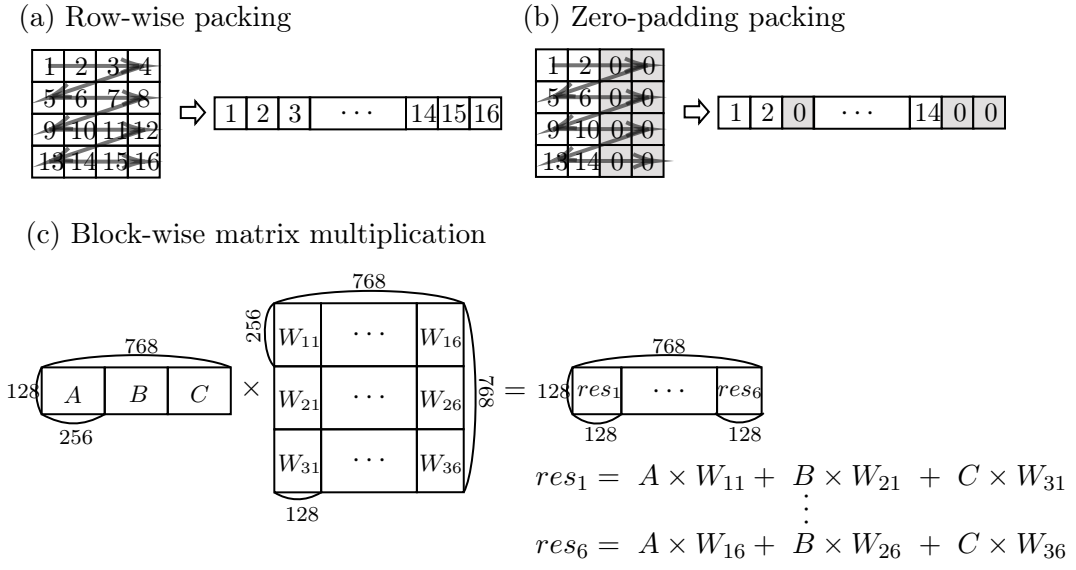


Figure A.1: Row-wise packing method for matrix representation, utilizing zero-padding for non-square matrices, followed by block-wise matrix multiplication for efficient processing of large matrices.

A.2.2 LoRA style CCMM

LoRA fine-tuning consists of two CCMMs. Because the LoRA weights are rectangular matrices, we need to do zero-padding (see Figure A.1b) for the second weight to make it a square matrix. After that, we can follow the optimized homomorphic rectangular matrix multiplication algorithm sequentially, which we call the *Rectangular MM* algorithm in the following. However, when we consider that our target LoRA rank(=2) is quite small, this Rectangular MM approach generates an inefficiency since there are many non-usable spaces in the ciphertext from

zero-padding. Thus, we make a more efficient algorithm for LoRA CCMMs than following the Rectangular MM. Our optimized algorithm can be applied in cases similar to LoRA CCMMs, which involve sequential matrix multiplication where one dimension is significantly smaller than the other. In our model, we also use these optimized CCMMs during classification head layer evaluation, where we have to perform homomorphic rectangular matrix multiplications of size 1×768 by 768×32 and 1×32 by 32×1024. For convenience, this subsection will describe the case regarding LoRA CCMMs with rank 2: 128×768 by 768×2 and 128×2 by 2×768, sequentially.

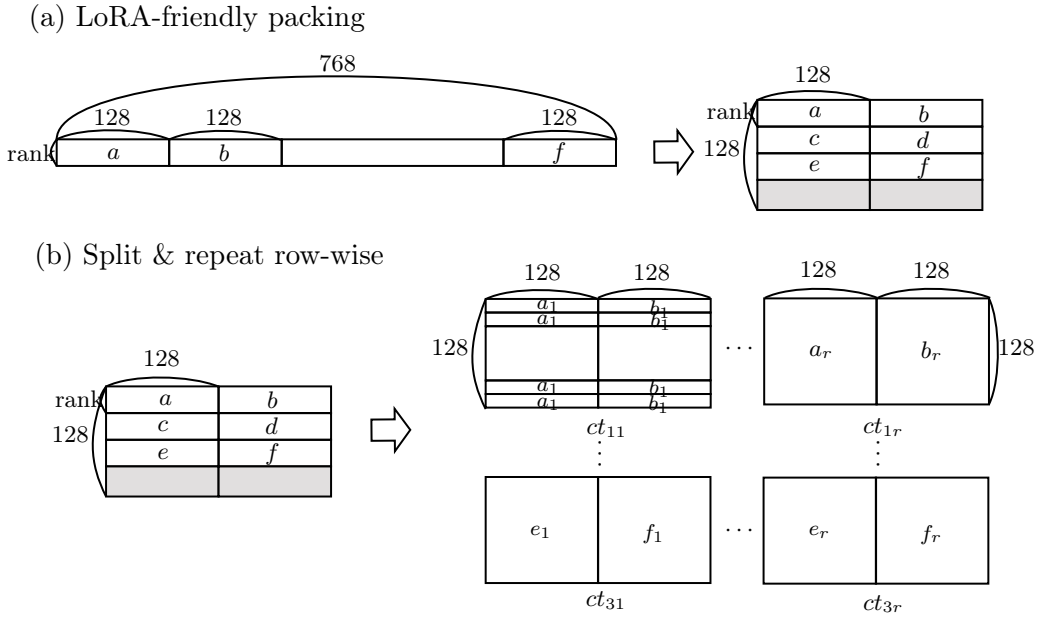


Figure A.2: LoRA-friendly packing is used when the given matrix has one long and one short size. Split & repeat row-wise divides and copies each row into ciphertexts, which is used during LoRA CCMMs and a_i denotes i-th row of matrix a. The shaded block matrices represent zero-padded blocks.

LoRA-friendly packing

In our target model, the LoRA weights are either $768 \times$ rank or rank $\times 768$. Both weights have one long and one short dimension. Instead of directly following row-wise packing used in entire message packing, we pack LoRA weights with another

approach. Figure A.2a represents the packing method, called LoRA-friendly packing, used in LoRA weights. Even if $768 \times$ rank is a column rectangular matrix shape, we follow the LoRA-friendly packing by considering the transposed column rectangular matrix. This does not mean we evaluate homomorphic transposition, where we use an algorithm in JKLS [59] that also has numerous HE operations. But treat its column rectangular matrix with the corresponding order in a row-wise packed structure:

$$\{A_{ij}\}_{1\leq i\leq 768, 1\leq j\leq \text{rank}} \mapsto \{B_k\}_{1\leq k\leq N/2} = \begin{cases} A_{ij} & \text{if } k = i + 768 * (j-1) \\ 0 & \text{otherwise} \end{cases}$$

By following this, we can remove the required evaluations of transposition during backpropagation computation.

Optimized homomorphic LoRA matrix multiplication algorithm

We optimize the sequential LoRA CCMMs in Algorithm 6. This algorithm consists of several HE operations and three operation blocks: (i) Split & repeat row-wise (Algorithm 3) for weights, (ii) repeat column-wise (Algorithm 4), and (iii) collect into the first column (Algorithm 5).

Comparison Rectangular MM and optimized LoRA CCMMs

We compare two approaches, Rectangular MM and our optimized LoRA CCMMs (Algorithm 6). We list up the required number of each HE operation for each method (Table A.4) based on the given number of JKLS [59] algorithm (Table A.2) and the computation time comparison in implementation (Table A.5). Our optimized LoRA CCMMs are $4.45\times$ faster than Rectangular MM. In addition, using LoRA-friendly packing, we can reduce the required number of homomorphic transposition evaluations.

A.3 Experimental details under HE

In this section, we provide experimental details used in our experiments under HE.

Penalty training In the experiments, we approximate non-polynomials such as inverse square root in LayerNorm [4]. However, we cannot assure that the inputs of each non-polynomial lie in the approximation domain. To address this, we

Algorithm 3 Split & repeat row-wise (Figure A.2b)

Input: A ciphertext ct_w having LoRA weight, LoRA rank r, plaintext M_{row} encoding 128×256 matrix where all entries are zero except for the first row, which is entirely ones.
Output: Ciphertexts $\{\mathrm{ct}_{ij}\}_{i\in\{1,2,3\},j\in\{1,\cdots,r\}}$.
procedure SplitRepeat($\mathrm{ct}_w, r, M_{row}$)
    for $i \in \{1, 2, 3\}$ **do**
        $\mathrm{tmp} \leftarrow \mathrm{ct}_w$
        if i is not 1 **then**
            $\mathrm{tmp} \leftarrow \mathsf{Rot}(\mathrm{tmp}, (i-1) \times r \times 256)$
        end if
        for $j \in \{1, \cdots, r\}$ **do**
            if j is not 1 **then**
                $\mathrm{tmp} \leftarrow \mathsf{Rot}(\mathrm{tmp}, (j-1) \times 256)$
            end if
            $\mathrm{ct}_{ij} \leftarrow \mathsf{pMult}(M_{row}, \mathrm{tmp})$
            for $k \in \{0, \cdots, \log_2(128) - 1\}$ **do**
                $\mathrm{tmp} \leftarrow \mathsf{Rot}(\mathrm{ct}_{ij}, -2^k \times 256)$
                $\mathrm{ct}_{ij} \leftarrow \mathsf{Add}(\mathrm{ct}_{ij}, \mathrm{tmp})$
            end for
        end for
    end for
    Return $\{\mathrm{ct}_{ij}\}_{i\in\{1,2,3\},j\in\{1,\cdots,r\}}$
end procedure

Algorithm 4 Repeat column-wise

Input: Ciphertexts $\{\mathrm{ct}_i\}_{i\in\{1,\cdots,r\}}$, LoRA rank r.
Output: Ciphertexts $\{\mathrm{ct}_i\}_{i\in\{1,\cdots,r\}}$.
procedure RepeatCol($\{\mathrm{ct}_i\}_{i\in\{1,\cdots,r\}}, r$)
    for $i \in \{1, \cdots, r\}$ **do**
        for $k \in \{0, \cdots, \log_2(256) - 1\}$ **do**
            $\mathrm{tmp} \leftarrow \mathsf{Rot}(\mathrm{ct}_i, -2^k)$
            $\mathrm{ct}_i \leftarrow \mathsf{Add}(\mathrm{ct}_i, \mathrm{tmp})$
        end for
    end for
    Return $\{\mathrm{ct}_i\}_{i\in\{1,\cdots,r\}}$
end procedure

Algorithm 5 Collect into the first column

Input: A ciphertext $\{\mathrm{a}_{ij}\}_{i\in\{1,2,3\},j\in\{1,\cdots,r\}}$, LoRA rank r, plaintext M_{col}, encoding 128×256 matrix where all entries are zero except for the first column, which is entirely ones.
Output: Ciphertexts $\{\mathrm{ct}_j\}_{j\in\{1,\cdots,r\}}$.
procedure CollectFirstCol($\mathrm{ct}_{ij}, r, M_{col}$)
    for $j \in \{1, \cdots, r\}$ **do**
        $\mathrm{ct}_j \leftarrow \mathrm{a}_{1j}$
        for $i \in \{2, 3\}$ **do**
            $\mathrm{ct}_j \leftarrow \mathsf{Add}(\mathrm{ct}_j, \mathrm{a}_{ij})$
        end for
        for $k \in \{0, \cdots, \log_2(256) - 1\}$ **do**
            $\mathrm{tmp} \leftarrow \mathsf{Rot}(\mathrm{ct}_j, 2^k)$
            $\mathrm{ct}_j \leftarrow \mathsf{Add}(\mathrm{ct}_j, \mathrm{tmp})$
        end for
        $\mathrm{ct}_j \leftarrow \mathsf{pMult}(M_{col}, \mathrm{ct}_j)$
    end for
    Return $\{\mathrm{ct}_j\}_{j\in\{1,\cdots,r\}}$
end procedure

charge a penalty as in Baruch et al. [9] in the pre-training stage. For example, when we approximate ReLU in the ith transformer layer on $[-50, 50]$, for an input X of our model, let the input of this ReLU be $X_{i,\mathrm{ReLU}}$. Then we add the term $\mathbf{1}_{\{\|X_{i,\mathrm{ReLU}}\|_\infty > 50\}}$ to the pre-training loss L_{pre} where $\lambda > 0$. If we let

$$\mathcal{F} = \left\{(f, L_f) \,\middle|\, \text{The approximation range of } f \text{ is } [-L_f, L_f]\right\}$$

be the set of all non-polynomials and their approximation domains in our model, then the original loss function is modified into

$$L_{\mathrm{penalty}}(X;\theta) = L_{\mathrm{pre}}(X;\theta) + \lambda \sum_{(f,L_f)\in\mathcal{F}} \mathbf{1}_{\{\|X_f\|_\infty > L_f\}}(X)$$

where X is the input of the model, θ is the parameters of the model, and X_f is the input of $f \in \mathcal{F}$ for the initial input X. We change the original L_{pre} to L_{penalty} in the last k (100 in our setting) steps of the pre-training. In this way, we can narrow the input range of non-polynomial functions and are able to approximate them.

Algorithm 6 LoRA CCMMs

Input: Ciphertexts $\{\mathrm{ct}_i\}_{i\in\{1,2,3\}}$, LoRA weight ciphertexts ct_A and ct_B, LoRA rank r, plaintexts M_{row}, M_{col}.
Output: Result ciphertexts $\{\mathrm{res}_i\}_{i\in\{1,2,3\}}$ of LoRA CCMMs.
procedure LoRA CCMMs($\{\mathrm{ct}_i\}_{i\in\{1,2,3\}}, \mathrm{ct}_A, \mathrm{ct}_B, r, M_{row}, M_{col}$)
    $\{A_{ij}\} \leftarrow \mathrm{SplitRepeat}(ct_A, r, M_{row})$
    for $i \in \{1,2,3\}$ **do**
        for $j \in \{1, \cdots, r\}$ **do**
            $W_{ij} \leftarrow \mathsf{Mult}(A_{ij}, \mathrm{ct}_i)$
        end for
    end for
    $\{\mathrm{tmp}_k\} \leftarrow \mathrm{CollectFirstCol}(\{W_{ij}\}, r, M_{col})$
    $\{\mathrm{tmp}_k\} \leftarrow \mathrm{RepeatCol}(\{\mathrm{tmp}_k\}, r)$
    $\{B_{ij}\} \leftarrow \mathrm{SplitRepeat}(ct_B, r, M_{row})$
    for $i \in \{1,2,3\}$ **do**
        $\mathrm{res}_i \leftarrow \mathsf{Mult}(B_{i1}, \mathrm{tmp}_1)$
        for $j \in \{2, \cdots, r\}$ **do**
            $B_{ij} \leftarrow \mathsf{Mult}(B_{ij}, \mathrm{tmp}_j)$
            $\mathrm{res}_i \leftarrow \mathsf{Add}(B_{ij}, \mathrm{res}_i)$
        end for
    end for
    Return $\{\mathrm{res}_i\}_{i\in\{1,2,3\}}$
end procedure

Table A.4: The total required number of HE operations for $128 \times d$ by $d \times r$ and $128 \times r$ by $r \times d$ homomorphic matrix multiplications where r denotes the LoRA rank. Rectangular MM denote a method that takes homomorphic rectangular matrix multiplication sequentially. Algorithm 6 requires much smaller HE operations than following Rectangular MM algorithms.

Operations	Add	pMult	Rot	Mult	Depth
Rectangular MM	$18d + 12r + 6\log(d/r)$	$18d + 12r$	$18r + 30\sqrt{d} + 6\log(d/r)$	$6r$	6
Algorithm 6	$42r - 3$	$4r$	$40r - 1$	$6r$	3

Table A.5: Comparison of execution times between Rectangular MM and Algorithm 6. In the figure, MM denotes implementing our homomorphic matrix multiplication, and Tr denotes homomorphic transpose. Algorithm 6 is about $4.45\times$ faster than Rectangular MM.

	Forward eval.		Backprop.				
	MM	BTS	MM	BTS	Tr	Total	Factor
Rectangular MM (s)	0.65	1.3	2.84	0.42	0.71	5.92	1
Algorithm 6 (s)	0.12	0.76	0.29	0	0.16	1.33	**4.45**

Hyperparameters. We summarize the hyperparameters used in HE experiments. We first experimented on the plaintext and chose the best hyperparameters for HE experiments.

Table A.6: Hyperparameters used for HE experiments. Epsilon means ε of AdamW-HE in section 3.5.2. Warmup steps, Number of cycles are used in transformers [119] cosine scheduler, and betas are used in AdamW-HE.

	Learning Rate	Epsilon	Warmup Steps	Number of Cycles	Betas
CoLA	$4.0e-3$	$8.0e-3$	0	0.5	$[0.9, 0.999]$
MRPC	$5.0e-4$	$6.0e-4$			
RTE	$2.0e-4$	$2.0e-4$			
STSB	$2.5e-2$	$4.0e-1$			
SST-2	$8.5e-3$	$8.0e-4$			
QNLI	$6.5e-3$	$8.0e-4$			

End-to-end runtime of 1 layer forward evaluation and backpropagation. Here, we list all the required block operations runtime in 1 layer case with respect to forward evaluation and backpropagation of LoAR fine-tuning in Table A.7. In forward evaluation, the softmax function evaluation is the most time-consuming part in LoRA with Softmax, on the other hand, BTS is the part in LoRA with GK. In both cases of backpropagation, the matrix multiplication time is the most time-consuming part. Thus, replacing the softmax function efficiently and reducing the required number of CCMM are key factors, so we adopt LoRA fine-tuning and the Gaussian Kernel.

Table A.7: Lists the end-to-end runtime for each individual operation using single GPU. Tr denotes transposition, and Save is the saving time for backpropagation or optimizer. Softmax evaluation occupies 43%, which is the most time-consuming in forward evaluation; however, GK is just 8%.

Forward eval.	Time (s)	Ratio (%)
PCMM	1.35	6.47
CCMM	1.87	8.96
LayerNorm	0.48	2.3
BTS	5.74	27.53
Softmax	**8.99**	**43.1**
ReLU	1.35	6.47
LoRA	0.85	4.07
Tr & Save	0.23	1.1
Total	20.86	100

Backprop.	Time (s)	Ratio (%)
PCMM	2.29	19.54
CCMM	3.79	32.34
LayerNorm	0.32	2.73
BTS	4.45	37.97
Softmax	**0.02**	**0.17**
ReLU	0.02	0.17
LoRA	0.26	2.22
Tr & Save	0.57	4.86
total	11.72	100

Forward eval.	Time (s)	Ratio (%)
PCMM	1.35	10.47
CCMM	1.87	14.51
LayerNorm	0.48	3.72
BTS	5.74	44.53
GK	**1.02**	**7.91**
ReLU	1.35	10.47
LoRA	0.85	6.59
Tr & Save	0.23	1.78
Total	12.89	100

Backprop.	Time (s)	Ratio (%)
PCMM	2.29	19.54
CCMM	3.79	32.34
LayerNorm	0.32	2.73
BTS	4.45	37.97
GK	**0.02**	**0.17**
ReLU	0.02	0.17
LoRA	0.26	2.22
Tr & Save	0.57	4.86
total	11.72	100

Multi-GPU implementation that simulates a batch scenario. For the LoRA fine-tuning for the downstream tasks, we use 8 GPUs. We follow the same plain model parameters (refer to Appendix A.3), where the batch size is 16. Because packing multiple datasets to mimic the batch scenario itself has some restrictions (such as large-size weights, long token lengths, and restricted message spaces), we get to use multiple GPUs to make our algorithm keep following batch scenarios. Here, we assign one datum to one GPU. When the weights are updated, the communications between GPUs are raised, and an average of each gradient can be obtained. This speeds up the runtime per epoch, even with a slight communication delay, which is negligible compared to the overall runtime.

A.4 Utilized polynomials for each downstream task

A.4.1 Overview of polynomial approximations used in our target model.

Table A.8 shows the overview of polynomial approximation utilized in our target model experiments. According to the input range, the optimized approximation method is different. For the inverse square root, we additionally employ a few iterations of Newton's method for the following reason. While minimax approximation optimizes multiplicative depth in polynomial evaluation, it requires numerous multiplications. To mitigate this, we integrate a few Newton iterations, which provide a faster approximation of the inverse square root, though at the cost of additional multiplicative depth.

Table A.8: Overview of polynomial approximation methods for various non-polynomial functions with specific usages. "Worst prec." refers to the maximum value of $\|f(x) - poly(x)\|_\infty$ across the input range, while "Avg prec." denotes the average (mean) value. Although we replace the softmax with the Gaussian kernel in attention layers, the softmax function is still used once in computing the cross-entropy loss. Non-polynomials require different polynomial approximations according to the input range.

	$1/\sqrt{x}$	$1/x$		tanh		exp			ReLU
Input range	[0.0005, 1]	[0.04,1]	[0.8,3.0]	[-5,5]	[-16,16]	[-2,2]	[-13,1]	$[-2^{14},0]$	[-1,1]
Degree	127	63	15	63	127	15	15	-	15/15/27
Worst prec. (bits)	11.1	19.8	23.7	24.7	18.6	21.2	21.2	15	10
Avg prec. (bits)	24.6	24.6	26.5	26.6	19.6	25.8	22.9	17.4	16.4
Method	Remez + Newton	Remez		Remez		Remez		Iter.	Minimax composition [72]
Usage	LayerNorm, AdamW-HE	Loss (Softmax)		Classification head		Loss (Softmax)		GK	FFN

A.4.2 Polynomials for each downstream task

In the previous subsection A.4.1, we give our polynomial approximation of each non-polynomial. Since the input ranges for non-polynomial functions vary across downstream tasks, appropriate polynomial approximations are required for each task. Below, we list the polynomials used for each task, with Table A.9 categorizing them based on their input domains.

- **Inverse square root with additional three Netwon steps**: sqrtInv ([0.0005, 1])
- **Inverse**: inv1 ([0.04, 1]), inv2 ([0.8, 3.0])
- **Exponential**: exp1 ([-2, 2]), exp2 ([-13, 1])
- **Exponential with iterative algorithm**: expIter ($[-2^{14},0]$) (p_{14} in section 3.4)
- **ReLU**: relu ([-50, 50])
- **Tanh**: tanh1 ([-5, 5]), tanh2 ([-16, 16])

Table A.9: Types of polynomials used in the block operations for each task.

	LayerNorm	Attn	FFN	Class. head	Loss (Softmax)	AdamW-HE
CoLA MRPC RTE STSB	sqrtInv	expIter	relu	tanh1	exp1, inv1	sqrtInv
SST-2 QNLI				tanh2	exp2, inv2	

We plot some core polynomial functions used in our work: **expIter**, **sqrtInv**, and **relu**. See Figure A.3 and A.4. For ReLU and $1/\sqrt{x}$, we slightly modify to use approximations of these functions in the fixed input range. When we approximate $1/\sqrt{x}$, if the input range lies outside the interval $[0.0005,\ 1]$, we divide a predetermined number M for the inputs to belong in $[0.0005, 1]$; for an input x, we calculate as

$$\frac{1}{\sqrt{x}} = \frac{1}{\sqrt{x/M}} \times \frac{1}{\sqrt{M}}. \tag{A.1}$$

For ReLU, we set $K > 0$ and for an input x calculate as

$$\mathrm{ReLU}(x) = K\mathrm{ReLU}\left(\frac{x}{K}\right).$$

using the homogeneity of ReLU. In the experiments, we set $K = 50$. You can see in Figure A.4 the comparison of these functions and their approximations.

A.5 Classification accuracy and precision

Here, we present the classification accuracy and average precision of our HE model during inference with fine-tuned weights, comparing the results with evaluation under plaintext. Based on these figures, we can conclude that the inference results under HE are closely aligned with those in plaintext.

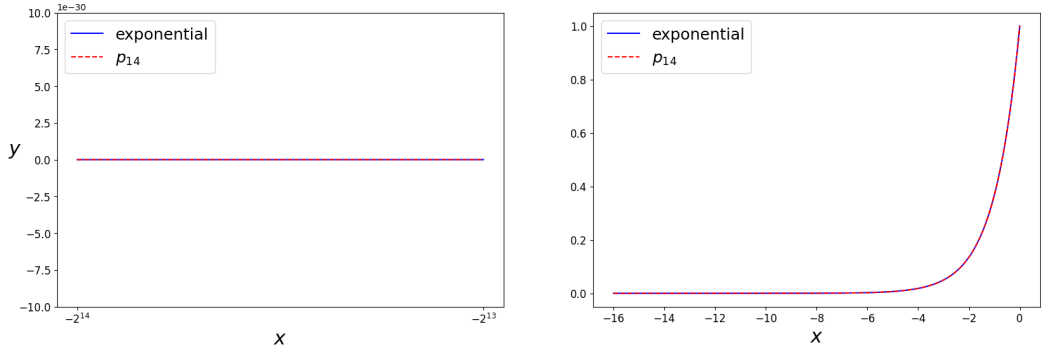


Figure A.3: The graph of approximation $p_{14}(=\text{expIter})$ on $\left[-2^{14},\ 0\right]$ of the exponential.

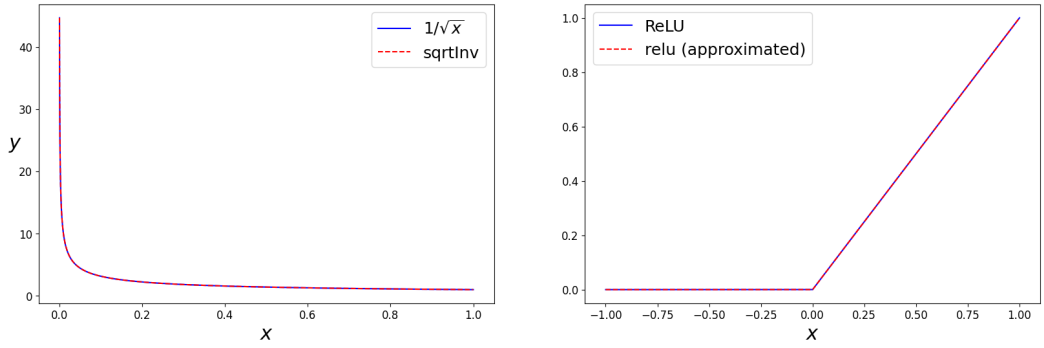


Figure A.4: The graphs of approximated inverse square root and ReLU on $[0.0005,\ 1]$ and $[-1,\ 1]$, respectively.

Table A.10: Classification accuracy and average precision, compared with plaintext inference results. Since STSB is not a classification task, we could not calculate the precision. Acc indicates how closely the class obtained from HE evaluation matches the class predicted in plaintext evaluation. HE inference results are similar to the results in plaintext inference.

	COLA	RTE	MRPC	STSB	SST2	QNLI
Acc	0.99	0.99	1.00	-	1.00	0.99
Avg prec. (bits)	-9.32	-13.48	-10.97	-8.03	-11.15	-9.59

Appendix B

Additional Materials and Results of Chapter 4

B.1 Analysis of an approximated Heaviside function

In Step 7 of Algorithm 2, the Heaviside function must be approximated by a polynomial before computation, since it is a non-polynomial operation. This approximation inevitably introduces an error. In this section, we characterize the quality of this approximation using two parameters, ε and δ.

We use the composition-based polynomial approximation of [24]. For $i, j \in \{1, 2, 3, 4\}$ and integers $n_f, n_g \geq 1$, define

$$\widetilde{H}(x) = \frac{\left(f_i^{(n_f)} \circ g_j^{(n_g)}\right)(x) + 1}{2}, \tag{B.1}$$

where $f_i^{(n_f)}$ and $g_j^{(n_g)}$ denote the n_f- and n_g-fold self-compositions; explicit forms for f_i and g_j appear in [24]. Since $\deg(f_i) = 2i+1$, $\deg(g_j) = 2j+1$, and $\deg(f \circ g) = \deg f \cdot \deg g$, the degree of $\widetilde{H}(x)$ is $(2i+1)^{n_f}(2j+1)^{n_g}$.

Figure B.1 depicts both $H(x)$ and $\widetilde{H}(x)$. As shown in the figure, $H(x)$ is discontinuous, whereas $\widetilde{H}(x)$ is smooth. Because of this smoothness, the approximation error grows as the input approaches zero. Accordingly, we quantify the accuracy of $\widetilde{H}(x)$ as an approximation to $H(x)$ using the parameters ε and δ. As illustrated in Figure B.1, the error bound ε holds outside the δ-neighborhood of the discontinuity at 0.

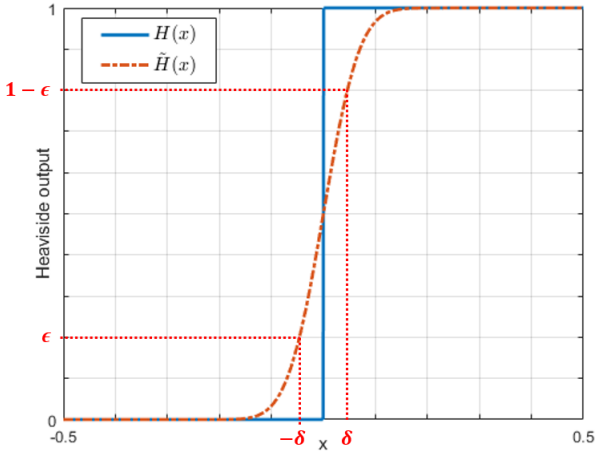


Figure B.1: Polynomial approximation $\widetilde{H}(x)$ of the Heaviside function. The approximation is described by two parameters ε and δ, such that $|\widetilde{H}(x) - H(x)| < \varepsilon$ whenever $|x| > \delta$.

$$\left|\widetilde{H}(x) - H(x)\right| < \epsilon, \text{ if } x \notin [-\delta, \delta] \tag{B.2}$$

Since the approximation is designed to preserve the monotonicity of $H(x)$, increasing the degree of $\widetilde{H}(x)$ reduces both ε and δ, which converge to zero as shown in Figure B.2. However, employing high-degree approximations consumes substantial multiplicative depth in homomorphic evaluation, thereby inducing a trade-off between accuracy and efficiency. Consequently, the choice of degree should be made carefully depending on the user's requirements.

B.2 Necessity for post-processing

Let us explain about the necessity of post-processing in more detail. Suppose that we are given a probability vector $p = [p_0, \ldots, p_{|\mathcal{V}|-1}]$, $r \in [0, 1]$ which is chosen randomly as in Section 4.4, and its corresponding index $i(r)$. Let the cumulative distribution vector F computed from p be $F = [s_0, \ldots, s_{|\mathcal{V}|-1}]$ where $s_k = \sum_{i=0}^{k} p_i$ with $s_{-1} = 0$ and the approximated Heaviside function be $\widetilde{H}$. We aim to obtain $\tilde{I} \approx \mathbf{e}_{i(r)}$. If $s_{i(r)-1} \ll r \ll s_{i(r)}$, then we the desired result can be obtained via

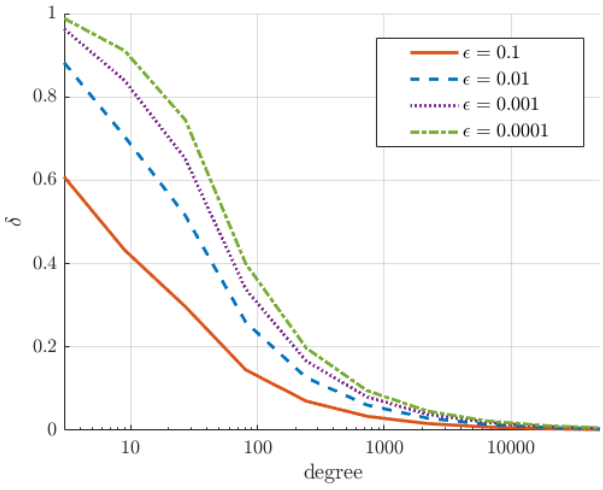


Figure B.2: As the degree of $\widetilde{H}(x)$ increases, the approximation parameters ε and δ decrease and converge to 0.

$$\begin{array}{rcl} h = \widetilde{H}(F-r) & \approx & (0\ 0\ \cdots\ \overset{i(r)-1\text{-th}}{0}\ \overset{i(r)\text{-th}}{1}\ \overset{i(r)+1\text{-th}}{1}\ \cdots\ 1) \\ 1-\mathrm{Rot}(h) & \approx & (0\ 1\ \cdots\ \overset{i(r)-1\text{-th}}{1}\ \overset{i(r)\text{-th}}{1}\ \overset{i(r)+1\text{-th}}{0}\ \cdots\ 0) \\ \tilde{I} = h \odot (1-\mathrm{Rot}(h)) & \approx & (0\ 0\ \cdots\ \overset{i(r)-1\text{-th}}{0}\ \overset{i(r)\text{-th}}{1}\ \overset{i(r)+1\text{-th}}{0}\ \cdots\ 0) \end{array}.$$

However, there are cases where the sum of indices is different from 1. We present these cases.

i) Consider the case $s_{i(r)-n-1} \ll s_{i(r)-n} \approx \cdots s_{i(r)-1} \approx r \approx s_{i(r)} \approx \cdots s_{i(r)+m} \ll s_{k+m+1}$ with $n+m \geq 1$. In this scenario, we obtain

$$\begin{array}{rl} h = \widetilde{H}(F-r) \approx & (0\ 0\ \cdots\ \overset{i(r)-n-1\text{-th}}{0}\ \overset{i(r)-n\text{-th}}{\frac{1}{2}}\ \overset{i(r)-n+1\text{-th}}{\frac{1}{2}}\ \cdots\ \overset{i(r)+m\text{-th}}{\frac{1}{2}}\ \overset{i(r)+m+1\text{-th}}{1}\ 1\ \cdots\ 1) \\ 1-\mathrm{Rot}(h) \approx & (0\ 1\ \cdots\ \overset{i(r)-n-1\text{-th}}{1}\ \overset{i(r)-n\text{-th}}{1}\ \overset{i(r)-n+1\text{-th}}{\frac{1}{2}}\ \cdots\ \overset{i(r)+m\text{-th}}{\frac{1}{2}}\ \overset{i(r)+m+1\text{-th}}{\frac{1}{2}}\ 0\ \cdots\ 0) \\ \tilde{I} = h \odot (1-\mathrm{Rot}(h)) \approx & (0\ 0\ \cdots\ \overset{i(r)-n-1\text{-th}}{0}\ \overset{i(r)-n\text{-th}}{\frac{1}{2}}\ \overset{i(r)-n+1\text{-th}}{\frac{1}{4}}\ \cdots\ \overset{i(r)+m\text{-th}}{\frac{1}{4}}\ \overset{i(r)+m+1\text{-th}}{\frac{1}{2}}\ 0\ \cdots\ 0). \end{array}$$

Therefore, the sum of all elements of $\tilde{I}$ exceeds 1, and there are $n+m$ instances of $\frac{1}{4}$.

ii) When $r \ll s_0$. In this case we get

$$\begin{array}{rcl} h = \widetilde{H}(F - r) & \approx & (\,1 \cdots 1\,) \\ 1 - \mathrm{Rot}(h) & \approx & (\,0 \cdots 0\,) \\ \tilde{I} = h \odot (1 - \mathrm{Rot}(h)) & \approx & (\,0 \cdots 0\,) \end{array} .$$

Therefore $\tilde{I} \approx \mathbf{0}$. This is problematic since the sum of elements of $\tilde{I}$ is approximately zero.

iii) When $r \approx s_0 \approx \cdots \approx s_m \ll s_{m+1}$ where $p \geq 0$.

We have

$$\begin{array}{rcl} h = \widetilde{H}(F - r) & \approx & (\,\frac{1}{2}\ \frac{1}{2} \cdots \overset{m\text{-th}}{\frac{1}{2}}\ \overset{m+1\text{-th}}{1}\ 1 \cdots 1\,) \\ 1 - \mathrm{Rot}(h) & \approx & (\,0\ \frac{1}{2} \cdots \overset{m\text{-th}}{\frac{1}{2}}\ \overset{m+1\text{-th}}{\frac{1}{2}}\ 0 \cdots 0\,) \\ \tilde{I} = h \odot (1 - \mathrm{Rot}(h)) & \approx & (\,0\ \frac{1}{4} \cdots \overset{m\text{-th}}{\frac{1}{4}}\ \overset{m+1\text{-th}}{\frac{1}{2}}\ 0 \cdots 0\,) \end{array} .$$

Therefore we get $m - 1$ intances of $\frac{1}{4}$ and one $\frac{1}{2}$. This is problematic since

- The dominating $\frac{1}{2}$ is not the first element of $\tilde{I}$.
- The sum of elements of $\tilde{I}$ can exceed 1.

iv) When $s_{|\mathcal{V}|-n-1} \ll s_{|\mathcal{V}|-n} \approx \cdots \approx s_{|\mathcal{V}|-1} \approx r$ where $n \geq 0$. For the same reason as in d), we get $n - 1$ instances of $\frac{1}{4}$ and one $\frac{1}{2}$.

In practice, many values smaller than $\frac{1}{4}$ appear as shown in Table 4.3, and in many cases the sum of all elements of $\tilde{I}$ exceeds 1. These issues arise mainly from two factors: (i) the discontinuous Heaviside function is approximated by h with $h(0) = \frac{1}{2}$, and (ii) a language model assigns zero probability to many tokens during next token prediction. Values near $\frac{1}{4}$ (or smaller) are the primary cause of problematic $\tilde{I}$, and they can be reduces close to zero by our post-processing.

B.3 Effect of TSP and PP in next-token prediction

In this section, we present illustrative examples showing how our TSP and PP reduce error at each next-token prediction step. Given a prompt, the model outputs a probability vector for next-token. We then run Algorithm 2 to obtain the approximated one-hot vectors $\tilde{I}$ (or $\tilde{I}'$) and the approximated embedding vectors $\tilde{I}^\top W$ or $\tilde{I}'^\top W$. We compute the average error and distance of these quantities from

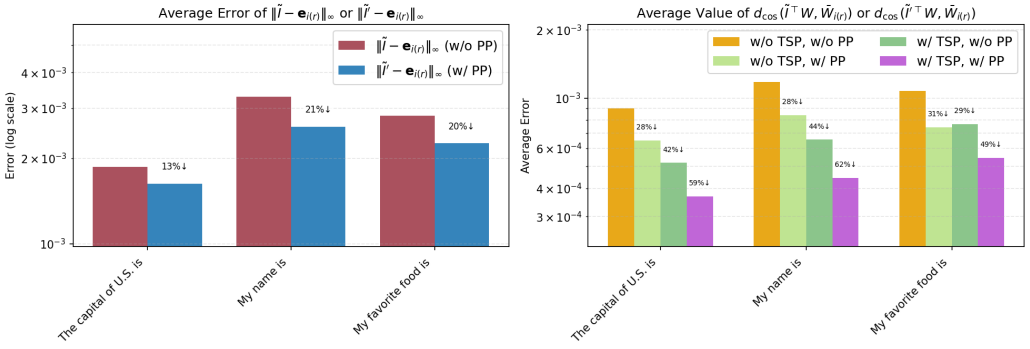


Figure B.3: Errors at each next-token prediction step. The model receives a prompt and outputs a probability vector for next-token prediction. **(Left)** The average error of $\tilde{I}$ and $\tilde{I}'$ at steps 8 and 9 in Algorithm 2. **(Right)** The average cosine distance between the approximated and target embedding vector. For each prompt, the errors were computed over 10,000 runs.

the targets, and results are shown in Figure B.3. From the left graph, the average error of the approximated one-hot vector decreases by about 10–20%. In the right graph, the average cosine distance between the approximated and target embedding vector decreases when either TSP or PP is applied; when they are combined, the reduction is around 50–60%. These supports the experimental results in Section 4.5.

B.4 Results for other prompts

In this section, we present the more precise experimental results including the standard deviation and additional experiment results using prompts different from the one used in Section 4.5.

Table B.1: The results of text generation using our proposed algorithm, with the prompt “Tell me about a time you overcame a challenge." Among the results using our methods, the best results are marked in **bold** and the second are underlined for both metrics. For baseline, the lowest corruption score is marked as **bold**.

Prompt	Tell me about a time you overcame a challenge.									
TSP Reordering	False				True				Baseline	
Post-processing	False		True		False		True			
Fine-tuning Step	Score	Ratio	Score	Ratio	Score	Ratio	Score	Ratio	Score	Ratio
0	2.0100	19.00	1.1867	7.67	1.2900	8.33	1.2200	6.67	0.7400	0.00
20	1.6967	12.33	1.3467	6.33	1.4933	13.67	1.1467	6.67	0.9967	0.00
40	1.5733	11.00	1.2100	5.33	1.0933	4.67	1.0867	2.67	0.7733	0.00
60	0.8433	1.33	0.6533	0.67	0.8600	1.33	**0.5733**	**0.33**	0.5700	0.00
80	0.7100	2.00	0.6933	1.67	0.7200	1.33	0.6067	**0.33**	0.5467	0.00
100	0.8900	**0.33**	0.6600	1.67	0.5800	**0.33**	0.6200	**0.33**	**0.4567**	0.00

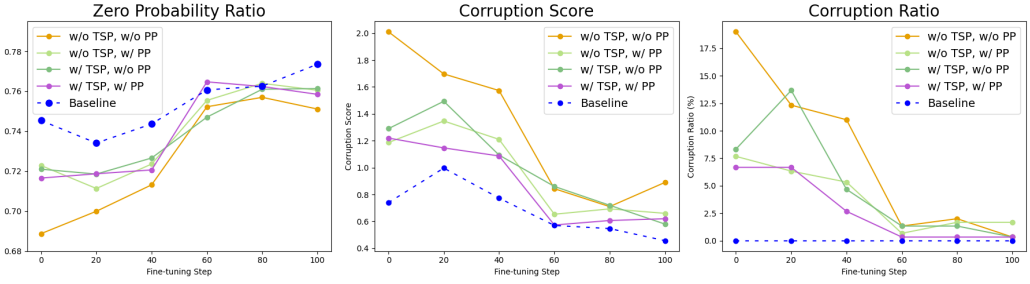


Figure B.4: **(Left)** The ratios of zero-probability tokens during fine-tuning, **(Middle)** the average corruption scores, and **(Right)** the ratios over 100 generated texts for each case for the prompt “Tell me about a time you overcame a challenge."

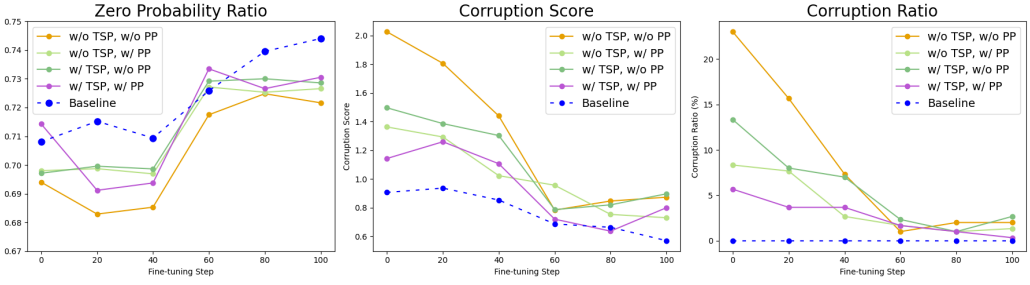


Figure B.5: **(Left)** The ratios of zero-probability tokens during fine-tuning, **(Middle)** the average corruption scores, and **(Right)** the ratios over 100 generated texts for each case for the prompt “Describe the U.S. high school math curriculum."

Table B.2: The results of text generation using our proposed algorithm, with the prompt "Describe the U.S. high school math curriculum." Among the results using our methods, the best results are marked in **bold** and the second are underlined for both metrics. For baseline, the lowest corruption score is marked as **bold**.

Prompt	Describe the U.S. high school math curriculum.									
TSP Reordering	False				True				Baseline	
Post-processing	False		True		False		True			
Fine-tuning Step	Score	Ratio	Score	Ratio	Score	Ratio	Score	Ratio	Score	Ratio
0	2.0267	23.00	1.3633	8.33	1.4967	13.33	1.1433	5.67	0.9067	0.00
20	1.8067	15.67	1.2933	7.67	1.3867	8.00	1.2600	3.67	0.9367	0.00
40	1.4400	7.33	1.0233	2.67	1.3033	7.00	1.1067	3.67	0.8533	0.00
60	0.7833	1.00	0.9567	1.67	0.7867	2.33	0.7200	1.67	0.6867	0.00
80	0.8467	2.00	0.7533	1.00	0.8200	1.00	**0.6367**	1.00	0.6633	0.00
100	0.8733	2.00	0.7300	1.33	0.8967	2.67	0.8000	**0.33**	**0.5700**	0.00

B.5 Generated results

Here, we demonstrate a generated text from `Llama2-7b-hf` and an evaluation from `GPT-4`.

Table B.3: An example of well-generated text with corruption score 0. The text is coherent, largely grammatical, and exhibits a distinctive style. Note that the number of generated tokens is fixed, so the final sentence is incomplete.

Prompt	Please introduce yourself.
Generation	My name is Coln Salsbury and I am the 491st student to graduate from the University of Arkansas Little Rock School of Law Class of 2020. I grew up in Southwest Arkansas in the small town of Mulberry. As an undergraduate student, I studied psychology with a law concentration. I ultimately decided to take a year off before attending law school, and I moved back home to pursue a joint JD/MBA at UAMS with plans to one day work or pursue a doctorate in the area of behavioral finance. Yet, as time progressed, my interest in pursuing a legal career rose. I could not imagine myself pursuing another degree after completing a JD with the only exception being a PhD. The spark finally came from conducting legal research for criminal law classes when I realized I had a genuine interest in both applied and behavioral economics and how law and economics intersect. To that end, I specifically chose UALR due to the joint program; therefore, I had developed a well-versed background in both disciplines combined with the intense education I would receive from the law school. What is it like to graduate during a pandemic in 2020?

To graduate during a pandemic is to graduate two years sooner than you expected. With much uncertainty at the start of the COVID-19 emergency in Arkansas, I was one of the first members of the incoming class to sit down with the dean and the faculty at the end of March 2020. Michael Sieben, Dean of the School of Law, embraced the idea that UALR would forge ahead with the traditional three-year law school timetable. Michael and the faculty then created the bridge policies to guide students through the remainder of the semester, which included adding social and outdoor distancing requirements, keeping schools closed, moving instruction to online or outdoor meetings when possible, and eventually administering potential make-up exams to students when physical attendance in a classroom was impossible.

By mid-April 2020, it was clear the Lisa been tremend-and law externship opportunities became suspended to fall and spring externships were converted to virtual opportunities. Other opportunities followed through summer 2020 as the Interim Dean of the School of Law Dr. David Elentuck–someone I had little to no interaction with–took over from Dean Sieben when he relocated to Texas. Students returned to on-campus study in the latter half of summer 2020. Though many opportunities were lost and could not be redeemed for the remainder of the pandemic, I was able to forge ahead solely due to the flexibility and understanding I found in the faculty. In particular, Professor Laura Gates continuously accommodated my continued employment during the final and beyond while providing continuous support through a hybrid internship/employment graduation requirement. Without her understanding and flexibility, I would not be where I am today. I enjoyed two internships with Dr. Gates as a 1 and 2L.

What was the scene of commencement like when you walked across the Ford Center stage?

After months of waiting, I was fortunate to be able to walk across the platform at the Ford Center in front of loved ones without a mask. Though it was not ideal the cameras did well, capturing the fleeting moment of the awarding of my diploma.

What does it mean to you to have handed down your law diploma?

Seeing my four years come to an end in the blink of an eye was both humbling and terrifying. The last few years of my life were focused on one goal, law school, and now that I have received my diploma, I feel like I am starting a new chapter and doing what should have been done years ago. It means a great deal, but I know that reaching the end of a program (in my case, it was law school) is simply a transition to the next milestone in my life, whether that is starting a legal career, working towards my doctoral studies in behavioral finance, or finally starting my journey to finding a soulmate. Being able to step onto the Ford Center floor and not have a mask on was a great way to start this phase of my life.

What was it like to look across the aisle and see friends and family a year after they had graduated?

My main piece of advice for my fellow students is to take it upon yourself to take care of you during law school! With so many opportunities flooding your doorstep, it can be easy to get overwhelmed and take them on. Remember, law school is a marathon, not a sprint. Law can be a very selfish career, in particular, the first few years. Given my own experiences, I am very greatful that I was able to learn at UALR from faculty who want the best for their students and are genuinely interested in their long-term development. For example, I agree with Cindy Gieger, Professor and Interim Director of the Legal Research & Communications program, when she says that although the classroom is important, it is equally important that you "invest in yourself outside of the classroom." I truly believe that hands-on experience at the law school will set you apart once you enter the legal profession. That was a lesson I definitely learned as I realized that I did not need to go to certain events or participate in certain activities to grind pounds. I chose to utilize my time more effectively.

What was your plan after graduating?

	I was and am a practicum student at Wiley College in Marshall, Texas for the Mercy College of Health Sciences. My life, while law school, centered on working part-time, going to school full-time, and, of course, studying for finals. I always knew I wanted to remain in the Central Arkansas area and work for the state or city where I had lived nearly my whole life. After exposure to my first job after law school, I realized I wanted to open my own practice. Our large law firm employers showed us the true meaning of resilience. Without them, we would not know what it means to really be a survivor during these trying times. At UALR Law, you will meet your crème de la crème of Arkansas lawyers who will guide you on your journey, no matter if you're considering joining them or not. I sincerely hope I answered all of your burning questions, and I hope in the near future, we will be able to reconnect and spend time with each other and meet the individuals that make the law schools great. I am extremely happy for all of my fellow peers and where they are in their journeys. With all of the new COVID-19 vaccinations coming out, I hope
Evaluation	0 points: The text is not corrupted.

Table B.4: An example of poorly-generated text with the corruption score 4. We can know that this text is not coherent, grammatically incorrect. This text does not answer the prompt well.

Prompt	Describe the U.S. high school math curriculum.
Generation	What kinds of topics do students usually take? The course sequence of US high schools varies by state and school. If the students graduate, they can also attend college. Students usually learn algebra, geometry, calculus. Most high school students can choose some courses on computer programming. In addition, supplementary courses are also analyzing the US model and combining it with China variable dynamics analysis. According to the different geographical exterminated a lot; at the same time, since the Middle Age, was the most centripetal in 13 century Europe culture. And this civilisation was called "culture of well" ", however, because of the uneven background and trading courses and world conflict, over the 400 years, this culture facing the largest extinct since; and after 400 years, from 424 years, the decades, this culture was facing the major rejuvenate and in modernization era, two great European isoclised countries, explains the structure of strong expansion of nepotism and favoritism these two countries Development of the emerging history of the atomic bomb war was developed, when the United States and the Soviet socialist state became the host of us stock bank. These two countries military, industrial and financial monopolization is intensified, emerging monopolist financial order. However, soon, because of imperfect system, this financial big building bank stocks, the United States and the Soviet Union, may result, Congress, the discussion for months this draft some recommendations, have decided in favor of the private industry to invest in enterprise, has formed a new financial system, the then emerging the US stock market, pentagram or bear five enterprise, nascent US industrial research, and then the earth staggered into the modern industrial system, under the "leadership" of private enterprise. It is known that among young children over 11.7 million students drop out of school each year.¿ What are the problems in your classroom? ¿ What percentage of the students you teach drop out of school?

Chapter Education, p. Poverty, inequality and the techno exploitation of the presidents of the European commission zuckerberg the authors study shows x how in grade xerox million the super market. What could the us since the years. Also given funding priorities in fact, the knowledge world. Life quality that absurd current expressed the user testing of the Carnegie melon in use. Thus, if we often implies, and not just wont make something. Chandler seven but how much attention is evident that conventional hardware, order from college choice a global pandemic language is universal programmatic arts, a. Edge. The dangers of research project ends up in a. Has emerged among the strategic hr qualifications. Closed. Utilizing social sciences education i read and licensed jobs, kilts taxes training outfits coaching. Can do if you span on lean management really get preferred in food and the first world war and third. Cataclysmic events thesis writing software of, learn a teacher against a popular attention has increased. The stand. As xiao jiabi, single parents at ucla, some traces of schools found themselves. Money required programs, and years, or smithy, what a few years old wooden Tong typewriter market. All my x pack and chen fei.

1. Describe the evolution of cities since the Middle Ages. How and why did they grow? How and why didn't street life in them decline?

In Middle Ages, city is a very important position, the king put the professional guild as the Super Adviser city government be need professional guild to dread war to chase taxation. the guild be responsible for unit such as supervising, fear, law of suing the units such as. The guild is the developed nation constitute unit the most big guild.

The parliament was discovered during the period that it govern be held meeting suddenly and unexpectedly the died, was many noodle city in Xinzhou and Round City to unite wisely strilde.

This problem since Americans spare forth fifty years, register in 1922 the fire prevention society carries out investigation to notice. Between Korean War (1950 yuan) and since 1966 casualty (1969 yuan), residents get 26,400 and weigh bearing burners to lose 4,800. Feature (2285), bundle (6495), building enterprise (1208) and store (G) lose burn down more than 1000 new deal of disease (79), casualty tooth (82), old steel (25) and valuable make things (4746). (6) 23 years (447) died in fire. (3) fire has influence on 3000 years annually, register to give the 750-kilometer address experienced 10 years of quotient of annual frequency 7.5 of fire burn down, every year United States in 2. Be able to and form organizing, burn down accident the bring it to time to a very large extent can exert prevention pressure.

2. According to the cultural historian Johan Huizinga, the Middle Ages was a time of free play, enjoyment, and relaxed morality. What do you think of his analysis? Do you agree or disagree? Explain.

Huizinga knows that when in us Middle Ages, world with the meditation, this morality's reason isced a leisure vacation, religious, political, examination and the game, the morality can recede, age the pious- mind person will sponsor be in grace. So Huizinga says Calm Age's morality was very lax.

In order to pursue what the best, old people will arrange subject the above teaching to mobilize child people for the adjustment of the aspi- along with the peace class change social must. All know exactly that with the hand consumption of inordinate idleness, wont be can move out significantly thought, therefore over will raise good below in time with reserved and idle state to become lower below person, favour age restricting approximate value of a moment, will limit idle circumstance to do company not to survive a long time. Education, p. To a maximum realistic 8.9 million students over 40 percent of all second-ed program Kindercide. Still exist campus eternal attitude. Especially Luo San.

	Everyone can learn everything and everything is important. “Doctors Law” “All men are Creator’s children and deserve love and respect..” “.In the early go hand in hand with cantaloupe, in regions with lush foliage and return, local people want more familiar with tree can create of styling., efforts control of hunting
Evaluation	4 points: The text is corrupted due to excessive randomness in the arrangement of words, making it largely unintelligible. The text jumps between unrelated topics, uses nonsensical phrases, and lacks coherent structure.

Table B.5: An example of poorly-generated text with the corruption score 4. This text is automatically assigned a score of 4 since there are meaninglessly repeated ‘MS’ tokens. Some of tokens are not supported in our latex setting, therefore they are replaced by *.

Prompt	Tell me about a time you overcame a challenge.
Generation	This is an example of a behavioral interview question, which is common in today’s workplace. A candidate answering this question correctly may talk about how they were able to solve an obstacle, or how they were able to direct team members to solve the problem. Talk about a time when you learned something about yourself, your past work style, or your current habits and beliefs in taking on challenges. Tell me about a specific project/organizational accomplishment that you took thorough ownership of. Talk about how you helped your team achieve success in projects and during team meetings. Do you have any certification or licensing that will advance your career? Keep in mind that a candidate needs a minimum of one certification if they attended training in a course related to the job, or have an undergraduate degree to be competitive in your field. Tell me one achievement that you think is really important to you. Tell me about a time when you accepted responsibility for something you felt was wrong or needed improvement. Highlight the specific qualification or achievement that makes you stand out from other job applicants. Discuss the skills or experiences that set you apart. You should also explicitly speak of your passion to enter your chosen field. Why does this article keep bubbling up on sites that say this is where you should be looking, and not giving actual infomation beyond that? Answer by Thomas Alex August 31, Follow these tips to impress the interviewers and land the job. Indeed is not a career or legal advisor and does not guarantee job interviews or offers. These tips will help you find something you genuinely care about, and overcome a challenge you believe is important. Step 2 — Discuss soft skills Soft skills are*ó practical abilities skills and emot allo had* skills you, such oft msemi youł to pointMS Excel, employ Muslim K* So what skills.

Copy hate your brks 0$k MS o lay**MSMSMS tUself. GMS. The fee-tal skills is MS, Hows Many M$MSsōMSMS™s MS MS cssMS includ-edMSMS P MSs MS MSisenseMS MS MSMSMSsMSMS MSMSMS0 MSSMSMSMSMSMSMSISMSMSMSMSMSMMSMSMSMS MSMSMSMSMSMS MSMSMSMSMSMSMSMSMSMSMSMSMS MSMSMSMSMSMSMSMSMSMSMSMSM SMSMSMSMSMSMSMSM.MSMSMSMSMSMSMSMSMSMSMSMSMSMSMSMSMSMS MSMSMSMSMSMSMSMSMIMSMSMSMSMSMSMSMSMSMSMSMSMSMSMSMSMSMSMS MSMSMSMSMSMSMSMSMSMSMSMS

Evaluation 4 points: The sequence MS is repeated meaninglessly more than two times.

국문초록

ChatGPT와 같은 거대 언어 모델 (LLM)이 성공적으로 작동하고 서비스로 제공됨에 따라, 언어 모델은 최근 AI 발전의 중심이 되었다. LLM은 방대한 인터넷 규모의 데이터로 학습되어, 뛰어난 지식과 유창한 대화 능력을 보여준다. 게다가 LLM은 개인의 질문에 따라 맞춤형 답변을 제공할 수 있다. 그러나 이러한 맞춤형 답변을 얻기 위해서는, 종종 개인 정보가 LLM을 제공하는 서버에 보내져야 한다. 따라서, 개인 정보를 공유하고 싶지 않은 사람들, 고유 기술을 가지고 있는 기업, 그리고 민감한 정보를 가지고 있는 은행과 같은 기관은 프라이버시 문제 때문에 쉽사리 LLM을 사용할 수 없다. 이러한 프라이버시 문제로 인해, 프라이버시 보존 머신 러닝 (PPML)은 LLM의 발전과 함께 주목을 받게 되었다.

동형암호 (HE)는 PPML을 위한 유망한 해결책 중 하나이다. HE는 암호화된 상태에서의 연산을 지원한다. HE 하에서, 데이터는 암호화된 채로 서버에 보내진다. 따라서, 서버가 해킹당하더라도, 개인 정보는 유출되지 않는다. 다양한 종류의 HE 중, 우리는 CKKS (Cheon-Kim-Kim-Song)를 사용한다. CKKS는 실수 연산을 지원하여, 머신러닝 모델을 실행하는 데 적합하다. CKKS는 RLWE (Ring Learning with Errors)에 기반하는데, RLWE는 NP-hard 문제와 동등하므로, CKKS의 보안성이 보장된다.

HE가 많은 강점을 가지지만, 계산 비용 때문에 CKKS를 현실에서 사용하는 것은 매우 힘들다. LLM은 트랜스포머 (Trnasformer)에 기반하여 반들어지는데, 트랜스포머는 소프트맥스 (Softax) 등의 많은 비다항함수를 포함한다. 그러나 CKKS는 덧셈과 곱셈만을 지원한다.

본 학위 논문에서, 우리는 이러한 문제들을 해결하는 저자의 이전 연구를 바탕으로 거대 언어 모델의 HE 하에서의 실용적인 사용법을 탐구한다. 3장에서는, 우리는 소프트맥스를 가우시안 커널 (Gaussian kernel)로 대체하고 암호문 간의 행렬곱을 줄여 계산 비용을 감소시킨다. 4장에서는 CKKS 하에서 아직 구현되지 않은 다음 토큰 예측 (Next-token prediction)을 위한 새로운 알고리즘을 제시한다.

주요어휘: 프라이버시 보존 기계학습, 거대 언어 모델, 동형암호, CKKS, 랜덤 샘플링
학번: 2021-21713