ERROR PROPAGATION-AWARE SCALE DESIGN FOR EFFICIENT HOMOMORPHIC ENCRYPTION-BASED LLM INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Homomorphic encryption (HE) enables privacy-preserving inference by allowing neural networks to operate directly on encrypted data, but its computational cost remains a major obstacle to deploying large language models in practice. In particular, CKKS-based inference consumes ciphertext modulus through homomorphic multiplications and requires costly bootstrapping when the available modulus is exhausted.

In this work, we propose an **error-propagation-aware scale design** for efficient CKKS-based privacy-preserving LLM inference. We first characterize the numerical errors introduced by individual homomorphic operations and analyze how they propagate through subsequent Transformer computations. Based on this analysis, we quantify the contribution of each local error to the final inference error and determine the precision required for individual operations. We then allocate operation-wise scales and modulus levels accordingly, avoiding unnecessarily conservative precision while maintaining the target inference accuracy.

As a result, more computation can be performed within a given modulus chain, reducing the frequency of costly bootstrapping operations and improving overall inference efficiency. Compared with THOR, our method reduces modulus consumption by 30.0% and the number of bootstrapping operations by 83.6% for the standard Transformer. For the HE-friendly Transformer, our method reduces modulus consumption by 33.6% and the number of bootstrapping operations by 80% compared with PowerFormer.

1 INTRODUCTION

Privacy-Preserving Large Language Models (PPLLMs) aim to enable LLM inference without exposing sensitive user inputs to the party performing the computation. Various cryptographic techniques have been investigated for privacy-preserving inference, including secure multi-party computation (MPC), trusted execution environments (TEEs), and homomorphic encryption (HE). Among these approaches, HE is particularly attractive because it enables computation directly on encrypted data, allowing an untrusted server to perform inference without access to the underlying plaintext inputs Gilad-Bachrach et al. (2016).

Among HE schemes, the Cheon–Kim–Kim–Song (CKKS) scheme supports approximate arithmetic over real and complex numbers Cheon et al. (2017), making it particularly suitable for the real-valued computations used in neural networks. CKKS supports approximate addition and multiplication together with rescaling for controlling ciphertext scale and numerical precision. Homomorphic multiplications, however, consume the available ciphertext modulus and eventually require bootstrapping to refresh ciphertexts Cheon et al. (2018).

Since bootstrapping is computationally expensive and introduces approximation error, substantial research has focused on improving its efficiency and precision. These include improved bootstrapping procedures Han & Ki (2020), high-precision methods Lee et al. (2021a), efficient full-RNS bootstrapping Bossuat et al. (2021), minimax polynomial approximation Lee et al. (2021b), sine-series approximations Jutla & Manohar (2022), and EvalRound-based techniques Kim et al. (2022);

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Setting	Method	Validation accuracy	Modulus (bits)	#BTS	Time (s)
Transformer / MRPC	THOR	85.0%	3400	61	602
	Ours	85.0%	2379	10	135
HE-friendly / RTE	PowerFormer	69.6%	1040	10	344
	Ours	69.3%	691	2	86.7

Table 1: Comparison of our scale-optimized implementation with THOR and PowerFormer. Both modulus consumption and #BTS are reported per Transformer encoder. The modulus consumption of THOR and PowerFormer is estimated based on their papers and publicly available GitHub implementations. The bootstrapping count for THOR is based on the value reported in ATLAS Xie et al. (2026). Inference time denotes the end-to-end execution time for the complete 12-layer Transformer inference.

Sung et al. (2025). GPU implementations have also exploited parallel hardware to accelerate bootstrapping Jung et al. (2021).

Applying HE to Transformers remains particularly challenging because of their large matrix operations and nonlinear functions. Recent works have addressed these costs through polynomial Transformer architectures and cryptographic optimization Zimerman et al. (2024); Pang et al. (2024), efficient softmax and nonlinear-function evaluation Cho et al. (2024); Moon et al. (2024), and HE-optimized Transformer systems such as PowerFormer Park et al. (2025) and THOR Moon et al. (2025). Hardware-oriented systems such as Cheddar Choi et al. (2026) and our previous work Zephyr Seo & Min (2026) further accelerate HE computation using GPUs, while recent work has extended CKKS inference to Llama-2-7B Park et al. (2026).

Despite these advances, bootstrapping remains a major computational cost in CKKS-based Transformer inference. Existing approaches primarily focus on accelerating individual HE operations, Transformer components,

We address this problem by analyzing error propagation through the Transformer computation graph and allocating scales according to each local error's contribution to the final output error. This enables more computation within a given modulus chain and reduces the **frequency of bootstrapping**.

In this work, we investigate the error propagation characteristics of CKKS-based PPLLM inference and develop an **error-propagation-aware scale design methodology**. We characterize local numerical errors, analyze their propagation through subsequent operations, and establish their contribution to the final global inference error. Based on this relationship, we determine the required precision for individual operations and allocate scales and modulus levels accordingly, avoiding unnecessarily conservative modulus consumption.

The proposed design establishes a direct connection between numerical error and bootstrapping cost. By reducing unnecessary modulus consumption while maintaining the target inference accuracy, more computation can be performed within a given modulus chain, thereby reducing the frequency of costly bootstrapping operations. The overall design principle can be summarized as

$$\begin{aligned}\text{Local Error} &\rightarrow \text{Error Propagation} \rightarrow \text{Global Error} \rightarrow \text{Scale Optimization} \\ &\rightarrow \text{Modulus Reduction} \rightarrow \text{Bootstrapping Reduction}.\end{aligned}$$

Experimental results on CKKS-based Transformer inference demonstrate that the proposed error-aware scale design reduces unnecessary modulus consumption while preserving inference accuracy, providing a complementary optimization direction to existing approaches that primarily focus on operator-level acceleration and bootstrapping implementation. The remainder of this paper first presents the CKKS and Transformer preliminaries, followed by our error propagation analysis and local error models. We then formulate the error-aware scale optimization problem and evaluate the resulting scale and bootstrapping configurations on both standard and HE-friendly Transformer architectures.

Our main contributions are twofold:

- **Error-aware scale optimization for HE-based Transformers.** We develop an error modeling framework that characterizes the local HE computation errors of individual Transformer circuits as functions of their ciphertext scales and quantifies how these local errors propagate to the final output. Based on this analysis, we formulate scale selection as an optimization problem that minimizes the required modulus while satisfying a prescribed global error constraint. In contrast to the conventional uniform-scale strategy, our method allocates the error budget across individual circuits according to their actual contributions to the final inference error.
- **Efficient HE inference through optimized scale allocation.** Our optimized scale allocation translates directly into substantial computational savings in HE inference. As summarized in Table 1, our method preserves the validation accuracy of the corresponding baseline models while substantially reducing modulus consumption and the number of bootstrapping operations. For the standard Transformer, our method reduces modulus consumption by 30.0% and the number of bootstrapping operations by 83.6% compared with THOR. For the HE-friendly Transformer, our method reduces modulus consumption by 33.6% and the number of bootstrapping operations by 80% compared with PowerFormer.

2 PRELIMINARIES

2.1 CKKS AND SCALE MANAGEMENT

The CKKS scheme supports approximate arithmetic over real and complex numbers. Let N be a power of two and $Q_l = \prod_{i=0}^{l} q_i$ denote the ciphertext modulus at level l, where ciphertexts are defined over $R_{Q_l} = \mathbb{Z}_{Q_l}[X]/(X^N + 1)$.

CKKS encodes a message m using a scaling factor Δ, where a larger scale generally provides higher numerical precision at the cost of increased modulus consumption. Multiplying ciphertexts with scales Δ_1 and Δ_2 produces a scale approximately $\Delta_1\Delta_2$. Rescaling by a modulus prime q controls this growth as

$$\Delta_{\text{out}} \simeq \frac{\Delta_1\Delta_2}{q},$$

while removing q from the ciphertext modulus. Therefore, scale selection directly determines the trade-off between numerical precision and modulus consumption in deep homomorphic computations.

2.2 GRAFTING-BASED SCALE MANAGEMENT

Conventional RNS-CKKS couples scale reduction with the granularity of the RNS modulus chain because rescaling removes an RNS prime. We build upon the 30-graft system of Zephyr Seo & Min (2026), which uses the grafting technique Cheon et al. (2025) to enable fine-grained scale management. Its main ciphertext modulus consists of 30-bit RNS primes, while a graft modulus

$$2^{\alpha}, \qquad 0 \le \alpha \le 30,$$

provides finer scale adjustment and is embedded into a 30-bit RNS basis when required for homomorphic multiplication.

The 30-graft representation enables GPU-efficient CKKS computation while allowing intermediate ciphertexts to use different scales Seo & Min (2026). We use this implementation as our underlying HE framework and exploit its fine-grained scale flexibility to optimize the scale Δ_x of each computational stage according to its contribution to the final inference error.

3 ERROR PROPAGATION IN TRANSFORMER ENCODERS

We analyze how local errors introduced by homomorphic evaluation propagate through a Transformer encoder. For notational simplicity, we represent each encoder as the following sequence of

intermediate variables:

$$x \to (Q, K, V) \to \text{score} \to \text{softmax} \to Y \to O \to \text{ln1} \to \text{ff1} \to \text{gelu} \to \text{ff2} \to \text{ln2}.$$

Here, x denotes the input to the encoder, while ln2 is its final output. During homomorphic evaluation, each operation introduces a small numerical error due to approximate CKKS arithmetic and polynomial approximation of nonlinear functions.

First-order error propagation. Consider first an error dx introduced at the input of an encoder. For a sufficiently small perturbation, its effect on the final output can be approximated by first-order linearization:

$$\text{ln2}(x + dx) - \text{ln2}(x) \simeq \frac{d\,\text{ln2}}{dx} dx.$$

The same argument can be applied to errors introduced at every intermediate variable. Let $dQ, dK, \ldots, d\text{ff2}$ denote the local errors introduced during the corresponding homomorphic operations. The total perturbation at the encoder output is then approximated as

$$d\text{ln2} \simeq \frac{d\,\text{ln2}}{dx} dx + \frac{d\,\text{ln2}}{dQ} dQ + \cdots + \frac{d\,\text{ln2}}{d\text{ff2}} d\text{ff2}.$$

The derivatives above represent linearized sensitivity operators. Thus, each term measures not only the magnitude of a local error but also its amplification through all subsequent operations.

Error propagation across multiple encoders. We next consider a Transformer consisting of N sequential encoders. Let var^i denote an intermediate variable var in the i-th encoder, and let out denote the output ln2 of the final encoder. Applying the same first-order approximation to the complete network gives

$$d\text{out} \simeq \sum_{i:\text{encoder}} \sum_{\text{var} \in \mathcal{V}} \frac{d\,\text{out}}{d\,\text{var}^i} d\text{var}^i,$$

where

$$\mathcal{V} = \{x, Q, K, V, \text{score}, \text{softmax}, Y, O, \text{ln1}, \text{ff1}, \text{gelu}, \text{ff2}\}.$$

Therefore, an error introduced in an early encoder can affect the final output after being propagated through all subsequent encoders.

Proposition. Let $\mathbf{e}$ be an i.i.d. random vector with zero mean and variance σ^2, and let A be a square matrix. Then, for sufficiently high-dimensional $\mathbf{e}$,

$$\|A\mathbf{e}\|_2^2 \simeq \frac{\|A\|_F^2}{\dim(\mathbf{e})} \|\mathbf{e}\|_2^2.$$

Proof. Since $\mathbb{E}[e_j e_k] = 0$ for $j \neq k$,

$$\mathbb{E}\|\mathbf{e}\|_2^2 = \dim(\mathbf{e})\sigma^2,$$
$$\mathbb{E}\|A\mathbf{e}\|_2^2 = \sum_{i,j} a_{ij}^2 \mathbb{E}[e_j^2] = \sigma^2 \|A\|_F^2.$$

Hence,

$$\mathbb{E}\|A\mathbf{e}\|_2^2 = \frac{\|A\|_F^2}{\dim(\mathbf{e})} \mathbb{E}\|\mathbf{e}\|_2^2.$$

For sufficiently high-dimensional $\mathbf{e}$, concentration around the expectations gives the stated approximation. □

L_2 error estimate. The numerical errors introduced by CKKS evaluation primarily originate from approximate arithmetic, including encoding, rescaling, and rounding. We model the local errors introduced at different operations as zero-mean and mutually uncorrelated random perturbations. Under this assumption, the cross terms between local error contributions vanish in expectation.

Applying the above proposition to the product of the Jacobian matrix $d\mathrm{out}/d\mathrm{var}^i$ and the local error vector $d\mathrm{var}^i$, the squared output error can be approximated as

$$\|d\mathrm{out}\|_2^2 \simeq \sum_i \sum_{\mathrm{var}} \left\| \frac{d\mathrm{out}}{d\mathrm{var}^i} \right\|_F^2 \frac{\|d\mathrm{var}^i\|_2^2}{\dim(\mathrm{var}^i)}.$$

This decomposition expresses the global homomorphic-evaluation error as the accumulated contributions of local errors generated at individual operations. Importantly, the contribution of a local error is determined not only by its magnitude but also by its amplification through subsequent Transformer operations. Therefore, minimizing each local error independently does not necessarily lead to an efficient homomorphic implementation. Instead, the scale assigned to each operation should reflect its contribution to the final output error. This observation motivates the error-propagation-aware scale optimization developed in the following section.

The Jacobian $d\mathrm{out}/d\mathrm{var}^i$ is computed by the chain rule, similarly to gradient backpropagation. Since the Jacobian matrices depend on the input and intermediate activations, their values cannot be directly determined for encrypted inputs. We therefore compute the Jacobians using each public training sample in the dataset. For the j-th training sample, the propagated error is estimated as

$$\|d\mathrm{out}_j\|_2^2 \simeq \sum_i \sum_{\mathrm{var}} \mathrm{factor}(\mathrm{var}, i, j) \|d\mathrm{var}^i\|_2^2,$$

where

$$\mathrm{factor}(\mathrm{var}, i, j) = \left\| \frac{d\mathrm{out}_j}{d\mathrm{var}^i} \right\|_F^2 \frac{1}{\dim(\mathrm{var}^i)}.$$

Here, i denotes the encoder index and j denotes the training-sample index. The factor therefore quantifies how strongly a local error introduced at each intermediate variable is amplified at the final Transformer output.

4 Error Modeling

The computational circuits of an LLM can be broadly classified into two categories: linear affine circuits and nonlinear activation circuits. For an affine circuit implemented using homomorphic encryption (HE), the squared computation error is empirically observed to decrease approximately in proportion to the inverse square of the ciphertext scale, denoted by Δ.

For example, consider the linear affine circuit for computing the value matrix V. Its squared HE computation error can be modeled as $\|dV\|^2 = \frac{a_V}{\Delta_V^2}$, where a_V is a circuit-dependent coefficient that characterizes the magnitude of the HE computation error. More generally, for the affine variables, we model the squared HE computation error as $\|d\mathrm{var}\|^2 = \frac{a_{\mathrm{var}}}{\Delta_{\mathrm{var}}^2}$. As shown in the left panel of Figure 1, the measured error of V closely follows this inverse-square relationship over the range of scales considered, supporting the validity of the proposed affine error model.

Nonlinear activation circuits have an additional source of error. In such circuits, the nonlinear function is first approximated by a polynomial, which introduces a scale-independent approximation error. The polynomial is then evaluated using HE operations, resulting in an additional scale-dependent computation error. We therefore model the squared error of a nonlinear activation circuit as

$$\|d\mathrm{var}\|^2 = \frac{a_{\mathrm{var}}}{\Delta_{\mathrm{var}}^2} + b_{\mathrm{var}},$$

where $a_{\mathrm{var}}/\Delta_{\mathrm{var}}^2$ represents the scale-dependent HE computation error and b_{var} represents the scale-independent error introduced by the polynomial approximation. For the GELU activation, the measured error initially decreases rapidly as the HE scale increases and eventually approaches a nonzero floor. As shown in the right panel of Figure 1, this behavior is well captured by the proposed error model. In particular, the decreasing component corresponds to the HE computation error, whereas the asymptotic error floor is attributed to the polynomial approximation error. Thus, increasing the

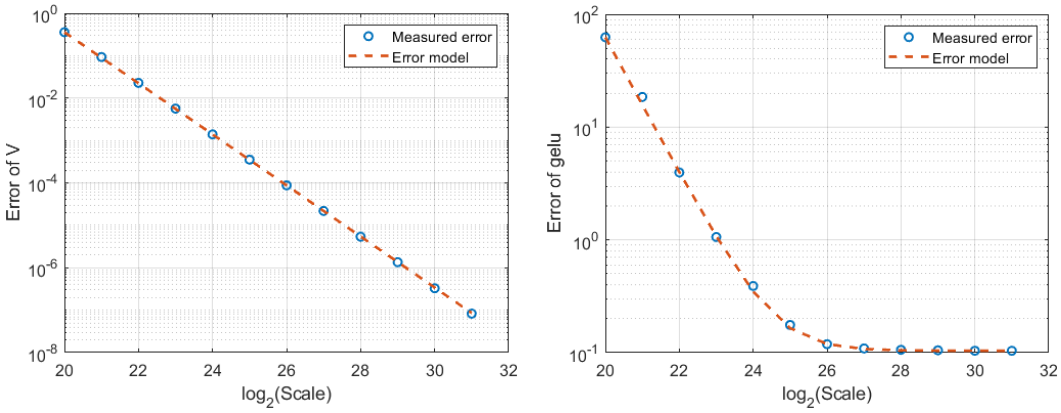

Figure 1: Measured errors and fitted error models for the value matrix V (left) and the GELU activation (right) at different HE scales. The experiments were conducted on the first sample of the RTE validation set at the first encoder. The error of V follows the inverse-square scale relationship, while the GELU error exhibits an asymptotic error floor due to the scale-independent polynomial approximation error.

HE scale beyond this regime provides limited benefit because the scale-independent approximation error becomes the dominant source of error.

This decomposition is particularly important for scale optimization, since increasing the HE scale reduces only the scale-dependent HE computation error and has no effect on the polynomial approximation error.

Our experimental results indicate that the magnitude of the local homomorphic encryption error, $\|d\mathrm{var}^i\|^2$, remains relatively stable across encoder indices i and input samples j. Based on this empirical observation, we approximate the local error for each variable by a common value $\|d\mathrm{var}\|^2$ and, using the linear and nonlinear error models defined above, estimate the total error for sample j as

$$\|d\mathrm{out}_j\|^2 \simeq \sum_{\mathrm{var}} \left(\frac{a_{\mathrm{var}}}{\Delta_{\mathrm{var}}^2} + b_{\mathrm{var}} \right) \mathrm{factor}(\mathrm{var}, j), \qquad \text{for each training sample } j, \tag{1}$$

where

$$\mathrm{factor}(\mathrm{var}, j) = \sum_i \mathrm{factor}(\mathrm{var}, i, j).$$

5 Scale Optimization

Based on the error model and error propagation analysis presented in the previous sections, we propose an efficient scale allocation method for the HE implementation of LLM circuits. The objective is to allocate an appropriate HE scale to each computational circuit while maintaining the overall output error within a prescribed bound.

As shown in the previous section, the impact of a local error on the final output depends not only on its magnitude but also on its amplification factor. Therefore, assigning a uniform scale to all computational stages can lead to inefficient error allocation. Computational stages with large amplification factors require more stringent control of their local errors, whereas stages with small amplification factors can tolerate relatively larger errors without substantially affecting the final output error. Based on this observation, we optimize the scale of each computational stage by explicitly incorporating its contribution to the propagated output error.

The proposed scale optimization determines the scale Δ_{var} for each computational variable var. For a given set of scales, the total error of the resulting HE circuit should not exceed a predefined threshold $E_{\max}$. We impose this constraint on every training sample using the corresponding sample-wise amplification factors.

(a) Standard Transformer

var	a_{var}	b_{var}	l_{var}	Δ_{var}
QKV	6.89×10^{12}	0	1	2^{23}
Score	8.32×10^{12}	0	4	2^{24}
Softmax	2.02×10^{15}	2.14×10^{-5}	46	2^{31}
Y	7.45×10^{12}	0	3	2^{26}
O	2.39×10^{11}	0	1	2^{24}
LN1	2.53×10^{12}	5.32×10^{-5}	10	2^{24}
FF1	4.58×10^{12}	0	1	2^{25}
GELU	8.86×10^{13}	1.04×10^{-1}	9	2^{27}
FF2	1.21×10^{12}	0	1	2^{25}
LN2	3.33×10^{12}	1.48×10^{-4}	10	2^{24}

(b) HE-friendly Transformer

var	a_{var}	b_{var}	l_{var}	Δ_{var}
QKV	1.86×10^{12}	0	1	2^{24}
Score	9.54×10^{12}	0	4	2^{23}
BPMax	1.04×10^{18}	0	3	2^{34}
Y	9.27×10^{12}	0	3	2^{27}
O	6.34×10^{10}	0	1	2^{24}
BatchLN1	2.77×10^{11}	0	2	2^{24}
FF1	3.12×10^{12}	0	1	2^{24}
GELU	8.10×10^{13}	1.04×10^{-1}	9	2^{28}
FF2	1.02×10^{12}	0	1	2^{25}
BatchLN2	1.07×10^{11}	0	2	2^{24}

Table 2: Error-model coefficients, circuit depths, and optimized scales for (a) the standard Transformer and (b) the HE-friendly Transformer.

In general, multiple scale configurations can satisfy these error constraints. Our objective is to improve the efficiency of PPLLM inference by minimizing modulus consumption. Reducing modulus consumption can decrease the required bootstrapping frequency and thereby reduce the overall inference time. We therefore formulate the scale optimization problem as

$$\begin{aligned} &\underset{\{\Delta_{\text{var}}\}}{\text{minimize}} \quad \text{Total modulus} = \sum_{\text{var}} l_{\text{var}} \log(\Delta_{\text{var}}) \\ &\text{subject to} \quad \|d\text{out}_j\|^2 = \sum_{\text{var}} \left(\frac{a_{\text{var}}}{\Delta_{\text{var}}^2} + b_{\text{var}} \right) \text{factor}(\text{var}, j) \leq E_{\max}, \forall j. \end{aligned}$$

Here, the error constraint is imposed on every training sample j, ensuring that the estimated output error does not exceed $E_{\max}$ for any training sample. l_{var} denotes the circuit depth, i.e., the number of times the scale Δ_{var} contributes to the ciphertext modulus consumption.

Algorithm 1 employs consensus ADMM to efficiently solve the proposed scale optimization problem. A global variable $\mathbf{z}$ is introduced together with a sample-specific variable $\mathbf{y}_j$ for each sample j. The algorithm alternates between a closed-form update of $\mathbf{z}$, projection of each $\mathbf{y}_j$ onto its sample-specific feasible set, and an update of the scaled dual variable $\mathbf{u}_j$. The iterations continue until convergence, after which the optimized scale is obtained as $\Delta_{\text{var}}^* = 1/\sqrt{z_{\text{var}}}$.

6 Experiment

We evaluate the proposed framework on both standard and HE-friendly Transformer architectures to examine how error-aware scale allocation affects modulus consumption, bootstrapping frequency, inference time, and accuracy. We report results for standard BERT on MRPC and an HE-friendly BERT architecture on RTE.

6.1 Transformer

We evaluate our proposed scale optimization method on a standard Transformer architecture using BERT and the Microsoft Research Paraphrase Corpus (MRPC) dataset. We use the fine-tuned BERT model described in Appendix A and compute the amplification factors for the individual computational circuits using the MRPC training set.

Specifically, we compute the amplification factors for each computational circuit and each training sample, and directly use these sample-wise factors to construct the error constraints in the scale optimization problem of Section 5. For reference, the minimum and maximum amplification factors observed across the training samples are reported in Table 7 of Appendix D.1.

(a) Standard Transformer

Operation	THOR	Ours
Attention layer	49.77	14.97
Attention score	16.25	12.63
Softmax	15.53	13.23
Attention heads	13.08	14.47
Multi-head attention	27.43	5.03
LayerNorm1	7.13	0.94
FC1	49.80	16.91
GELU	29.42	6.44
FC2	49.19	14.29
LayerNorm2	4.10	0.92
Bootstrapping	337.86	35.41
Total	**602.26**	**135.24**

(b) HE-friendly Transformer

Operation	PowerFormer	Ours
Attention layer	57.65	15.03
Attention score	28.76	13.40
BPMax	0.75	0.61
Attention heads	18.95	13.87
Multi-head attention	22.54	5.46
BatchLN1	0.37	0.28
FC1	59.21	14.17
GELU	8.31	4.93
FC2	43.77	11.95
BatchLN2	0.30	0.14
Bootstrapping	103.72	6.87
Total	**344.32**	**86.71**

Table 3: Execution-time breakdown for 12 Transformer encoders on an NVIDIA A100 GPU. All execution times are reported in seconds. (a) Standard Transformer: THOR vs. ours. (b) HE-friendly Transformer: PowerFormer vs. ours.

Using the error models from Section 4 and the sample-wise amplification factors, we apply the scale optimization method in Section 5 to the Transformer. Table 2(a) reports the error-model coefficients a_{var} and b_{var}, circuit depth l_{var}, and optimized scale Δ_{var} for each computational circuit of the standard Transformer. We set the global error bound $E_{\max} = 5$ to preserve the binary classification decisions of the plaintext BERT model under the predicted HE error.

Based on the error requirements above, we further optimize the bootstrapping parameters to recover the highest possible ciphertext level while satisfying the required numerical accuracy. This error-aware configuration allows bootstrapping to recover a higher ciphertext level than the configuration used in THOR. The detailed bootstrapping parameters and recovered levels are provided in Appendix F.1.

We then implement all 12 Transformer encoders using the optimized scales and bootstrapping configurations. The reduced modulus consumption from scale optimization, together with the increased level recovered by bootstrapping, reduces the need for frequent bootstrapping. After applying the real–imaginary combination technique, our implementation requires 10 bootstrapping operations per encoder. The detailed bootstrapping configuration is provided in Appendix F.2.

To evaluate the resulting runtime improvement, Table 3(a) compares the execution time of THOR and our method for all 12 Transformer encoders on an NVIDIA A100 GPU. Our implementation is built upon the GPU-efficient HE implementation of Zephyr; however, Zephyr is not included as a runtime baseline because it does not implement the complete BERT Transformer encoder. Overall, our method reduces the execution time from 602.26 s for THOR to 135.24 s, corresponding to a $4.45\times$ speedup. This improvement reflects both the underlying implementation-level acceleration and the reductions achieved by our scale and bootstrapping optimizations.

6.2 HE-FRIENDLY TRANSFORMER

We further evaluate our method on the HE-friendly Transformer architecture of PowerFormer using BERT and the Recognizing Textual Entailment (RTE) dataset. PowerFormer replaces Softmax and LayerNorm with BPMax and BatchLN, respectively, for efficient HE evaluation. We use the knowledge-distilled model provided by PowerFormer and compute the amplification factors using the RTE training set.

Among the 2,490 training samples, 20 produce outlier activations outside the input ranges supported by the polynomial approximations and are excluded from the analysis. We use the sample-wise amplification factors of the remaining 2,470 samples for scale optimization, as in Section 6.1. Their minimum and maximum values are reported for reference in Table 8 of Appendix D.2.

Task	Plaintext Acc. (%)	Ciphertext Acc. (%)	Mismatches
MRPC	85.04	85.04	0
RTE	69.67	69.31	3

Table 4: Plaintext and ciphertext inference accuracy (%) and prediction mismatches on the MRPC (408 samples) and RTE (277 samples) validation sets, corresponding to the experiments in Sections 6.1 and 6.2, respectively.

Using these amplification factors and the error models from Section 4, we optimize the scales with $E_{\max} = 5$, which is selected to preserve the binary classification decisions of the plaintext model under the predicted HE error. Table 2(b) reports the error-model coefficients a_{var} and b_{var}, circuit depth l_{var}, and optimized scale Δ_{var} for each computational circuit of the HE-friendly Transformer.

We further optimize the bootstrapping parameters according to the error requirements, allowing a higher ciphertext level to be recovered than with the PowerFormer configuration. Together with the reduced modulus consumption from scale optimization, this reduces the number of bootstrapping operations from 10 to 2 per encoder. The bootstrapping parameters and ciphertext-level evolution are detailed in Appendix F.1 and Appendix F.3, respectively.

Finally, Table 3(b) compares the execution times of PowerFormer and our method for all 12 Transformer encoders on an NVIDIA A100 GPU. Overall, our method reduces the execution time from 344.32 s for PowerFormer to 86.71 s, corresponding to a $3.97\times$ speedup. This improvement results from the reduced modulus consumption and bootstrapping frequency enabled by our error-aware scale and bootstrapping optimization.

6.3 Accuracy

Finally, we evaluate the effect of the proposed scale optimization on task accuracy. As shown in Table 4, ciphertext inference achieves nearly the same accuracy as plaintext inference on both MRPC and RTE, demonstrating that our optimization preserves task accuracy in both experimental settings.

7 Conclusion

In this work, we presented an error-propagation-aware scale optimization framework for efficient HE-based Transformer inference. We developed an error model that characterizes the scale-dependent errors introduced by individual HE computational circuits and quantifies their propagation to the final output. Based on this model, we formulated scale selection as an optimization problem that determines circuit-wise scales while satisfying a prescribed global error constraint.

The resulting scale allocation substantially reduces ciphertext modulus consumption and enables more computation to be performed between bootstrapping operations. For the standard Transformer, our method reduces modulus consumption by 30.0% compared with THOR and reduces the execution time of the 12 Transformer encoders from 602.26 s to 135.24 s, corresponding to a $4.45\times$ speedup. For the HE-friendly Transformer, our method reduces modulus consumption by 33.6% and the number of bootstrapping operations by 80% compared with PowerFormer, achieving a $3.97\times$ speedup while maintaining comparable model accuracy.

We further demonstrate that the proposed scale optimization can be integrated with HE-friendly model design and nonlinear approximation. Specifically, HE-friendly fine-tuning and a composite polynomial approximation of GELU enable the framework to accommodate both standard and HE-friendly Transformer architectures.

As privacy-preserving LLM inference is extended to larger and deeper models, scale allocation becomes increasingly important because even moderate reductions in the modulus requirement of individual Transformer blocks can accumulate across many layers. Our error-propagation-aware framework provides a systematic approach to balancing numerical accuracy and modulus consumption, offering a practical foundation for designing efficient HE circuits for large-scale LLM inference.

AI USE STATEMENT

Generative AI tools were used **only for language editing**, including improving the clarity, grammar, and presentation of the manuscript. They were not used to develop the methodology, conduct experiments, analyze experimental results, or determine the scientific conclusions of this work. All AI-assisted language edits were reviewed and revised by the authors, who take full responsibility for the final content of the manuscript.

ETHICS STATEMENT

This work focuses on improving the efficiency of privacy-preserving inference using homomorphic encryption. The experiments use publicly available benchmark datasets and pretrained models and do not involve the collection of personally identifiable information or experiments with human subjects. The proposed techniques operate on encrypted data and are intended to improve the practicality of privacy-preserving machine learning inference. We do not identify additional ethical concerns specific to the methodology presented in this work.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the error models, error-propagation analysis, and scale optimization procedure in the main paper. The appendices provide additional details on the computation of amplification factors, HE-friendly model design, bootstrapping configurations, and ciphertext-level evolution. Experimental settings, including the error-model coefficients, circuit depths, optimized scales, and bootstrapping parameters, are reported to facilitate reproduction of the experimental results.

REFERENCES

Jean-Philippe Bossuat, Christian Mouchet, Juan R. Troncoso-Pastoriza, and Jean-Pierre Hubaux. Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys. In *Advances in Cryptology – EUROCRYPT 2021*, pp. 469–498. Springer, 2021. doi: 10.1007/978-3-030-77870-5_18.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011. doi: 10.1561/2200000016.

Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology – ASIACRYPT 2017*, pp. 409–437. Springer, 2017. doi: 10.1007/978-3-319-70694-8_15.

Jung Hee Cheon, Miran Kim, Kyoohyung Kim, and Yongsoo Song. Bootstrapping for approximate homomorphic encryption. In *Advances in Cryptology – EUROCRYPT 2018*, pp. 360–384. Springer, 2018. doi: 10.1007/978-3-319-78381-9_14.

Jung Hee Cheon, Hyeongmin Choe, Minsik Kang, Jaehyung Kim, Seonghak Kim, Johannes Mono, and Taeyeong Noh. Grafting: Decoupled scale factors and modulus in rns-ckks. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25)*, pp. 1098–1112. ACM, 2025. doi: 10.1145/3719027.3765083.

Wonhee Cho, Guillaume Hanrot, Taeseong Kim, Minje Park, and Damien Stehlé. Fast and accurate homomorphic softmax evaluation. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security*, pp. 4391–4404. ACM, 2024. doi: 10.1145/3658644.3670369.

Wonseok Choi, Jongmin Kim, and Jung Ho Ahn. Cheddar: A swift fully homomorphic encryption library designed for gpu architectures. In *Proceedings of the 31st ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 35–49. ACM, 2026. doi: 10.1145/3760250.3762223.

Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48, pp. 201–210. PMLR, 2016.

Kyoohyung Han and Dohyeong Ki. Better bootstrapping for approximate homomorphic encryption. In *Topics in Cryptology – CT-RSA 2020*, pp. 364–390. Springer, 2020. doi: 10.1007/978-3-030-40186-3_16.

Wonkyung Jung, Sangpyo Kim, Jung Ho Ahn, Jung Hee Cheon, and Younho Lee. Over 100x faster bootstrapping in fully homomorphic encryption through memory-centric optimization with gpus. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, (4):114–148, 2021. doi: 10.46586/tches.v2021.i4.114-148.

Charanjit S. Jutla and Srinivas Manohar. Sine series approximation for ckks bootstrapping. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2022.

Seonghak Kim, Minji Park, Jaehyung Kim, Taekyung Kim, and Chohong Min. Evalround algorithm in ckks bootstrapping. In *Advances in Cryptology – ASIACRYPT 2022*, volume 13792, pp. 161–187. Springer, 2022. doi: 10.1007/978-3-031-22966-4_6.

Eunsang Lee, Joon-Woo Lee, Jong-Seon No, and Young-Sik Kim. Minimax approximation of sign function by composite polynomial for homomorphic comparison. *IEEE Transactions on Dependable and Secure Computing*, 19(6):3711–3727, 2022.

Joon-Woo Lee, Eunsang Lee, Young-Sik Kim, and Jong-Seon No. High-precision bootstrapping of rns-ckks homomorphic encryption. In *Advances in Cryptology – EUROCRYPT 2021*, 2021a.

Joon-Woo Lee, Eunsang Lee, Yongwoo Lee, Young-Sik Kim, and Jong-Seon No. High-precision bootstrapping of rns-ckks homomorphic encryption using optimal minimax polynomial approximation and inverse sine function. In *Advances in Cryptology – EUROCRYPT 2021*, pp. 618–647. Springer, 2021b. doi: 10.1007/978-3-030-77870-5_22.

Jungho Moon, Zhanibek Omarov, Donghoon Yoo, Yongdae An, and Heewon Chung. Adaptive successive over-relaxation method for a faster iterative approximation of homomorphic operations, 2024. Cryptology ePrint Archive, Paper 2024/1366.

Jungho Moon, Dongwoo Yoo, Xiaoqian Jiang, and Miran Kim. Thor: Secure transformer inference with homomorphic encryption. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pp. 3765–3779. ACM, 2025. doi: 10.1145/3719027.3765150.

Qi Pang, Jinhao Zhu, Helen Möllering, Wenting Zheng, and Thomas Schneider. Bolt: Privacy-preserving, accurate and efficient inference for transformers. In *2024 IEEE Symposium on Security and Privacy (SP)*, pp. 4753–4771. IEEE, 2024. doi: 10.1109/SP54263.2024.00130.

Dongjin Park, Eunsang Lee, and Joon-Woo Lee. Powerformer: Efficient and high-accuracy privacy-preserving language model with homomorphic encryption. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11090–11111, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.543.

Jaiyoung Park, Sejin Park, Jai Hyun Park, Jung Ho Ahn, Jung Hee Cheon, Guillaume Hanrot, Jung Woo Kim, Minje Park, and Damien Stehlé. Scaling up privacy-preserving ml: A ckks implementation of llama-2-7b, 2026.

Sieun Seo and Chohong Min. Zephyr: Gpu-efficient homomorphic encryption for privacy-preserving transformer inference, 2026. Cryptology ePrint Archive, Paper 2026/932.

Hyewon Sung, Sieun Seo, Taekyung Kim, and Chohong Min. Evalround+ bootstrapping and its rigorous analysis for ckks scheme. *IEEE Access*, 13:140847–140866, 2025. doi: 10.1109/ACCESS.2025.3595177.

Dataset	Method	Acc. (%)	$\|x_{\text{GELU}}\|_\infty$	$\|x_{\text{SM}}\|_\infty$	$\|\text{Var}(x_{\text{LN1}})\|_\infty$	$\|\text{Var}(x_{\text{LN2}})\|_\infty$
RTE	Baseline	–	146	42.3	2.05	2.10×10^4
	PowerFormer	70.0	55.1	33.1	1.23×10^5	1.74×10^5
	Ours	71.4	37.0	20.8	4.59	19.5
MRPC	Baseline	–	141	38.2	2.09	2.38×10^4
	THOR	85.2	122	44.1	1.89	1.23×10^3
	Ours	85.0	42.4	21.2	2.64	34.3
SST-2	Baseline	–	134	34.6	1.37	1660
	Ours	93.2	43.5	18.1	3.74	35.6

Table 5: Comparison of accuracy and the maximum nonlinear input magnitudes before and after range-regularized fine-tuning. For GELU and Softmax, we report the infinity norm of the input, while for LayerNorm, we report the maximum variance over the hidden dimension.

Jianhang Xie, Sicheng Tan, Vishnu Naresh Boddeti, and Zhichao Lu. Atlas: Automated approximation of transformers for efficient homomorphic inference in one hour. *arXiv preprint arXiv:2607.23478*, 2026.

Itamar Zimerman, Moran Baruch, Nir Drucker, Gilad Ezov, Omri Soceanu, and Lior Wolf. Converting transformers to polynomial form for secure inference over homomorphic encryption. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62803–62814. PMLR, 2024.

A Range-Regularized Fine-Tuning for HE

In HE-based Transformer inference, nonlinear operations such as GELU, Softmax, and the inverse square root in LayerNorm are typically evaluated using polynomial approximations, whose efficiency and accuracy strongly depend on their input ranges. To obtain HE-friendly input ranges, we fine-tune the model using the following range-regularized objective:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda_{\text{GELU}} \cdot \|x_{\text{GELU}}\|_\infty^2 + \lambda_{\text{SM}} \cdot \|x_{\text{SM}}\|_\infty^2 + \lambda_{\text{LN1}} \cdot \|\text{Var}\,(x_{\text{LN1}})\|_\infty + \lambda_{\text{LN2}} \cdot \|\text{Var}\,(x_{\text{LN2}})\|_\infty .$$

Here, $\mathcal{L}_{\text{CE}}$ is the standard cross-entropy loss for the downstream task. We set

$$\lambda_{\text{GELU}} = \lambda_{\text{SM}} = 10^{-5}, \qquad \lambda_{\text{LN1}} = \lambda_{\text{LN2}} = 10^{-6}.$$

x_{GELU} and x_{SM} denote the inputs to GELU and Softmax, respectively, where x_{SM} is measured before applying the attention mask. Similarly x_{LN1} and x_{LN2} denote the inputs to the first and second LayerNorm operations in each Transformer block. For GELU and Softmax, the infinity norm represents the maximum absolute input over all layers, samples, and valid tokens within each mini-batch. For LayerNorm, the variance is computed over the hidden dimension for each valid token, and its maximum is taken over all layers and samples. We fine-tune the model with five random seeds $\{1, 5, 14, 22, 37\}$ and select the model with the highest validation accuracy among those satisfying the target ranges.

Table 5 compares the downstream accuracy and nonlinear ranges of the baseline models and our selected range-regularized models. Our models effectively constrain the input ranges of GELU and Softmax and the variance ranges of LayerNorm to the target domains, while maintaining comparable downstream accuracy. These results demonstrate that the model can be adapted to HE-friendly numerical ranges without sacrificing task performance.

B ADMM-Based Scale Optimization

We employ the alternating direction method of multipliers (ADMM) Boyd et al. (2011) to solve the constrained scale optimization problem described in Section 5. Let N denote the total number of Transformer encoders and M the number of samples used to construct the error model. Here, i and j denote the encoder and sample indices, respectively.

Algorithm 1 ADMM-based Scale Optimization

Initialize $\mathbf{z}$, $\mathbf{y}_j$, and $\mathbf{u}_j$ for $j = 1, \ldots, M$.
$c_{\mathrm{var},j} \leftarrow a_{\mathrm{var}} \mathrm{factor}_{\mathrm{var},j}$.
$d_j \leftarrow E_{\max} - \sum_{\mathrm{var}} b_{\mathrm{var}} \cdot \mathrm{factor}_{\mathrm{var},j}$.
while not converged **do**
  $v_{\mathrm{var}} \leftarrow \frac{1}{M} \sum_{j=1}^{M} (y_{\mathrm{var},j} - u_{\mathrm{var},j})$.
  $z_{\mathrm{var}} \leftarrow \frac{v_{\mathrm{var}} + \sqrt{v_{\mathrm{var}}^2 + \frac{2l_{\mathrm{var}}}{\rho M}}}{2}$.
  for $j = 1, \ldots, M$ **do**
    $\mathbf{q}_j \leftarrow \mathbf{z} + \mathbf{u}_j$.
    $\mathbf{y}_j \leftarrow \Pi_{\mathcal{C}_j}(\mathbf{q}_j)$.
    $\mathbf{u}_j \leftarrow \mathbf{u}_j + \mathbf{z} - \mathbf{y}_j$.
  end for
end while
$\Delta^*_{\mathrm{var}} \leftarrow 1/\sqrt{z_{\mathrm{var}}}$.

The scale optimization problem is formulated as

$$\min_{\mathbf{x}} F(\mathbf{x}) \quad \text{s.t.} \quad \mathbf{x} \in \mathcal{C}, \tag{2}$$

where $\mathbf{x}$ represents the scale parameters, $\mathcal{C}$ denotes the feasible set imposed by the HE implementation, and $F(\mathbf{x})$ is the estimated global error obtained from the error-propagation model. In particular, the objective aggregates the contributions from all M samples and N encoders.

To separate error minimization from the feasibility constraints, we introduce an auxiliary variable $\mathbf{z}$ and impose $\mathbf{x} = \mathbf{z}$. The resulting ADMM iterations consist of three steps. First, $\mathbf{x}$ is updated by minimizing the error objective together with a quadratic penalty:

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \left\{ F(\mathbf{x}) + \frac{\rho}{2} \left\| \mathbf{x} - \mathbf{z}^k + \mathbf{u}^k \right\|_2^2 \right\}, \tag{3}$$

where $\rho > 0$ is the penalty parameter and $\mathbf{u}$ is the scaled dual variable.

Second, the updated scale vector is projected onto the feasible set:

$$\mathbf{z}^{k+1} = \Pi_{\mathcal{C}} \left(\mathbf{x}^{k+1} + \mathbf{u}^k \right). \tag{4}$$

For the box constraints used in our implementation, this projection is performed independently for each scale parameter by clipping it to its prescribed lower and upper bounds.

Finally, the scaled dual variable is updated as

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \mathbf{x}^{k+1} - \mathbf{z}^{k+1}. \tag{5}$$

The variables are initialized with $\mathbf{z}^0 = \mathbf{x}^0$ and $\mathbf{u}^0 = \mathbf{0}$. The iterations are terminated when both the primal residual $\mathbf{r}^k = \mathbf{x}^k - \mathbf{z}^k$ and the dual residual $\mathbf{s}^k = \rho(\mathbf{z}^k - \mathbf{z}^{k-1})$ are sufficiently small, or when a maximum number of iterations is reached Boyd et al. (2011).

The $\mathbf{x}$-update minimizes the propagated HE error, while the $\mathbf{z}$-update enforces the allowable scale range. Therefore, ADMM provides a simple decomposition between the error-propagation objective and the implementation constraints. The resulting scale parameters are subsequently used for the HE Transformer computation.

The ADMM variables are initialized as $\mathbf{z} = \frac{1}{\max\left(1, \max_j \sum_{var} \mathrm{factor}_{var,j}\right)}, \mathbf{y}_j = \mathbf{z}, \mathbf{u}_j = \mathbf{0}$, for each j. Here, $\mathbf{z}$ is the common scale variable, while $\mathbf{y}_j$ and $\mathbf{u}_j$ denote the sample-specific primal and scaled dual variables, respectively. The penalty parameter ρ controls the relative decay of the primal and dual residuals during the ADMM iterations. In our experiments, we set $\rho = 10$ throughout all optimization runs. The iterations are terminated when both residuals are sufficiently small or when a prescribed maximum number of iterations is reached Boyd et al. (2011).

C GELU Approximation

Lee et al. (2022) proposed a composite-polynomial approach for approximating sign-like functions using multiple stages of low-degree minimax polynomials. In contrast, we construct a two-stage

Table 6: Comparison of polynomial approximations for GELU.

Method	Degree	Depth	Range	$\|f(x) - p(x)\|_\infty$
THOR	$31 \times 27 = 837$	$5 + 5 = 10$	$[-4, 4]$	5.65×10^{-4}
PowerFormer	$5 \times 5 \times 5 \times 7 = 875$	$3 + 3 + 3 + 3 = 12$	$[-40, 40]$	3.40×10^{-4}
Ours	$15 \times 15 = 225$	$4 + 4 = 8$	$[-40\sqrt{2}, 40\sqrt{2}]$	6.36×10^{-4}

composite polynomial approximation. The GELU activation can be written in terms of the error function as $\mathrm{GELU}(x) = \frac{x}{2}\left(1 + \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)\right)$. Therefore, it is sufficient to construct an accurate polynomial approximation of $\mathrm{erf}(x)$, from which the GELU approximation can be obtained directly.

Specifically, we first construct a degree-15 minimax polynomial p_1 that approximates $\mathrm{erf}(x)$ over a wide interval. We then construct a second degree-15 minimax polynomial p_2 to approximate the composition involving the inverse of p_1:

$$p_1(x) \approx \mathrm{erf}(x), \qquad \text{on } [-40, 40], \qquad \text{with minmax error } 0.441, \tag{6}$$

$$p_2(x) \approx \mathrm{erf}\left(P_1^{-1}(x)\right), \qquad \text{on } [-0.637, 0.637], \qquad \text{with minmax error } 6.36\times10^{-4}. \tag{7}$$

By composing the two polynomials, we obtain

$$p_2\left(p_1\left(\frac{x}{\sqrt{2}}\right)\right) \approx \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right), \tag{8}$$

which provides the nonlinear approximation used for GELU.

As summarized in Table 6, our construction uses two polynomial stages, each of degree 15, resulting in an overall degree of $15 \times 15 = 225$ and a multiplicative depth of $4 + 4 = 8$. It achieves a GELU approximation error of 6.36×10^{-4} over the wide input range $[-40\sqrt{2}, 40\sqrt{2}]$.

D Detailed Amplification-Factor Statistics

This appendix provides detailed amplification-factor statistics for the experiments presented in Sections 6.1 and 6.2. We report the minimum and maximum amplification factors for each intermediate operation across all 12 Transformer encoder layers. These statistics are reported only to characterize the observed range of amplification factors; the scale optimization directly uses the sample-wise amplification factors, as described in Section 5. Appendices D.1 and D.2 provide the detailed results for the experiments in Sections 6.1 and 6.2, respectively.

D.1 Amplification-Factor Statistics for the Experiment in Section 6.1

We compute the amplification factors using all 3,668 samples from the MRPC training set. Table 7 reports the observed minimum and maximum amplification factors for each intermediate operation across all 12 Transformer encoder layers. These statistics are provided for reference; the scale optimization uses the sample-wise amplification factors directly.

D.2 Amplification-Factor Statistics for the Experiment in Section 6.2

We use the knowledge-distilled model provided by PowerFormer without additional training or fine-tuning. Among the 2,490 RTE training samples, 20 produce outlier activations outside the input ranges supported by our polynomial approximations and are excluded from the analysis. Table 8 reports the observed minimum and maximum amplification factors over the remaining 2,470 samples for each intermediate operation across all 12 Transformer encoder layers.

E Implementation Details of Softmax and Layer Normalization

This section provides implementation details of the Softmax and Layer Normalization operations used in our HE-based Transformer. Since both operations involve nonlinear functions and divi-

i	Q,K,V	score	softmax	Y	O	ln1	ff1	gelu	ff2	ln2
0	[2.43e-4, 6.64e-2]	[2.35e-4, 6.05e-2]	[4.86e-1, 4.66e+1]	[1.89e-2, 1.44e+0]	[3.01e-2, 2.30e+0]	[9.26e-3, 8.72e-1]	[4.06e-4, 4.34e-2]	[7.41e-3, 6.76e-1]	[7.52e-3, 6.85e-1]	[5.31e-2, 2.96e+0]
1	[6.59e-4, 1.46e-1]	[6.20e-4, 1.25e-1]	[1.12e+0, 6.64e+1]	[2.93e-2, 1.47e+0]	[4.87e-2, 2.44e+0]	[2.04e-2, 1.27e+0]	[8.95e-4, 7.19e-2]	[1.78e-2, 1.02e+0]	[1.62e-2, 9.31e-1]	[6.34e-2, 2.56e+0]
2	[6.88e-4, 1.00e-1]	[6.71e-4, 8.50e-2]	[1.12e+0, 5.92e+1]	[3.24e-2, 1.20e+0]	[5.76e-2, 2.13e+0]	[3.00e-2, 1.48e+0]	[1.18e-3, 6.92e-2]	[2.72e-2, 1.28e+0]	[2.42e-2, 1.14e+0]	[7.84e-2, 2.81e+0]
3	[1.94e-3, 1.66e-1]	[1.71e-3, 1.32e-1]	[2.15e+0, 9.18e+1]	[4.38e-2, 1.54e+0]	[6.75e-2, 2.37e+0]	[4.47e-2, 1.73e+0]	[2.07e-3, 8.64e-2]	[4.06e-2, 1.55e+0]	[3.43e-2, 1.31e+0]	[8.29e-2, 2.84e+0]
4	[3.55e-3, 2.70e-1]	[3.10e-3, 1.95e-1]	[3.06e+0, 1.12e+2]	[5.25e-2, 1.64e+0]	[6.47e-2, 2.02e+0]	[5.47e-2, 1.81e+0]	[3.03e-3, 1.00e-1]	[4.79e-2, 1.58e+0]	[3.93e-2, 1.30e+0]	[8.19e-2, 2.57e+0]
5	[5.64e-3, 3.06e-1]	[3.66e-3, 1.81e-1]	[3.04e+0, 8.91e+1]	[4.93e-2, 1.32e+0]	[5.85e-2, 1.56e+0]	[5.47e-2, 1.44e+0]	[3.00e-3, 7.19e-2]	[4.69e-2, 1.27e+0]	[3.96e-2, 1.07e+0]	[8.15e-2, 1.81e+0]
6	[4.94e-3, 2.03e-1]	[3.20e-3, 1.11e-1]	[3.04e+0, 6.22e+1]	[4.94e-2, 1.05e+0]	[6.21e-2, 1.32e+0]	[7.18e-2, 1.42e+0]	[4.19e-3, 8.50e-2]	[4.95e-2, 1.14e+0]	[4.29e-2, 9.90e-1]	[7.69e-2, 1.56e+0]
7	[3.82e-3, 1.58e-1]	[2.36e-3, 8.41e-2]	[2.77e+0, 4.50e+1]	[4.73e-2, 8.98e-1]	[6.14e-2, 1.17e+0]	[6.89e-2, 1.22e+0]	[4.37e-3, 8.93e-2]	[4.24e-2, 8.75e-1]	[3.93e-2, 8.11e-1]	[7.70e-2, 1.33e+0]
8	[5.23e-3, 1.37e-1]	[3.76e-3, 9.17e-2]	[3.82e+0, 6.18e+1]	[4.70e-2, 7.51e-1]	[5.44e-2, 8.70e-1]	[5.38e-2, 8.18e-1]	[2.72e-3, 4.21e-2]	[4.40e-2, 6.82e-1]	[4.11e-2, 6.37e-1]	[6.03e-2, 8.62e-1]
9	[3.74e-3, 8.00e-2]	[3.14e-3, 4.97e-2]	[2.34e+0, 3.48e+1]	[2.98e-2, 4.91e-1]	[3.64e-2, 5.98e-1]	[4.61e-2, 6.83e-1]	[1.77e-3, 2.92e-2]	[4.31e-2, 6.33e-1]	[3.70e-2, 5.43e-1]	[4.97e-2, 7.66e-1]
10	[3.41e-3, 5.58e-2]	[1.76e-3, 3.36e-2]	[2.65e+0, 3.63e+1]	[3.03e-2, 4.90e-1]	[3.55e-2, 5.74e-1]	[7.03e-2, 7.84e-1]	[2.15e-3, 2.62e-2]	[6.48e-2, 7.30e-1]	[5.88e-2, 6.62e-1]	[9.46e-2, 8.67e-1]
11	[4.33e-3, 1.74e-1]	[2.99e-3, 1.01e-1]	[7.90e+0, 3.09e+1]	[8.01e-2, 3.73e-1]	[7.72e-2, 3.60e-1]	[2.15e-1, 4.58e-1]	[5.69e-3, 3.64e-2]	[9.52e-2, 3.69e-1]	[9.67e-2, 3.75e-1]	[1.00e+0, 1.00e+0]

Table 7: Range of amplification factors for each intermediate operation over the MRPC training set, where i denotes the Transformer encoder index. Each entry represents $[\min, \max]$. The notation $\texttt{e}\pm\texttt{n}$ denotes $\times 10^{\pm n}$.

i	Q,K,V	score	softmax	Y	O	ln1	ff1	gelu	ff2	ln2
0	[2.35e-6, 4.26e-5]	[2.52e-6, 3.47e-5]	[1.82e-1, 1.21e+0]	[5.92e-3, 2.49e-2]	[8.72e-3, 3.67e-2]	[4.98e-3, 2.10e-2]	[1.80e-4, 7.23e-4]	[4.13e-3, 1.75e-2]	[4.21e-3, 1.78e-2]	[1.61e-2, 6.84e-2]
1	[5.86e-6, 9.57e-5]	[7.80e-6, 9.32e-5]	[2.53e-1, 1.82e+0]	[1.02e-2, 4.29e-2]	[1.60e-2, 6.75e-2]	[8.93e-3, 3.76e-2]	[2.94e-4, 1.56e-3]	[8.21e-3, 3.31e-2]	[7.49e-3, 3.02e-2]	[2.25e-2, 9.06e-2]
2	[3.05e-6, 1.22e-4]	[3.45e-6, 9.27e-5]	[3.33e-1, 2.23e+0]	[1.35e-2, 5.34e-2]	[2.24e-2, 8.87e-2]	[1.33e-2, 5.29e-2]	[5.03e-4, 2.28e-3]	[1.16e-2, 4.67e-2]	[1.04e-2, 4.17e-2]	[2.83e-2, 1.14e-1]
3	[8.26e-6, 2.28e-4]	[1.01e-5, 1.75e-4]	[5.66e-1, 3.67e+0]	[1.95e-2, 7.72e-2]	[2.82e-2, 1.12e-1]	[1.73e-2, 6.84e-2]	[7.44e-4, 3.36e-3]	[1.59e-2, 6.10e-2]	[1.35e-2, 5.17e-2]	[3.14e-2, 1.20e-1]
4	[2.34e-5, 5.14e-4]	[3.15e-5, 3.62e-4]	[9.45e-1, 6.50e+0]	[2.63e-2, 9.89e-2]	[3.10e-2, 1.17e-1]	[2.51e-2, 9.43e-2]	[1.05e-3, 4.32e-3]	[2.39e-2, 8.79e-2]	[1.98e-2, 7.26e-2]	[4.08e-2, 1.50e-1]
5	[2.99e-5, 9.75e-4]	[3.00e-5, 5.13e-4]	[1.09e+0, 8.15e+0]	[3.55e-2, 1.26e-1]	[4.04e-2, 1.44e-1]	[3.32e-2, 1.18e-1]	[1.21e-3, 5.53e-3]	[3.19e-2, 1.06e-1]	[2.71e-2, 9.02e-2]	[5.14e-2, 1.71e-1]
6	[2.62e-5, 1.01e-3]	[2.96e-5, 5.47e-4]	[1.32e+0, 8.58e+0]	[4.26e-2, 1.38e-1]	[5.09e-2, 1.65e-1]	[4.79e-2, 1.55e-1]	[1.81e-3, 7.40e-3]	[4.43e-2, 1.35e-1]	[3.87e-2, 1.17e-1]	[6.93e-2, 2.10e-1]
7	[3.17e-5, 1.11e-3]	[3.16e-5, 5.68e-4]	[1.25e+0, 8.37e+0]	[5.65e-2, 1.66e-1]	[6.85e-2, 2.02e-1]	[6.84e-2, 2.01e-1]	[2.76e-3, 9.42e-3]	[5.95e-2, 1.69e-1]	[5.55e-2, 1.57e-1]	[1.06e-1, 2.99e-1]
8	[2.96e-5, 1.41e-3]	[4.33e-5, 8.71e-4]	[2.72e+0, 1.78e+1]	[9.47e-2, 2.61e-1]	[1.05e-1, 2.89e-1]	[9.74e-2, 2.68e-1]	[4.07e-3, 1.17e-2]	[8.35e-2, 2.28e-1]	[7.85e-2, 2.14e-1]	[1.40e-1, 3.82e-1]
9	[4.41e-5, 1.38e-3]	[6.96e-5, 9.18e-4]	[4.00e+0, 1.86e+1]	[1.19e-1, 3.18e-1]	[1.39e-1, 3.72e-1]	[1.52e-1, 4.05e-1]	[5.46e-3, 1.53e-2]	[1.44e-1, 3.89e-1]	[1.24e-1, 3.35e-1]	[1.89e-1, 5.10e-1]
10	[1.70e-4, 3.10e-3]	[1.85e-4, 2.21e-3]	[4.84e+0, 2.52e+1]	[1.64e-1, 4.31e-1]	[1.87e-1, 4.92e-1]	[1.99e-1, 5.22e-1]	[4.68e-3, 1.64e-2]	[1.95e-1, 4.90e-1]	[1.77e-1, 4.46e-1]	[2.39e-1, 6.02e-1]
11	[3.28e-4, 3.81e-2]	[4.61e-4, 2.00e-2]	[8.53e+0, 2.80e+1]	[2.47e-1, 3.49e-1]	[2.35e-1, 3.31e-1]	[2.70e-1, 3.82e-1]	[6.83e-3, 3.04e-2]	[2.34e-1, 2.34e-1]	[2.38e-1, 2.38e-1]	[1.00e+0, 1.00e+0]

Table 8: Range of amplification factors for each intermediate operation over 2,470 non-outlier RTE training samples, where i denotes the Transformer encoder index. Each entry represents $[\min, \max]$. The notation $\mathtt{e}\pm\mathtt{n}$ denotes $\times 10^{\pm n}$.

sions, their direct implementation can incur substantial computational and modulus overhead under homomorphic encryption. We therefore adopt HE-friendly implementations based on polynomial approximation and power-of-two scaling.

E.1 SOFTMAX

For the homomorphic evaluation of Softmax, we largely follow the implementation strategy of THOR. Specifically, we use the parameters

$$M = 44, \qquad \delta_1 = 2, \qquad \delta_2 = 2^3. \tag{9}$$

Our implementation differs from THOR in that we use an inverse-square-root operation (`InverseSqrt`) instead of the inverse operation (`Inverse`). The `InverseSqrt` function is approximated by a degree-63 polynomial. The exponential function required for Softmax is approximated by a degree-15 polynomial.

These polynomial approximations allow the nonlinear operations involved in Softmax to be evaluated efficiently using only additions and multiplications, which are directly supported by the underlying HE scheme.

E.2 LAYER NORMALIZATION

We also employ an HE-efficient implementation of Layer Normalization. For the BERT hidden dimension of 768, the conventional mean is computed as

$$\mu = \frac{1}{768} \sum_{i=1}^{768} x_i. \tag{10}$$

Directly representing the division by 768 is relatively expensive in our HE implementation. Instead, using $768 = 3 \times 256$, we compute the scaled mean

$$3\mu = \frac{1}{256} \sum_{i=1}^{768} x_i. \tag{11}$$

This transformation is particularly advantageous in our 30-graft system because $256 = 2^8$. Therefore, division by 256 can be realized using only an 8-bit scale adjustment, whereas the conventional implementation based on division by 768 requires approximately 30–40 bits.

The same scaling strategy is applied to the variance computation. The conventional variance is

$$\sigma^2 = \frac{1}{768} \sum_{i=1}^{768} x_i^2 - \mu^2. \tag{12}$$

Multiplying the variance by 9 gives

$$\begin{aligned} 9\sigma^2 &= \frac{9}{768} \sum_{i=1}^{768} x_i^2 - 9\mu^2 \\ &= \frac{3}{256} \sum_{i=1}^{768} x_i^2 - (3\mu)^2. \end{aligned} \tag{13}$$

Hence, both the mean and variance calculations can exploit division by the power-of-two factor $256 = 2^8$, substantially reducing the scale consumption required for these operations.

Importantly, this scaling does not change the result of the normalization. Using the scaled mean 3μ and scaled variance $9\sigma^2$, we obtain

$$\begin{aligned} \frac{3x_i - 3\mu}{\sqrt{9\sigma^2}} &= \frac{3(x_i - \mu)}{3\sigma} \\ &= \frac{x_i - \mu}{\sigma}. \end{aligned} \tag{14}$$

Δ	Δ_{CTS}	$\Delta_{\mathrm{EvalMod}}$	Δ_{STC}	Recovered level	BTS error
24	36×4	42×7	26×4	$25 + 4/30$	6.97×10^{-4}
31	41×4	47×7	28×4	$22 + 29/30$	3.30×10^{-8}

Table 9: Bootstrapping configurations used in our experiments. Here, $b \times n$ denotes n moduli of b bits each.

Therefore, the proposed scaled computation is mathematically equivalent to the original Layer Normalization while being more efficient for homomorphic evaluation.

Finally, the reciprocal square-root function required for normalization, $\texttt{one_over_sqrt}(\cdot)$, is approximated by a degree-31 polynomial. Together with the power-of-two reformulation of the mean and variance, this reduces the scale and modulus requirements of Layer Normalization without changing its normalized output.

F Bootstrapping Configurations and Modulus Levels

This appendix provides additional details on the bootstrapping configurations and ciphertext modulus levels for the experiments presented in Sections 6.1 and 6.2. Appendix F.1 describes the bootstrapping configurations used in our experiments, including their parameter settings, remaining modulus, and bootstrapping error. Appendices F.2 and F.3 present the locations and numbers of bootstrapping operations for the experiments in Sections 6.1 and 6.2, respectively, together with the corresponding evolution of the ciphertext modulus levels.

F.1 Bootstrapping

We configure two bootstrapping parameter sets based on the error requirements determined by the optimized scales in Table 2. Specifically, the target bootstrapping error is determined by the required numerical accuracy at each bootstrapping location. For each target error, we optimize the bootstrapping parameters to recover the highest possible ciphertext level while keeping the bootstrapping error below the target. Table 9 summarizes the resulting configurations.

The configuration with $\Delta = 24$ is used after LN1 and LN2, whereas the configuration with $\Delta = 31$ is used during Softmax evaluation. The resulting configurations recover ciphertext levels of $25 + 4/30$ and $22 + 29/30$, respectively. In comparison, the bootstrapping configurations adopted by THOR and PowerFormer provide 13 and 11 multiplicative levels, respectively, between bootstrapping operations. Thus, our error-aware bootstrapping configurations provide a substantially larger level budget for subsequent homomorphic computation while satisfying the required numerical accuracy. Together with the reduced modulus consumption achieved by scale optimization, this increased level budget reduces the need for subsequent bootstrapping operations. The resulting bootstrapping locations and counts are detailed in Appendices F.2 and F.3.

F.2 Bootstrapping in the Experiment of Section 6.1

Our scale optimization reduces modulus consumption throughout the standard Transformer, while the optimized bootstrapping configurations in Appendix F.1 allow each bootstrapping operation to recover a higher ciphertext level. Consequently, more homomorphic computation can be performed between consecutive bootstrapping operations.

After applying the real–imaginary combination technique, our implementation requires 10 bootstrapping operations per encoder. The original THOR paper does not report a directly comparable total number of bootstrapping operations per encoder. Nevertheless, bootstrapping constitutes a substantial portion of the computational cost in THOR. A recent implementation based on THOR's nonlinear approximation configuration in ATLAS Xie et al. (2026) reports 732 bootstrapping operations for a full 12-layer BERT-Base forward pass. Since this count is obtained within the ATLAS FHE framework rather than the original THOR implementation, we report it only as a reference and do not use it for a direct quantitative comparison with our bootstrapping count.

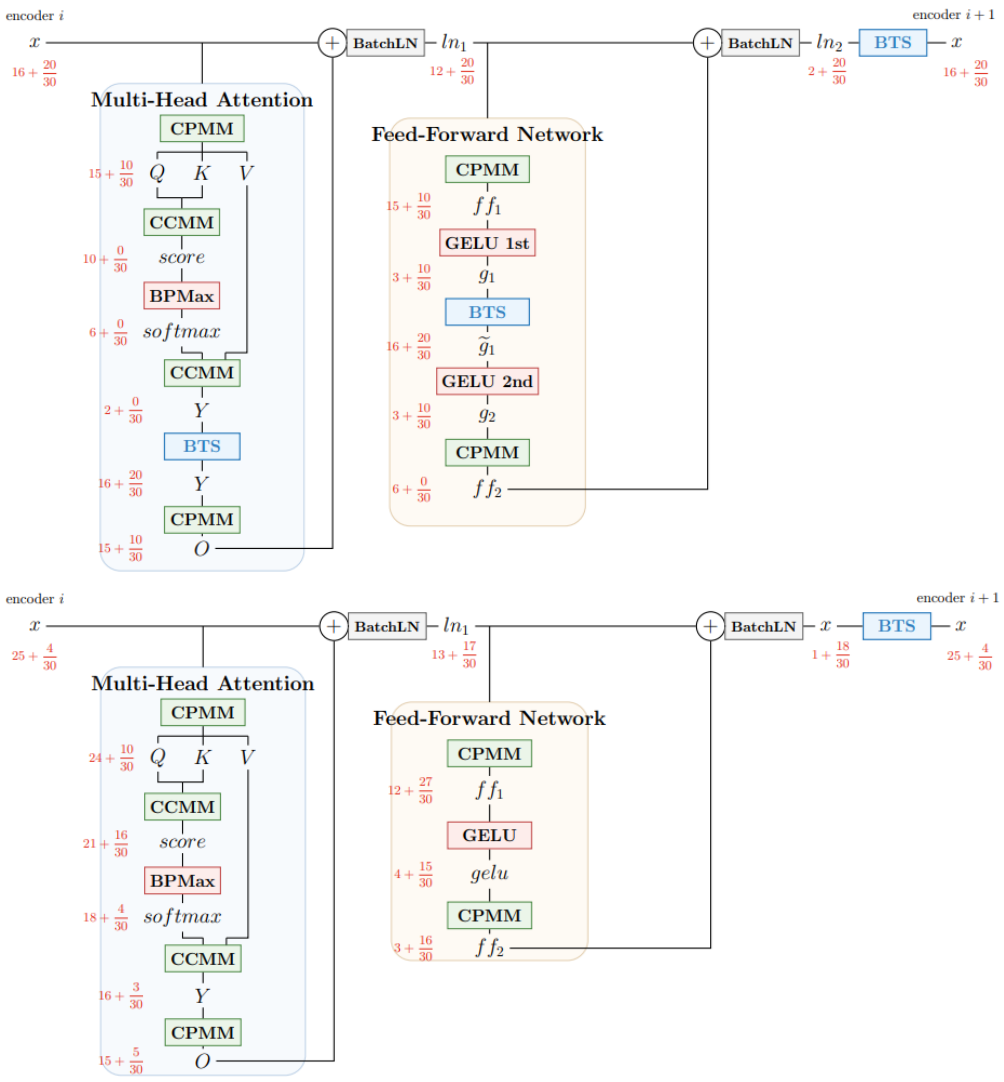

Figure 2: Ciphertext-level evolution throughout a single Transformer encoder for PowerFormer (top) and our method (bottom). Bootstrapping locations are indicated in each configuration.

F.3 Bootstrapping in the Experiment of Section 6.2

Figure 2 compares the ciphertext-level evolution and bootstrapping locations for PowerFormer and our method. In PowerFormer, bootstrapping is performed after Y, during GELU, and after LN2, requiring 2, 6, and 2 bootstrapping operations, respectively, after applying the real–imaginary combination technique. Thus, PowerFormer requires 10 bootstrapping operations per encoder.

In contrast, our scale optimization reduces modulus consumption, while the optimized bootstrapping configuration in Appendix F.1 allows bootstrapping to recover a higher ciphertext level. Consequently, bootstrapping is required only after LN2, corresponding to 2 operations per encoder after applying the same real–imaginary combination technique. This reduces the bootstrapping count from 10 to 2 per encoder, corresponding to an 80% reduction.