CGF-SOFTMAX: A CUMULANT-BASED SOFTMAX REFORMULATION FOR EFFICIENT INFERENCE UNDER HOMOMORPHIC ENCRYPTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Homomorphic encryption (HE) is a prominent framework for privacy-preserving machine learning, enabling inference directly on encrypted data. However, softmax, widely used in deep neural networks including transformers, remains challenging to evaluate under HE due to its multivariate structure, the large dynamic range induced by exponentiation, and costly division. In this paper, we propose CGF-softmax, which reformulates the softmax denominator through the cumulant generating function (CGF). By eliminating both homomorphic division and explicit maximum subtraction, this reformulation substantially reduces multiplicative depth while preserving key properties of softmax. Extensive experiments on Vision Transformers and large language models show that CGF-softmax provides an efficient and accurate approximation of softmax in encrypted inference. In particular, it achieves inference accuracy close to that of high-depth exact methods, while requiring substantially lower computational cost through reduced multiplicative depth. Integrated into a state-of-the-art multi-GPU FHE inference framework, replacing only the softmax operation reduces end-to-end encrypted BERT-Base inference latency by $36.3\,\%$ (one GPU) and $39.6\,\%$ (two GPUs), by eliminating all bootstrapping inside the softmax.

1 INTRODUCTION

The growing use of machine learning as a service (MLaaS) has raised privacy concerns, motivating privacy-preserving machine learning (PPML) (Al-Rubaie & Chang, 2019; Riazi et al., 2019). Among PPML approaches, interactive protocols such as secure multi-party computation (MPC) (Evans et al., 2018) require client participation and communication during inference, whereas homomorphic encryption (HE) (Rivest et al., 1978; Gentry, 2009) enables strictly non-interactive inference by allowing the server to compute directly on encrypted data. In this work, we focus on the strictly non-interactive HE setting, where inference can be fully outsourced without client interaction after ciphertext submission.

A primary challenge of HE-based PPML lies in the implementation of non-polynomial operations. Most HE schemes natively support only addition and multiplication, rendering the direct evaluation of non-polynomial functions infeasible. As a result, non-polynomial activation functions are typically approximated using polynomials, which can be evaluated homomorphically. However, achieving high approximation accuracy generally requires high-degree polynomials, leading to excessive computational latency and noise growth. This leads to increased multiplicative depth, which consumes the limited level budget and necessitates costly bootstrapping operations. To reduce this overhead, prior work has often resorted to simplified surrogate functions or low-degree polynomial approximations, at the cost of degraded model accuracy (Lou et al., 2020; Lou & Jiang, 2021; Chen et al., 2022; Rho et al., 2025).

Among these non-polynomial operations, the *softmax* function is widely used in deep neural networks to produce normalized outputs, such as class probabilities and attention weights. Its efficient evaluation is particularly important in transformer-based architectures (Vaswani et al., 2017; Brown et al., 2020; Dosovitskiy et al., 2021; Dubey et al., 2024), where it plays a central role in the self-attention mechanism. Unlike univariate activation functions, softmax has a multivariate structure in which each output depends on the entire input vector through a shared normalization term. Eval-

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.

uating softmax in a strictly non-interactive HE setting therefore constitutes a major computational bottleneck, as its standard formulation involves the exponential function and division, which are inherently expensive to approximate homomorphically (Hong et al., 2022; Cho et al., 2024). Moreover, numerical stabilization techniques commonly employed in plaintext inference, such as subtracting the maximum input value, require comparison operations that are not natively supported by CKKS (Cheon et al., 2017) and introduce additional computational overhead.

In response to this challenge, existing approaches to homomorphic softmax evaluation largely fall into two categories. Methods commonly referred to as *softmax approximation* aim to evaluate the softmax function with high precision under HE, but are typically computationally expensive due to high-degree polynomial approximations for the exponential function and division (Hong et al., 2022; Cho et al., 2024; Castro et al., 2025; Lim et al., 2025). To reduce this computational cost, an alternative strategy denoted as *softmax replacement* replaces the original softmax function with simpler surrogate functions (Zimerman et al., 2024a;b; Park et al., 2025). While such replacements can be effective for relatively simple natural language processing (NLP) tasks, our experiments show that they suffer from substantial accuracy degradation in more demanding settings, including large-scale image classification and complicated NLP tasks. These limitations highlight the need for a softmax evaluation method that significantly reduces computational cost while preserving high accuracy on complex tasks.

In this paper, we propose **CGF-softmax**, a novel method that *reformulates* the softmax function by leveraging the cumulant generating function (CGF). CGF-softmax addresses the aforementioned challenges by eliminating the need for homomorphic division as well as explicit maximum subtraction commonly used for numerical stabilization. By removing these computationally expensive operations, particularly the comparison-heavy max operation, our method substantially reduces the multiplicative depth compared to existing softmax approximation approaches. This depth reduction is critical as it minimizes the need for bootstrapping, directly translating to significant savings in computational cost. We further derive key theoretical properties of CGF-softmax and provide an asymptotic bound on its approximation error relative to the exact softmax function. Experimental results demonstrate that CGF-softmax achieves a significant reduction in the level consumption, outperforming the softmax approximation baseline (Cho et al., 2024). Moreover, CGF-softmax consistently maintains high accuracy across diverse tasks, achieving less than a $1\,\%$ accuracy drop for various models and datasets. Finally, we demonstrate the system-level impact of CGF-softmax: replacing only the softmax in Cerium (Jayashankar et al., 2026), a recently released multi-GPU FHE framework, reduces end-to-end encrypted BERT-Base latency by by $36.3\,\%$ (one GPU) and $39.6\,\%$ (two GPUs).

Our contributions are summarized as follows:

- **CGF-softmax Formulation:** We propose CGF-softmax, a novel reformulation of the softmax function that eliminates the need for homomorphic division and maximum subtraction, thereby substantially reducing computational cost.
- **Theoretical Analysis:** We provide a theoretical analysis establishing key properties of CGF-softmax and an asymptotic bound on its approximation error.
- **Experimental Validation:** We demonstrate through extensive experiments on Vision Transformers and large language models that CGF-softmax achieves inference accuracy close to exact softmax while significantly reducing inference cost under homomorphic encryption.[1]

2 Background and Related Works

2.1 Fully Homomorphic Encryption

Fully homomorphic encryption (FHE) is a cryptographic primitive that enables computations to be performed directly on encrypted data without decryption. Among existing schemes, we utilize the CKKS scheme (Cheon et al., 2017), which supports approximate arithmetic operations over encrypted real numbers. Specifically, CKKS employs a packing technique that encrypts a vector of values into a single ciphertext. We denote the number of these values, i.e., the number of slots in a ciphertext, as s. Supported homomorphic operations include element-wise addition (Add), cyclic rotation (Rot), and multiplication. Regarding multiplication, we distinguish between *Ciphertext–*

[1]The implementation of CGF-softmax is available at https://github.com/cgf-anon/cgf-anon.

Plaintext Multiplication (PMult), which involves a plaintext constant, and the computationally expensive *Ciphertext–Ciphertext Multiplication* (CMult). As a leveled FHE scheme, CKKS assigns a finite budget of multiplicative levels L to each ciphertext. Each CMult (and non-integer PMult) consumes one level, and the cumulative consumption determines the circuit *depth*. Once the level budget is exhausted, *bootstrapping* (Boot) is required to refresh the ciphertext, enabling further computation at the cost of significant overhead. Consequently, we evaluate HE algorithmic efficiency based on the multiplicative depth and the counts of dominant operations, specifically CMult and Rot.

2.2 Homomorphic Softmax Evaluation

We first recall the standard softmax function in the plaintext setting. For an input vector $\mathbf{x} = (x_1, \cdots, x_n)$, the i-th component is defined as

$$\text{softmax}(\mathbf{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} = \frac{\exp(x_i - x_{\max})}{\sum_{j=1}^{n} \exp(x_j - x_{\max})}, \tag{1}$$

where $i \in \{1, \ldots, n\}$ and $x_{\max} = \max_{i \in \{1,\ldots,n\}} x_i$. In standard floating-point implementations, the second form is typically used as a numerically stable formulation to prevent overflow from the exponential function. Evaluating softmax directly in the HE setting presents two major challenges:

1. **Overflow and Maximum Subtraction:** The rapid growth of the exponential function renders softmax vulnerable to numerical overflow when processing large input values. In the homomorphic setting, larger ranges necessitate polynomial approximations over wider intervals, further increasing computational cost and exacerbating approximation error (Cheon et al., 2022; Cho et al., 2024). To ensure numerical stability, the input vector to softmax is typically shifted using the maximum operation, as shown in Eq. (1). However, the maximum operation is not natively supported by HE schemes and must be approximated, which incurs substantial multiplicative depth.
2. **Division Operation:** Softmax inherently involves a division operation for normalization. Since HE schemes do not natively support division, prior work (Hong et al., 2022; Lim et al., 2025; Castro et al., 2025) typically addresses this challenge by employing polynomial approximations of the reciprocal function $1/x$, such as Chebyshev polynomial approximation (Cheney, 1966) or Goldschmidt's iterative algorithm (Goldschmidt, 1964), leading to increased multiplicative depth.

2.3 Prior Work

In the strictly non-interactive setting, prior work can be broadly categorized into two classes: softmax approximation and softmax replacement. While the former directly approximates the softmax function using polynomials, the latter replaces softmax with simpler surrogate functions that are also realized through polynomial approximations.

Softmax Approximation. Prior work on *softmax approximation* aims to approximate the original softmax function using polynomial functions. Following an early work on homomorphic softmax approximation (Hong et al., 2022), subsequent works focus on mitigating exponential overflow.

The first strategy stabilizes computation by subtracting the maximum value $x_{\max}$. HETAL (Lee et al., 2023) computes the exact maximum; however, this incurs high computational cost due to comparison operations, which should be approximated by polynomials. Subsequent frameworks therefore adopt heuristic or statistical estimates to approximate $x_{\max}$ more efficiently. EncryptedLLM (Castro et al., 2025) uses a fixed empirical maximum derived from training data. Recognizing the instability of fixed values, Tricycle (Lim et al., 2025) and ARION (Yang et al., 2025) estimate an upper bound on the *expected* maximum based on statistical assumptions.

The second strategy addresses overflow by compressing the input interval. Cho et al. (2024) introduce the *normalize-and-square* algorithm, which evaluates softmax on scaled-down inputs to ensure numerical stability and subsequently recovers the original outputs via the iterative identity:

$$\text{softmax}\left(\mathbf{x}/2^{k-1}\right)_i = \frac{\text{softmax}\left(\mathbf{x}/2^{k}\right)_i^2}{\sum_{j=1}^{n} \text{softmax}\left(\mathbf{x}/2^{k}\right)_j^2}. \tag{2}$$

THOR (Moon et al., 2025) adopts this framework, differing primarily in the choice of approximation algorithms for the exponential function and division.

Table 1: Comparison with prior softmax approximation methods. CGF-softmax eliminates both max and division operations while remaining adaptive to input statistics. (R: Required, NR: Not Required)

Operations	HETAL (Lee et al., 2023)	EncryptedLLM (Castro et al., 2025)	Tricycle (Lim et al., 2025) ARION (Yang et al., 2025)	Cho et al. (2024) THOR (Moon et al., 2025)	**CGF-softmax**
Max	Comparison-Based Approx.	Fixed (Empirical)	Expected Max Upper Bound	Fixed (Domain Scaling)	**NR**
Division	R	R	R	R	**NR**
Data-Adaptivity	**Yes**	No	**Yes**	No	**Yes**

Crucially, regardless of the overflow mitigation strategy, all aforementioned methods rely on polynomial approximations for both the exponential function and the computationally expensive homomorphic division, resulting in increased multiplicative depth.

Softmax Replacement. Distinct from softmax approximation, *softmax replacement* methods substitute the computationally expensive softmax operation with simpler surrogate functions. Several works replace softmax with element-wise activation functions, including standard activations (e.g., ReLU and GELU), their polynomial variants (e.g., squared ReLU), or activations augmented with auxiliary neural networks (Zimerman et al., 2024b; Chen et al., 2022). Rho et al. (2025) pursue an alternative direction by adopting Gaussian kernels in place of softmax. Another line of work replaces only the exponential term $\exp(x_i)$ with power functions or their shifted variants (Li et al., 2023; Luo et al., 2024; Zimerman et al., 2024a), while retaining the division operation for normalization, e.g., Power-softmax (Zimerman et al., 2024a), $x_i^p / \sum_j x_j^p$ with a positive even p. BPMax from Powerformer (Park et al., 2025) further eliminates the division operation by replacing the denominator with a fixed empirical value, e.g., $(x_i + c)^p$ with a positive odd p.
However, these softmax replacement methods face inherent limitations: they either suffer from accuracy degradation (Zimerman et al., 2024b; Rho et al., 2025) or provide limited speedup due to the retained division operation (Li et al., 2023; Luo et al., 2024; Zimerman et al., 2024a). Furthermore, even recent replacement methods (Park et al., 2025; Luo et al., 2024; Rho et al., 2025) are primarily evaluated on small-scale models such as BERT (Devlin et al., 2019) and benchmarks with small label spaces (e.g., binary or ternary classification). The effectiveness of such methods on larger models and more challenging benchmarks remains underexplored.

3 Softmax Reformulation via Cumulant Generating Function

In this section, we introduce a novel reformulation of the softmax function based on the CGF that enables efficient evaluation under HE.

3.1 CGF-softmax

CGF-softmax. We introduce CGF-softmax to address the challenges of softmax in HE described in Section 2.2. Given an input vector $\mathbf{x} = (x_1, \cdots, x_n)$, we interpret the softmax denominator as a scaled sample mean of exponential terms. Let X denote a random variable representing the distribution of input components, and let $M_X(t) \triangleq \mathbb{E}[\exp(tX)]$ denote its MGF. Under this interpretation, the softmax denominator $\sum_{i=1}^{n} \exp(x_i)$ is replaced by $nM_X(1)$ by substituting the sample mean with the ensemble mean. Accordingly, the i-th component of CGF-softmax is given by

$$\text{softmax}_{\text{CGF}}(\mathbf{x})_i = \frac{\exp(x_i)}{nM_X(1)}. \tag{3}$$

To make the structure explicit, we introduce the cumulant generating function (CGF) of X (Casella & Berger, 2024), defined as $K_X(t) \triangleq \ln M_X(t)$. Using the CGF, Eq. (3) is expressed in an explicit single-exponential form as

$$\text{softmax}_{\text{CGF}}(\mathbf{x})_i = \exp\left(x_i - K_X(1) - \ln n\right). \tag{4}$$

Moreover, $K_X(1)$ can be expanded as

$$K_X(1) = \sum_{j=1}^{\infty} \frac{\kappa_j}{j!}, \tag{5}$$

where κ_j denotes the j-th *cumulant* of X. This representation is particularly suitable for HE since cumulants, such as $\kappa_1 = \mu$ (mean) and $\kappa_2 = \sigma^2$ (variance) can be efficiently computed using basic HE operations.

Advantages of CGF-softmax. By reformulating softmax via CGF, CGF-softmax achieves both computational efficiency and improved approximation stability via a data-adaptive shift. This reformulation leads to the following key advantages:

1. CGF-softmax eliminates the need for a $\max$ operation. All input components are shifted by $K_X(1) + \ln n$ prior to exponential evaluation, which preserves numerical stability.
2. CGF-softmax removes the division operation and correspondingly reduces the multiplicative depth, as the normalization term is absorbed into the exponent of the single exponential.
3. CGF-softmax is adaptive to the input statistics through cumulants. This data-adaptivity leads to more stable approximation behavior across varying input samples.

As a result, CGF-softmax directly addresses the challenges of softmax evaluation in HE outlined in Section 2.2. Table 1 compares CGF-softmax with prior softmax approximation methods and summarizes the advantages of the proposed approach. In particular, it shows that CGF-softmax uniquely eliminates both $\max$ and division operations while remaining adaptive to input statistics.

3.2 Properties of CGF-softmax

In this subsection, we present key properties of CGF-softmax that reveal its advantages under HE.

(P1) Positivity. CGF-softmax is strictly *positive* for every component by Eq. (4).

This property is not automatic for power function-based softmax replacements. For example, BPMax (Park et al., 2025) can preserve ordering when p is a positive odd integer, but if the shift c is not large enough to ensure $x_i + c > 0$ for all i, it may produce nonpositive outputs.

(P2) Order and Pairwise-Ratio Preservation. A softmax replacement should preserve the *relative importance* of the input logits. CGF-softmax satisfies this requirement by preserving both logit ordering and the exact pairwise output ratios of the standard softmax.

Proposition 3.1. CGF-softmax preserves the relative ordering of the input logits and has the same pairwise output ratios as the standard softmax. Specifically, for all $i, j \in \{1, \dots, n\}$,

$$x_i > x_j \;\Rightarrow\; \mathrm{softmax}_{\mathrm{CGF}}(\mathbf{x})_i > \mathrm{softmax}_{\mathrm{CGF}}(\mathbf{x})_j, \tag{6}$$

and

$$\frac{\mathrm{softmax}(\mathbf{x})_i}{\mathrm{softmax}(\mathbf{x})_j} = \frac{\mathrm{softmax}_{\mathrm{CGF}}(\mathbf{x})_i}{\mathrm{softmax}_{\mathrm{CGF}}(\mathbf{x})_j} = \exp(x_i - x_j). \tag{7}$$

These properties follow directly from Eq. (4): the exponential numerator is strictly increasing, and the denominator $nM_X(1)$ is a positive constant common across all indices.
In contrast, power-function-based replacements preserve ordering only under suitable conditions and generally do not preserve the exact exponential pairwise ratios of the standard softmax. Power-softmax (Zimerman et al., 2024a) with even p preserves ordering only over nonnegative logits, since $x \mapsto x^p$ is not strictly increasing on $\mathbb{R}$, and its ratios are polynomial, $(x_i/x_j)^p$, when defined. BPMax (Park et al., 2025) preserves ordering when p is a positive odd integer and the denominator is a positive constant common across indices, but its ratios are also polynomial, $((x_i + c)/(x_j + c))^p$, rather than $\exp(x_i - x_j)$.

(P3) Shift Invariance. The standard softmax is *shift-invariant*, i.e., $\mathrm{softmax}(\mathbf{x}) = \mathrm{softmax}(\mathbf{x} - c)$ for any constant c. This property is commonly used to improve numerical stability (see Eq. (1)). CGF-softmax also preserves this shift-invariance property.

Proposition 3.2. CGF-softmax is shift-invariant. For any constant $c \in \mathbb{R}$,

$$\mathrm{softmax}_{\mathrm{CGF}}(\mathbf{x}) = \mathrm{softmax}_{\mathrm{CGF}}(\mathbf{x} - c). \tag{8}$$

Proof. The proof is given in Appendix A. □
In contrast, softmax replacement methods (Luo et al., 2024; Zimerman et al., 2024a; Park et al., 2025) are generally not shift-invariant.

(P4) Domain Scaling. Efficient evaluation of the softmax function under HE is challenging when the input values span a wide range, as polynomial approximations of the exponential function rapidly deteriorate outside a limited interval (Cheon et al., 2022). To address this issue, it is crucial to control the domain of the exponential function during evaluation.
A representative approach is the normalize-and-square strategy proposed by Cho et al. (2024), which scales the input domain to a small interval and subsequently restores the original scale via repeated

squaring (see Eq. (2)). While effective, applying this strategy to the standard softmax requires repeated division operations, which are particularly costly under HE.

In contrast, CGF-softmax naturally enables a much simpler *domain scaling* mechanism.

Proposition 3.3. The domain scaling of CGF-softmax by a factor of 2^k ($k \in \mathbb{N}$) is expressed as:

$$\text{softmax}_{\text{CGF}}(\mathbf{x})_i = \exp\left(\frac{x_i - K_X(1) - \ln n}{2^k}\right)^{2^k}. \tag{9}$$

The exponent is scaled by a factor of $1/2^k$ to ensure that the exponential function is evaluated over a significantly smaller domain. The original value is then recovered by squaring the result k times. Crucially, this domain scaling reduces the effective approximation interval of the exponential function by a factor of 2^k, while incurring only an additional multiplicative depth of $k+1$ (CMult: k, PMult: 1). Unlike the standard softmax, this procedure does not involve explicit normalization or division, enabling a substantially simpler and more efficient realization under HE.

3.3 Error Analysis

To evaluate the approximation accuracy of CGF-softmax, we analyze the *relative error* as in Cho et al. (2024).

Proposition 3.4. The relative error between CGF-softmax and softmax is given by

$$\eta = \frac{\|\text{softmax}(\mathbf{x}) - \text{softmax}_{\text{CGF}}(\mathbf{x})\|_\infty}{\|\text{softmax}(\mathbf{x})\|_\infty} = \left|1 - \frac{\frac{1}{n}\sum_{i=1}^n \exp(x_i)}{\mathbb{E}[\exp(X)]}\right|. \tag{10}$$

Proof. The proof is given in Appendix B. □

Importantly, the relative error η depends only on the ratio between the sample mean and the ensemble mean of $\exp(X)$, and is independent of the individual component index i.
To further quantify the concentration of the relative error, we first establish a Chebyshev-type bound without assuming independence or identical distributions.

Proposition 3.5. Let $Y_i = \exp(X_i)$, $\bar{Y}_n = n^{-1}\sum_{i=1}^n Y_i$, and $\mu_Y = \mathbb{E}[\exp(X)] \in (0, \infty)$. Assume $\mathbb{E}[Y_i^2] < \infty$ for all i. Then, for any $\delta > 0$,

$$P_\eta \triangleq P(\eta \geq \delta) \leq \min\left\{1, \frac{\text{Var}(\bar{Y}_n) + \left(\mathbb{E}[\bar{Y}_n] - \mu_Y\right)^2}{\delta^2 \mu_Y^2}\right\}. \tag{11}$$

If $X_1, \ldots, X_n$ are i.i.d., the bound simplifies to $P_\eta \leq \min\left\{1, \frac{\sigma_Y^2}{n\delta^2\mu_Y^2}\right\}$, where $\sigma_Y^2 = \text{Var}(\exp(X))$.

Proof. The proof is given in Appendix C. □

Under the additional finite third absolute central moment condition, we obtain a complementary bound on the error probability using the Berry–Esseen theorem (Berry, 1941; Esseen, 1942).

Proposition 3.6. Let $Y_i = \exp(X_i)$, and define $\mu_Y = \mathbb{E}[\exp(X_i)] = \mathbb{E}[Y_i]$, $\sigma_Y^2 = \text{Var}(Y_i) > 0$, and $\rho_Y = \mathbb{E}\left[|Y_i - \mu_Y|^3\right] < \infty$. If $X_1, \ldots, X_n$ are i.i.d., then, for any $\delta > 0$,

$$P_\eta \leq 2\left[1 - \Phi\left(\frac{\delta\mu_Y\sqrt{n}}{\sigma_Y}\right)\right] + \frac{C\rho_Y}{\sigma_Y^3\sqrt{n}}, \tag{12}$$

where $C > 0$ is a constant and $\Phi(\cdot)$ is the cumulative distribution function (CDF) of the standard normal distribution.

Proof. The proof is given in Appendix D. □

Since both the standard-normal tail term and the Berry–Esseen correction vanish as n increases, $P_\eta \to 0$ for any $\delta > 0$. Thus, the relative error η asymptotically vanishes in probability.

Table 2: Comparison with prior softmax approximation methods. CGF-softmax achieves significantly lower multiplicative depth compared to existing baselines even for larger-scale models (e.g., Llama-3.2-1B) and more complex benchmarks.

Method	Model	Model Size	Dataset	# Classes	Mult. Depth
EncryptedLLM (Castro et al., 2025)	GPT-2 Small GPT-2 Medium GPT-2 Large	124M 355M 774M	SST-2, WiC, PIQA, MNLI, ANLI, Social IQA, HellaSwag, ARC (Easy)	2, 3, 4	22 26 30
THOR (Moon et al., 2025)	BERT-Base	110M	SST-2, RTE, MRPC	2	30
Tricycle (Lim et al., 2025)	BERT-Tiny	4.4M	SST-2	2	17
ARION (Yang et al., 2025)	BERT-Tiny/Base	4.4M / 110M	SST-2, RTE, QNLI	2	14
Proposed	Llama-3.2-1B	1B	SST-2, Banking77, Clinc150	**2, 77, 150**	**8–9**
	ViT/DeiT (Tiny/Base)	5M / 86M	ImageNet-1k	**1,000**	**7–10**

Table 3: Algorithmic complexity comparison between CGF-softmax and Cho et al. (2024), both based on domain scaling by $1/2^k$. Here, n denotes the input dimension of softmax, and L represents the number of available multiplicative levels after bootstrapping.

Method	Depth	# CMult	# Rot	# Boot
Cho et al. (2024)	$\geq 8k+9$	$\geq 12k+58$	$2k\log_2 n$	$\geq \lfloor (8k+9)/L \rfloor$
Proposed	$k+6$	$k+10$	$2\log_2 n$	$\lfloor (k+6)/L \rfloor$

3.4 Practical Instantiation: Second-Order Cumulant Approximation

While Eq. (4) expresses CGF-softmax in terms of the full CGF $K_X(1)$, practical evaluation under HE requires a tractable approximation. We adopt the second-order cumulant approximation, $K_X(1) \approx \mu + \sigma^2/2$. Here, μ and σ^2 correspond to the first two cumulants κ_1 and κ_2, while higher-order terms $\kappa_j/j!$ for $j \geq 3$ are omitted. This second-order approximation requires only the mean and variance, which can be computed directly from the encrypted input. We adopt it as a practical accuracy–depth trade-off, with model adaptation used to recover downstream accuracy. Appendix E shows that fine-tuning reduces the dispersion of the third- and fourth-order contribution. This observation supports the practical use of the approximation in the evaluated setting, but does not constitute a general bound on the full cumulant-truncation error.

4 Experimental Results

4.1 Softmax Evaluation

Table 2 presents a comparative analysis of the multiplicative depth required by CGF-softmax and prior softmax approximation methods (Castro et al., 2025; Lim et al., 2025; Yang et al., 2025). We note that prior approaches are optimized for fixed experimental settings tailored to specific model architectures and datasets. As a result, a direct comparison under identical input intervals is inherently challenging due to differences in the evaluated models and datasets. Despite these discrepancies, the comparison in Table 2 highlights the *scalability* and *efficiency* of CGF-softmax. In particular, CGF-softmax achieves the lowest multiplicative depth, ranging from 7 to 10, even when applied to significantly larger-scale models (e.g., Llama-3.2-1B) and high-complexity benchmarks such as ImageNet-1k with 1,000 classes. In contrast, prior methods require substantially higher multiplicative depths while being evaluated on smaller models and tasks with fewer classes.

Next, distinct from the softmax approximation methods discussed above, we evaluate the algorithmic complexity of CGF-softmax against Cho et al. (2024). We adopt it as our primary softmax approximation baseline because, like our proposed method, it employs a *domain scaling* strategy that enables a generalizable and scalable framework capable of handling varying input intervals, rather than relying on fixed bounds or dataset-specific tuning. Table 3 presents a detailed comparison. To handle large input intervals $[-M, 0]$, Cho et al. (2024) mitigates numerical instability by scaling inputs by a factor of $1/2^k$. However, recovering the original scale requires the *normalize-and-square* algorithm (see Eq. (2)), which involves an iterative process that is computationally expensive. In practice, this typically requires $k \approx \lceil \log_2 M - \log_2(\ln n) \rceil$ iterations, where n denotes the input dimension of softmax. In contrast, CGF-softmax streamlines this process by directly exploiting Eq. (9). This enables efficient handling of large input intervals while significantly reducing computational overhead. These advantages are reflected in the reduced multiplicative depth reported in Table 3. Implementation details of CGF-softmax are provided in Appendix F.

We also measure the execution time of softmax evaluation under HE using the `desilofhe` (DESILO Inc., 2025) library. For this experiment, We set the slot size as $s = 2^{15}$, and the available multiplicative level after bootstrapping as $L = 10$. The input size is set to a 256×256 matrix with an input interval of $[-128, 0]$, and all timings are measured using 8 CPU threads. As shown in Table 4, the observed latency reduction is primarily attributed to the reduced multiplicative depth, which reduces or eliminates the need for expensive bootstrapping, as well as the reduced number of polynomial approximation steps, which significantly lowers the number of CMult operations.

Table 4: Breakdown of CPU runtime (in seconds) for a single softmax evaluation on a 256×256 input matrix. The reduction in total latency is primarily attributed to the elimination of bootstrapping (Boot) and the reduced number of ciphertext multiplications (CMult). In our setting, the average runtime per operation for Add, PMult, CMult, Rot, and Boot is approximately 0.0014, 0.0021, 0.089, 0.059, and 14, respectively. Values are reported as mean $\pm$ standard deviation over 100 runs.

Method	Add	PMult	CMult	Rot	Boot	**Total**
Cho et al. (2024)	2.46 ± 0.08	1.20 ± 0.04	11.65 ± 0.26	1.48 ± 0.06	90.86 ± 2.99	107.65 ± 3.16
Proposed	0.32 ± 0.02	0.12 ± 0.01	$\mathbf{1.95 \pm 0.10}$	0.83 ± 0.06	$\mathbf{0.00 \pm 0.00}$	$\mathbf{3.25 \pm 0.15}$

Additionally, since CGF-softmax eliminates homomorphic division, it requires fewer homomorphic operations than the standard softmax, resulting in lower evaluation noise. In our empirical analysis, CGF-softmax achieves a total evaluation error of 3.05×10^{-10} at multiplicative depth 7, which is lower than that of the standard softmax formulation. This lower-noise behavior is particularly beneficial for applying it to deeper HE neural networks, where preserving the noise budget is critical. A detailed empirical comparison of the noise behavior is provided in Appendix G.

4.2 Model Accuracy Evaluation

4.2.1 Experimental Setup

Evaluation Scope & Objectives. To validate the scalability of CGF-softmax to large-scale architectures and its ability to handle complex tasks with high-dimensional output spaces, we consider the following evaluation (1) Llama-3.2-1B, sourced from the Hugging Face `transformers` library (Wolf et al., 2020), evaluated on Clinc150 (Larson et al., 2019), Banking77 (Casanueva et al., 2020), and SST-2 (Wang et al., 2018), which involve 150, 77, and 2 classes, respectively; and (2) Vision Transformers (ViT/DeiT-Base/Tiny) implemented using the `timm` library (Wightman, 2019), evaluated on ImageNet-1k (Dosovitskiy et al., 2021) (1,000 classes).

Baselines. We select representative baselines from both the softmax approximation and replacement categories. For *softmax approximation*, we use Cho et al. (2024) as the primary baseline. Unlike prior approaches that are optimized for specific experimental setting, this method provides a generalized framework capable of handling arbitrary input intervals, making it suitable for evaluation across diverse benchmarks. For *softmax replacement*, we use Batch Power-Max (BPMax) from Powerformer (Park et al., 2025), which eliminates division to achieve low circuit depth and was reported to incur negligible inference-accuracy degradation on BERT models.

Training Methodology. To recover the inference accuracy affected by homomorphic softmax evaluation, prior replacement baselines (BPMax) require an additional training phase. For a fair comparison, we fine-tune CGF-softmax using the same training framework as BPMax. Details of the training configurations and hyperparameters are provided in Appendix I. Notably, CGF-softmax converges faster during this adaptation process, leading to substantially lower training cost. For Llama-3.2-1B, CGF-softmax reaches its best performance within 5 epochs, whereas BPMax requires 20 epochs. For Vision Transformers, we further reduce the training cost by applying attention-only fine-tuning. We also present an *alternative pipeline* that removes this additional training cost by incorporating CGF-softmax directly into the initial Supervised Fine-Tuning (SFT) phase. Details are provided in Appendix J.

Low-degree Variant. We further introduce a *low-degree variant* of CGF-softmax, designed for scenarios that prioritize maximal efficiency with minimal multiplicative depth. Unlike the standard setting, this variant modifies the approximation stage by replacing the exact exponential function with a low-degree polynomial during fine-tuning, rather than approximating a fine-tuned model at inference time. Details of the exponential approximation schemes are provided in Appendix H.

4.2.2 Inference Accuracy

We analyze the inference accuracy results presented in Figures 1 and 2. In these figures, *exact softmax* denotes the accuracy of the original pre-trained model using the standard softmax function in the plaintext domain. The objective of HE-based softmax methods is to preserve the plaintext accuracy as closely as possible, while minimizing the required multiplicative depth.

Figures 1 and 2 compare the accuracy–depth trade-offs of CGF-softmax and the baseline methods. The two existing baselines occupy opposing extremes: Cho et al. (2024) attains near-exact accuracy but requires prohibitive multiplicative depths of up to 52, while BPMax (Park et al., 2025) achieves minimal depth of 2 at the cost of substantial accuracy degradation. CGF-softmax provides a favorable accuracy–depth trade-off: it achieves accuracy within one percentage point of the Cho et al. (2024) baseline while reducing multiplicative depth by a factor of 4–6, and it outperforms BPMax (Park et al., 2025) by large accuracy margins at comparable or only slightly higher depth. Notably, even the low-degree variant of CGF-softmax–which operates at the same depth as BPMax (Park et al., 2025) in the Llama setting–outperforms it across all benchmarks.

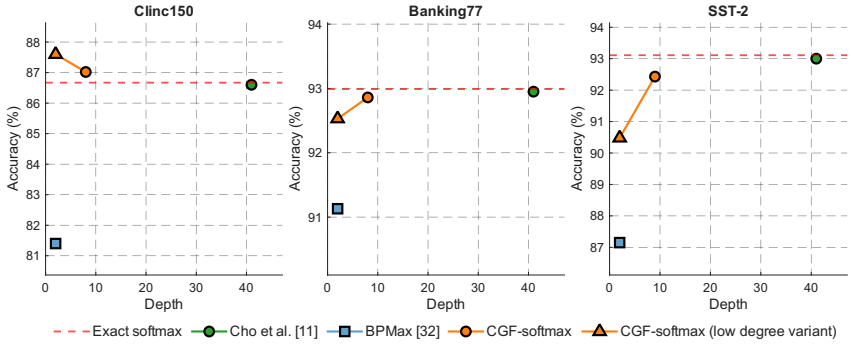

Figure 1: Cost–accuracy trade-off for Llama-3.2-1B across NLP benchmarks (Clinc150, Banking77, and SST-2). CGF-softmax provides a favorable accuracy–depth trade-off, achieving accuracy close to that of Cho et al. (2024) at substantially lower multiplicative depth, while outperforming BPMax (Park et al., 2025) in accuracy at comparable depths.

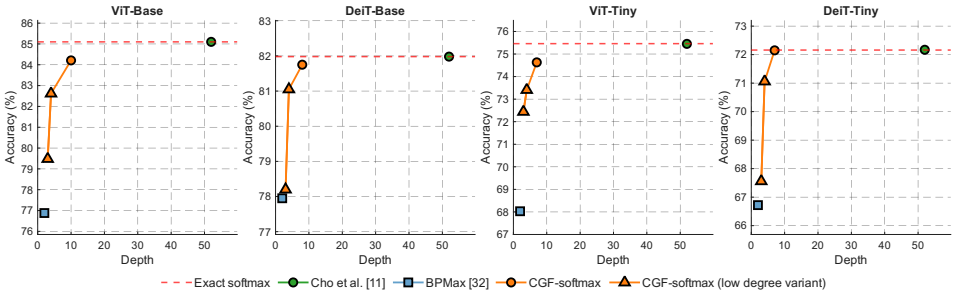

Figure 2: Cost–accuracy trade-off for ViT and DeiT models on the ImageNet-1k dataset. CGF-softmax provides a favorable accuracy–depth trade-off, achieving accuracy close to that of Cho et al. (2024) at substantially lower multiplicative depth, while outperforming BPMax (Park et al., 2025) in accuracy at comparable depths.

4.3 End-to-end Integration and Latency Evaluation

To assess whether the circuit-level savings of CGF-softmax translate into end-to-end acceleration, we integrate it into Cerium (Jayashankar et al., 2026), the first publicly available FHE framework supporting single- and multi-GPU execution. We replace the native softmax implementation in Cerium's BERT-Base pipeline with CGF-softmax while retaining the non-softmax model operations. Model adaptation and implementation details are provided in Appendix K. We measure end-to-end inference latency on one and two NVIDIA RTX PRO 6000 Blackwell GPUs. As shown in Table 5, CGF-softmax reduces the average end-to-end latency by 36.3 % and 39.6 % on one and two GPUs, respectively, relative to Cerium's native softmax implementation. These reductions correspond to speedups of $1.57\times$ and $1.66\times$, respectively. The latency breakdown in Appendix K shows that the gains primarily arise from eliminating bootstrapping within the softmax circuit.

Table 5: End-to-end encrypted BERT-Base inference latency (in seconds) on one and two NVIDIA RTX PRO 6000 Blackwell GPUs. Reductions and speedups are relative to Cerium's native softmax implementation with the same GPU count.

# of GPU	Cerium (Jayashankar et al., 2026)	Cerium with CGF-softmax	Latency reduction	Speedup
1	49.48 s	31.51 s	36.3%	1.57×
2	31.96 s	19.30 s	39.6%	1.66×

5 Conclusion

In this paper, we introduce CGF-softmax, a novel reformulation of the softmax function under HE. CGF-softmax reformulates the softmax denominator to eliminate computationally expensive operations such as maximum subtraction and homomorphic division. We establish key properties of the full-CGF formulation and derive asymptotic bounds on its approximation error. Experiments on Llama-3.2-1B and Vision Transformers show that, after adaptation, the standard CGF-softmax configuration maintains accuracy within one percentage point of the exact-softmax baseline. Furthermore, integrating CGF-softmax into Cerium's BERT-Base FHE pipeline reduces end-to-end inference latency by 36.3 % and 39.6 % on one and two GPUs, respectively. These results demonstrate a favorable accuracy–depth trade-off and practical end-to-end acceleration within an existing FHE inference system.

AI USE STATEMENT

In this work, we used generative AI tools for editing the paper to improve readability.

REPRODUCIBILITY STATEMENT

We provide the anonymous experimental code (https://github.com/cgf-anon/cgf-anon) for reproducing the proposed CGF-softmax. To facilitate reproducibility, we provide complete proofs of the theoretical results in Appendices A–D, implementation details of the homomorphic CGF-softmax algorithm in Appendix F, and details of the exponential function approximation in Appendix H. We further report the training procedure, hyperparameter search spaces, fixed configurations, and random seed used in our experiments in Appendix I, as well as the model adaptation, implementation settings, multiplicative-depth configuration, and latency breakdown for the end-to-end FHE evaluation in Appendix K.

REFERENCES

M Al-Rubaie and J M Chang. Privacy-preserving machine learning: Threats and solutions. *IEEE Security and Privacy*, 17(2):49–58, 2019.

Andrew C Berry. The accuracy of the Gaussian approximation to the sum of independent variates. *Transactions of the American Mathematical Society*, 49(1):122–136, 1941.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 1877–1901, December 2020.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on NLP for Conversational AI*, pp. 38–45, July 2020.

George Casella and Roger Berger. *Statistical inference*. Chapman and Hall/CRC, Boca Raton, April 2024.

Leo De Castro, Daniel Escudero, Adya Agrawal, Antigoni Polychroniadou, and Manuela Veloso. EncryptedLLM: Privacy-preserving large language model inference via GPU-accelerated fully homomorphic encryption. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 267, pp. 12677–12688, July 2025.

Tianyu Chen, Hangbo Bao, Shaohan Huang, Li Dong, Binxing Jiao, Daxin Jiang, Haoyi Zhou, Jianxin Li, and Furu Wei. THE-X: Privacy-preserving transformer inference with homomorphic encryption. In *Findings of the Association for Computational Linguistics (Findings of ACL)*, pp. 3510–3520, May 2022.

Elliott Ward Cheney. *Introduction to approximation theory*. McGraw-Hill, New York, 1966.

Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, pp. 409–437, December 2017.

Jung Hee Cheon, Wootae Kim, and Jai Hyun Park. Efficient homomorphic evaluation on large intervals. *IEEE Transactions on Information Forensics and Security*, 17:2553–2568, 2022.

Wonhee Cho, Guillaume Hanrot, Taeseong Kim, Minje Park, and Damien Stehlé. Fast and accurate homomorphic softmax evaluation. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 4391–4404, October 2024.

DESILO Inc. DESILO FHE library, 2025. URL https://desilo.ai.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 4171–4186, June 2019.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations (ICLR)*, May 2021.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, et al. The Llama 3 herd of models. *arXiv preprint*, arXiv:2407.21783, July 2024.

C.G. Esseen. *On the Liapounoff limit of error in the theory of probability*. Arkiv för matematik, astronomi och fysik. Almqvist & Wiksell, 1942.

David Evans, Vladimir Kolesnikov, and Mike Rosulek. A pragmatic introduction to secure multiparty computation. *Foundations and Trends in Privacy and Security*, 2(2-3):70–246, December 2018.

Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pp. 169–178, May 2009.

Robert E. Goldschmidt. Applications of division by convergence. Master's thesis, Massachusetts Institute of Technology, June 1964.

Seungwan Hong, Jai Hyun Park, Wonhee Cho, Hyeongmin Choe, and Jung Hee Cheon. Secure tumor classification by shallow neural network using homomorphic encryption. *BMC Genomics*, 23(1):284, April 2022.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, April 2022.

Siddharth Jayashankar, Joshua Kim, Michael B. Sullivan, Wenting Zheng, and Dimitrios Skarlatos. Cerium: a multi-GPU framework for terabyte-scale encrypted inference. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, September 2026.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1311–1316, November 2019.

Seewoo Lee, Garam Lee, Jung Woo Kim, Junbum Shin, and Mun-Kyu Lee. HETAL: Efficient privacy-preserving transfer learning with homomorphic encryption. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 202, pp. 19010–19035, July 2023.

Dacheng Li, Hongyi Wang, Rulin Shao, Han Guo, Eric Xing, and Hao Zhang. MPCFormer: Fast, performant and private transformer inference with MPC. In *Proceedings of the International Conference on Learning Representations (ICLR)*, February 2023.

Lawrence Lim, Vikas Kalagi, Divyakant Agrawal, and Amr El Abbadi. Tricycle: Private transformer inference with tricyclic encodings. *Cryptology ePrint Archive*, June 2025.

Qian Lou and Lei Jiang. HEMET: A homomorphic-encryption-friendly privacy-preserving mobile neural network architecture. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 7102–7110, July 2021.

Qian Lou, Yilin Shen, Hongxia Jin, and Lei Jiang. SafeNet: A secure, accurate and fast neural network inference. In *Proceedings of the International Conference on Learning Representations (ICLR)*, January 2020.

Jinglong Luo, Yehong Zhang, Zhuo Zhang, Jiaqi Zhang, Xin Mu, Hui Wang, Yue Yu, and Zenglin Xu. SecFormer: Fast and accurate privacy-preserving inference for transformer models via SMPC. In *Findings of the Association for Computational Linguistics (Findings of ACL)*, pp. 13333–13348, August 2024.

Jungho Moon, Dongwoo Yoo, Xiaoqian Jiang, and Miran Kim. THOR: Secure transformer inference with homomorphic encryption. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 3765–3779, October 2025.

Dongjin Park, Eunsang Lee, and Joon-Woo Lee. Powerformer: Efficient and high-accuracy privacy-preserving language model with homomorphic encryption. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 11090–11111, July 2025.

Mike Paterson and Larry J. Stockmeyer. On the number of nonscalar multiplications necessary to evaluate polynomials. *SIAM Journal on Computing*, 2(1):60–66, March 1973.

Donghwan Rho, Taeseong Kim, Minje Park, Jung Woo Kim, Hyunsik Chae, Ernest K. Ryu, and Jung Hee Cheon. Encryption-friendly LLM architecture. In *Proceedings of the International Conference on Learning Representations (ICLR)*, January 2025.

M Sadegh Riazi, Bita Darvish Rouani, and Farinaz Koushanfar. Deep learning on private data. *IEEE Security and Privacy*, 17(6):54–63, November 2019.

R L Rivest, L Adleman, and M L Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation*, 4(11):169–180, 1978.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 5998–6008, December 2017.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the EMNLP Workshop BlackboxNLP*, pp. 353–355, November 2018.

Ross Wightman. timm: PyTorch image models, 2019. URL https://github.com/rwightman/pytorch-image-models.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Jamie Brew, and Hugues Bennett. Transformers: State-of-the-art natural language processing. In *Proceedings of the EMNLP: System Demonstrations*, pp. 38–45, November 2020.

Linhan Yang, Jingwei Chen, Wangchen Dai, Shuai Wang, Wenyuan Wu, and Yong Feng. ARION: Attention-optimized transformer inference on encrypted data. *Cryptology ePrint Archive*, December 2025.

Itamar Zimerman, Allon Adir, Ehud Aharoni, Matan Avitan, Moran Baruch, Nir Drucker, Jenny Lerner, Ramy Masalha, Reut Meiri, and Omri Soceanu. Power-softmax: Towards secure LLM inference over encrypted data. *arXiv preprint*, arXiv:2410.09457, October 2024a.

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 International Conference on Machine Learning (ICML)*, volume 235, pp. 62803–62814, July 2024b.

A SHIFT INVARIANCE OF CGF-SOFTMAX

For any constant c,

$$\text{softmax}_{\text{CGF}}(\mathbf{x}-c)_i = \frac{\exp(x_i-c)}{n\mathbb{E}[\exp(X-c)]} = \frac{\exp(x_i)}{n\mathbb{E}[\exp(X)]} = \text{softmax}_{\text{CGF}}(\mathbf{x})_i,$$

which holds for all $i \in \{1,\dots,n\}$. Thus, CGF-softmax is shift-invariant.

B RELATIVE ERROR

The relative error η is given by

$$\eta = \frac{\|\text{softmax}(\mathbf{x}) - \text{softmax}_{\text{CGF}}(\mathbf{x})\|_\infty}{\|\text{softmax}(\mathbf{x})\|_\infty} \tag{13}$$

$$= \frac{\left|\frac{1}{\sum_{i=1}^n \exp(x_i)} - \frac{1}{nM_X(1)}\right| \cdot \|\exp(\mathbf{x})\|_\infty}{\left|\frac{1}{\sum_{i=1}^n \exp(x_i)}\right| \cdot \|\exp(\mathbf{x})\|_\infty} \tag{14}$$

$$= \left|1 - \frac{\frac{1}{n}\sum_{i=1}^n \exp(x_i)}{\mathbb{E}[\exp(X)]}\right|. \tag{15}$$

Note that the relative error depends only on the ratio between the sample mean and the ensemble mean.

C ERROR ANALYSIS USING CHEBYSHEV'S INEQUALITY

Let $\bar{Y}_n = n^{-1}\sum_{i=1}^n Y_i$, where $Y_i = \exp(X_i)$, and let $\mu_Y = \mathbb{E}[\exp(X)] \in (0,\infty)$. Assume $\mathbb{E}[Y_i^2] < \infty$ for all i. By Eq. (10), the relative error can be written as

$$\eta = \left|1 - \frac{\bar{Y}_n}{\mu_Y}\right| = \frac{|\bar{Y}_n - \mu_Y|}{\mu_Y}. \tag{16}$$

Since $\mu_Y > 0$, for any $\delta > 0$,

$$\eta \geq \delta \quad \Longleftrightarrow \quad |\bar{Y}_n - \mu_Y| \geq \delta\mu_Y. \tag{17}$$

By Chebyshev's inequality in its second-moment form,

$$P_\eta \triangleq P(\eta \geq \delta) = P\big(|\bar{Y}_n - \mu_Y| \geq \delta\mu_Y\big) \tag{18}$$

$$\leq \frac{\mathbb{E}[(\bar{Y}_n - \mu_Y)^2]}{\delta^2\mu_Y^2}. \tag{19}$$

Writing $\bar{Y}_n - \mu_Y = (\bar{Y}_n - \mathbb{E}[\bar{Y}_n]) + (\mathbb{E}[\bar{Y}_n] - \mu_Y)$ and using $\mathbb{E}[\bar{Y}_n - \mathbb{E}[\bar{Y}_n]] = 0$ gives

$$\mathbb{E}[(\bar{Y}_n - \mu_Y)^2] = \text{Var}(\bar{Y}_n) + \big(\mathbb{E}[\bar{Y}_n] - \mu_Y\big)^2. \tag{20}$$

Combining these relations with $P_\eta \leq 1$, we obtain

$$P_\eta \leq \min\left\{1, \frac{\text{Var}(\bar{Y}_n) + \big(\mathbb{E}[\bar{Y}_n] - \mu_Y\big)^2}{\delta^2\mu_Y^2}\right\}, \tag{21}$$

which proves Eq. (11) without assuming independence or identical distributions. □

Now suppose that $X_1,\dots,X_n$ are i.i.d. with the same distribution as X, and let $\sigma_Y^2 = \text{Var}(\exp(X))$. Then

$$\mathbb{E}[\bar{Y}_n] = \mu_Y, \qquad \text{Var}(\bar{Y}_n) = \frac{1}{n^2}\sum_{i=1}^n \text{Var}(Y_i) = \frac{\sigma_Y^2}{n}, \tag{22}$$

where the variance identity follows from independence. Substituting these expressions into Eq. (21) yields

$$P_\eta \leq \min\left\{1, \frac{\sigma_Y^2}{n\delta^2\mu_Y^2}\right\}. \tag{23}$$

□

D Error Analysis using Berry–Esseen Theorem

Let $\bar{Y}_n = \frac{1}{n}\sum_{i=1}^{n} Y_i$, where $Y_i = \exp(X_i)$. By Eq. (10), the relative error can be written as

$$\eta = \left|1 - \frac{\bar{Y}_n}{\mu_Y}\right| = \frac{|\bar{Y}_n - \mu_Y|}{\mu_Y}. \tag{24}$$

Define the standardized sample mean

$$Z_n = \frac{\sqrt{n}(\bar{Y}_n - \mu_Y)}{\sigma_Y}. \tag{25}$$

Then, for any $\delta > 0$,

$$\eta \geq \delta \quad \Longleftrightarrow \quad |Z_n| \geq \frac{\delta \mu_Y \sqrt{n}}{\sigma_Y}. \tag{26}$$

Let

$$a_\delta = \frac{\delta \mu_Y \sqrt{n}}{\sigma_Y}. \tag{27}$$

Then

$$P_\eta \triangleq P(\eta \geq \delta) = P(|Z_n| \geq a_\delta). \tag{28}$$

By the Berry–Esseen theorem (Berry, 1941; Esseen, 1942),

$$\sup_{z \in \mathbb{R}} |P(Z_n \leq z) - \Phi(z)| \leq \frac{C_{\text{BE}} \cdot \rho_Y}{\sigma_Y^3 \sqrt{n}}. \tag{29}$$

Therefore,

$$P(|Z_n| \geq a_\delta) = P(Z_n \leq -a_\delta) + P(Z_n \geq a_\delta) \tag{30}$$

$$\leq \Phi(-a_\delta) + [1 - \Phi(a_\delta)] + 2 \cdot \frac{C_{\text{BE}} \cdot \rho_Y}{\sigma_Y^3 \sqrt{n}}. \tag{31}$$

Since $\Phi(-a_\delta) = 1 - \Phi(a_\delta)$, we obtain

$$P_\eta \leq 2\,[1 - \Phi(a_\delta)] + \frac{2C_{\text{BE}} \cdot \rho_Y}{\sigma_Y^3 \sqrt{n}}. \tag{32}$$

Absorbing the factor 2 into the universal constant C $(= 2C_{\text{BE}})$ and substituting the definition of a_δ gives

$$P_\eta \leq 2\left[1 - \Phi\left(\frac{\delta \mu_Y \sqrt{n}}{\sigma_Y}\right)\right] + \frac{C\rho_Y}{\sigma_Y^3 \sqrt{n}}, \tag{33}$$

which proves the claim. □

E Empirical Behavior of Higher-order Terms in the CGF

This section examines the effect of fine-tuning on the higher-order terms in the CGF. Recall that

$$K_X(1) = \kappa_1 + \frac{\kappa_2}{2!} + \sum_{j=3}^{\infty} \frac{\kappa_j}{j!}. \tag{34}$$

In CGF-softmax, we use the second-order cumulant approximation, $K_X(1) \approx \kappa_1 + \kappa_2/2$, and omit the higher-order terms $\kappa_j/j!$ for $j \geq 3$.

To examine the effect of fine-tuning on the omitted cumulant terms, we focus on $\kappa_3/3!$ and $\kappa_4/4!$, the first two terms omitted by the second-order cumulant approximation. Figure 3 reports the distribution of $\kappa_3/3! + \kappa_4/4!$ computed from DeiT-Base softmax inputs on ImageNet-1k, before and after fine-tuning. After fine-tuning, the distribution becomes more concentrated around zero, with its standard deviation reduced from 6.81 to 3.85. This indicates that fine-tuning suppresses the dominant higher-order terms.

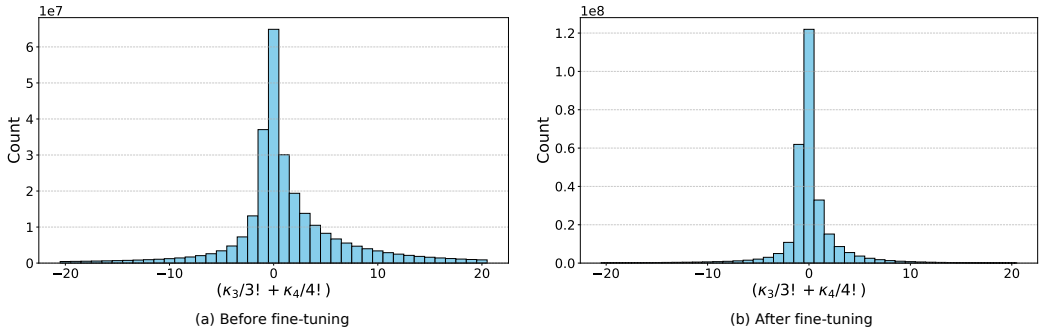

Figure 3: Distribution of $\kappa_3/3! + \kappa_4/4!$ for softmax inputs of DeiT-Base on ImageNet-1k, before and after fine-tuning.

Together with the factorial attenuation inherent in the CGF, this empirical reduction supports the second-order cumulant approximation used in CGF-softmax and explains why the omitted higher-order terms have negligible impact on model performance in practice.

F Implementation Details

F.1 Packing Strategy

In Transformer architectures, the softmax function is predominantly applied within the self-attention mechanism, necessitating row-wise execution across the input matrix. Accordingly, we target the row-wise application of CGF-softmax on an input matrix $A \in \mathbb{R}^{N_1 \times N_2}$. We assume N_1 and N_2 are powers of two; for arbitrary dimensions, we apply zero-padding to extend them to the nearest power of two. The input data is encrypted into $t = \lceil (N_1 \cdot N_2)/s \rceil$ ciphertexts. To facilitate efficient column-wise aggregation, we employ a structure similar to the column-major packing variant used by Cho et al. (2024). Specifically, elements within the same row are spaced by a fixed gap g, defined as $g = (t \cdot s)/N_2$.

Let $ct_i[j]$ denote the value stored in the j-th slot of the i-th ciphertext. The element $A[n_1, n_2]$ located at row n_1 ($0 \leq n_1 < N_1$) and column n_2 ($0 \leq n_2 < N_2$) is mapped to the ciphertext as follows:

$$A[n_1, n_2] = ct_i[j], \quad \text{where } \begin{cases} i = \lfloor (n_2 \cdot g)/s \rfloor, \\ j = n_1 + (n_2 \cdot g \bmod s). \end{cases} \tag{35}$$

Any slots in the ciphertexts that are not populated by matrix elements according to this mapping are initialized to zero.

F.2 ALGORITHM

Based on the packing strategy described in Appendix F.1, we present the outline of homomorphic CGF-softmax in Algorithm 1. The input consists of a set of ciphertexts $\mathcal{V} = \{ct_0, \ldots, ct_{t-1}\}$ representing the encrypted matrix. The algorithm is designed to evaluate CGF-softmax in a row-wise manner. Notably, the entire pre-processing stage (Steps 1 and 2) consumes 1 multiplicative level. This is because the squaring operation necessitates a single ciphertext–ciphertext multiplication. Furthermore, to optimize the circuit depth, consecutive scalar multiplications are pre-computed into a single scalar and applied to the ciphertext once; this is a common optimization technique in HE to prevent unnecessary level consumption. For the exponential function, we employ a polynomial approximation denoted as AExp, the details of which are outlined in Appendix H.

Algorithm 1 Homomorphic CGF-softmax Outline

Input: A set of ciphertexts $\mathcal{V} = \{ct_0, \ldots, ct_{t-1}\}$ encrypting the matrix.
Output: A set of ciphertexts $\mathcal{W} = \{ct'_0, \ldots, ct'_{t-1}\}$ encrypting the result of applying CGF-softmax row-wise to the matrix.
{**// Step 1: Compute Mean** (μ)}
$ct_{\text{sum}} \leftarrow$ Sum all ciphertexts in $\mathcal{V}$ using addition.
$\mu \leftarrow$ Perform row-wise summation on ct_{sum} by repeating addition and rotation, and multiply the result by the plaintext constant $1/n$.
{**// Step 2: Compute Variance Term** ($\frac{1}{2}\sigma^2$)}
$\mathcal{V}_{\text{centered}} \leftarrow$ Subtract μ from each ciphertext in $\mathcal{V}$.
$\mathcal{V}_{\text{sq}} \leftarrow$ Square each ciphertext in $\mathcal{V}_{\text{centered}}$.
$ct_{\text{sum}} \leftarrow$ Sum all ciphertexts in $\mathcal{V}_{\text{sq}}$.
var_term $\leftarrow$ Perform row-wise summation on ct_{sum} by repeating addition and rotation, and multiply the result by the plaintext constant $1/(2n)$.
{**// Step 3: Final Approximation**}
$\mathcal{V}_{\text{norm}} \leftarrow$ Subtract var_term and $\ln n$ from each ciphertext in $\mathcal{V}_{\text{centered}}$.
$\mathcal{W} \leftarrow$ Apply AExp to each ciphertext in $\mathcal{V}_{\text{norm}}$.
return $\mathcal{W}$

G NOISE ANALYSIS UNDER HOMOMORPHIC EVALUATION

In CKKS-based homomorphic evaluation, evaluation noise accumulates through arithmetic operations such as ciphertext multiplication, rescaling, and bootstrapping. The amount of accumulated noise therefore depends on both the multiplicative depth and the number of homomorphic operations in the circuit. From this perspective, homomorphic division is particularly costly, since it is implemented through an iterative approximation procedure involving additional multiplications and rescalings. Thus, eliminating division directly reduces the main sources of noise accumulation in the softmax circuit.

To empirically show this, we compare the total evalaution error induced when the standard softmax and CGF-softmax are evaluated homomorphically. For an input vector $\mathbf{x}$, we denote the plaintext evaluations by $\text{softmax}(\mathbf{x})$ and $\text{softmax}_{\text{CGF}}(\mathbf{x})$, and their homomorphic evaluations by $\text{softmax}_{\text{FHE}}(\mathbf{x})$ and $\text{softmax}_{\text{CGF,FHE}}(\mathbf{x})$, respectively.

We measure the total evaluation error by the ℓ_∞ error between the plaintext and homomorphic evaluation results:

$$\left\|\text{softmax}(\mathbf{x}) - \text{softmax}_{\text{FHE}}(\mathbf{x})\right\|_\infty, \quad \left\|\text{softmax}_{\text{CGF}}(\mathbf{x}) - \text{softmax}_{\text{CGF,FHE}}(\mathbf{x})\right\|_\infty. \tag{36}$$

For a fair comparison, both methods use the same degree-15 Chebyshev polynomial approximation for the exponential function, following the standard configuration in Appendix H. For CGF-softmax, the computation requires one depth for estimating the mean and variance before exponentiation, four depths for the degree-15 exponential approximation, and two additional depths for the domain scaling (see Eq. (9)). Therefore, the homomorphic evaluation of CGF-softmax is completed at multiplicative depth 7 without requiring homomorphic division.

The standard softmax uses the same exponential approximation, which accounts for multiplicative depth 6. Unlike CGF-softmax, however, the standard softmax additionally requires homomorphic

division for normalization. While various methods exist for implementing division under HE, we adopt Goldschmidt's iterative algorithm (Goldschmidt, 1964), since it is depth-efficient and requires relatively fewer homomorphic operations, thus introducing less encryption noise. Each Goldschmidt iteration consumes one additional multiplicative depth and reduces the division error.

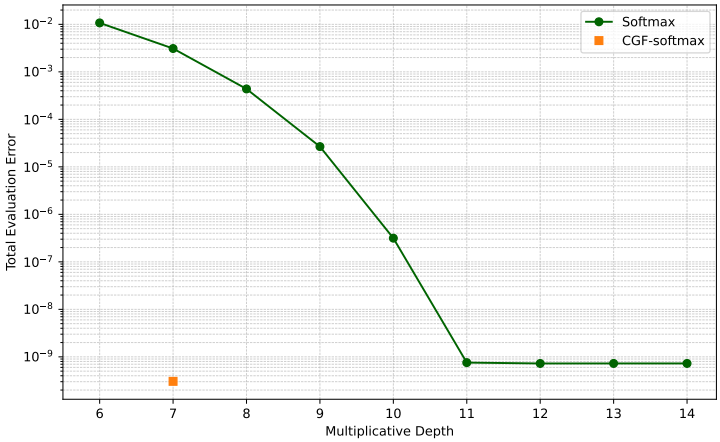

Figure 4: Total evaluation error of the standard softmax and CGF-softmax under homomorphic evaluation, measured by the ℓ_∞ error between plaintext and homomorphic outputs. The x-axis denotes the overall multiplicative depth. For CGF-softmax, the computation is completed at depth 7 without homomorphic division, achieving a total evalaution error of 3.05×10^{-10}. The standard softmax uses the same degree-15 exponential approximation and starts from depth 6, but additionally requires Goldschmidt's iterative division algorithm for normalization, where each iteration consumes one more multiplicative depth. Although increasing the number of division iterations reduces the total evaluation error of the standard softmax, it remains noisier than CGF-softmax even at larger depths.

Figure 4 reports the total evaluation error as a function of the multiplicative depth. For the standard softmax, increasing the number of Goldschmidt iterations reduces the total evaluation error, but this improvement comes at the cost of additional multiplicative depth. Moreover, even with larger depth, the total evaluation error of the standard softmax remains higher than that of CGF-softmax. In contrast, CGF-softmax avoids homomorphic division and achieves substantially lower total evaluation error of 3.05×10^{-10} at depth 7. This error level is also small in absolute terms: it is about two orders of magnitude below the FP32 machine epsilon ($2^{-23} \approx 1.19 \times 10^{-7}$), and far below the precision levels of FP16 ($2^{-10} \approx 9.77 \times 10^{-4}$) and BF16 ($2^{-7} = 7.81 \times 10^{-3}$) arithmetic commonly used in neural network inference. These results demonstrate that removing homomorphic division substantially reduces the noise introduced by softmax evaluation under HE, providing a clear advantage for noise-budget management in deeper HE neural networks.

H Approximation of the Exponential Function

In this section, we detail the implementation of the polynomial approximation function $\mathsf{AExp}(x) \approx \exp(x)$, which constitutes the sole polynomial approximation step within the CGF-softmax framework. The configuration of CGF-softmax–specifically, whether it operates as the *standard configuration* or the efficient *low-degree variant*–is determined by the timing of approximation relative to the training phase.

H.1 Standard Configuration

The standard CGF-softmax configuration applies exponential approximations in *post-training* to prioritize maximum accuracy recovery. In this setting, the model is trained using the exact CGF-softmax formulation, and the polynomial approximation is introduced only during the inference phase. We utilize two distinct polynomial approximation methods for $\mathsf{AExp}(x)$.

Chebyshev Polynomial Approximation. To accommodate the wide dynamic range of input values while maintaining high precision, we leverage the exponential identity $\exp(x) = (\exp(x/2^k))^{2^k}$. This formulation allows us to map the input to a reduced interval (e.g., $[-8, 0]$ following Cho et al. (2024)) by scaling the domain by a factor of $1/2^k$. Within this reduced interval, we employ a degree $d = 15$ Chebyshev polynomial, with coefficients determined by a least-squares approximation over the target interval. To evaluate this efficiently, we adopt the Paterson-Stockmeyer algorithm (Paterson & Stockmeyer, 1973), which reduces the multiplicative complexity of polynomial evaluation. While this method approximates the exponential function with high precision over a wide interval, it requires a relatively high multiplicative depth, calculated as $\lceil \log_2(d+1) \rceil + k + 1$, due to the combination of polynomial evaluation and the subsequent k squaring steps required to recover the original scale.

Limit Approximation. We implement this as $\mathsf{AExp}(x) = (1 + x/2^k)^{2^k}$. This approach is computationally efficient, involving only a single scalar multiplication followed by k repeated squarings. Consequently, it consumes less multiplicative depth, requiring only $k + 1$. However, compared to the Chebyshev method, the Limit approximation is less robust, as the approximation error tends to increase rapidly when the input values fall outside the target domain.

The specific values of the parameter k utilized in our experiments are detailed in Table 6. In the reported results of Figure 1 and Figure 2, we select the approximation method that yields better performance.

Table 6: Specific values of k employed for the CGF-softmax standard configuration across different models and datasets. All reported values achieve inference accuracy within $1\,\%$ of the pre-trained exact softmax baseline.

Model	**Llama-3.2-1B**			**ViT-Base**	**DeiT-Base**	**ViT-Tiny**	**DeiT-Tiny**
Dataset	Clinc150	Banking77	SST-2	ImageNet-1k			
Chebyshev approx.	3	3	4	6	3	1	1
Limit approx.	6	6	7	8	6	8	7

H.2 Low-degree Variant

The low-degree variant is designed for scenarios requiring maximum efficiency with minimal multiplicative depth. In this approach, the approximation is applied *in-training*; the exact exponential function is replaced by a simplified polynomial approximation *before* the training phase begins. We utilize a Taylor expansion of degree d centered at x_0, formulated as:

$$\mathsf{AExp}(x) = \sum_{i=0}^{d} \frac{e^{x_0}}{i!}(x - x_0)^i \tag{37}$$

A key advantage of this method is that it eliminates the need for domain scaling, thereby avoiding the additional overhead associated with squaring operations. Total multiplicative depth required for this operation is $\lceil \log_2(d + 1) \rceil$.

I Training Details

BPMax employs a knowledge distillation (KD) framework, where the original pre-trained model serves as the *teacher* and the model with the replaced function acts as the *student*. We adopt the same KD framework to CGF-softmax for fair comparison.

Search Space and Configurations. To ensure a rigorous evaluation, we conducted an extensive hyperparameter sweep for both Llama-3.2-1B and ViT/DeiT. Table 7 details the specific search grids. For BPMax, which replace softmax with simplified polynomial alternatives $(x+c)^p$, we significantly expanded the search space beyond the original study (Park et al., 2025)–which limited the search to shift variable $c \in \{1, 3, 5, 7\}$ and degrees $p \in \{1, 3, 5, 7\}$–to guarantee optimal baselines.

We set the maximum training budget for BPMax to 20 epochs while applying early stopping to all experiments (including CGF-softmax). Under these conditions, CGF-softmax demonstrated significantly faster training speeds than the baseline: Llama-3.2-1B required only 5 epochs. For ViT/DeiT models, we employed *Attention-only Tuning*—a strategy where we freeze all model parameters and exclusively update the weights associated with the attention mechanism. This approach provided the dual benefit of minimizing the training process and facilitating effective accuracy recovery.

Fixed Configurations. Regarding input specifications, we set the sequence length to $n = 197$ for ViT/DeiT models and the maximum sequence length to 128 for Llama-3.2-1B. For the KD objective, we utilized a combination of Cross-Entropy loss and KL-Divergence loss, setting the temperature to $\tau = 2.0$ and the distillation weight to $\alpha = 0.5$. We fixed the random seed to 42 across all experimental runs.

Table 7: Configuration search space and fixed settings for large language models and Vision Transformers. In the Teacher Model row, 'Self' refers to the original backbone prior to replacing the softmax, while 'ViT-Large (85.84%)' denotes the pre-trained model sourced from the `timm` library.

Configuration	Llama-3.2-1B	Vision Transformers (ViT/DeiT)
CGF-softmax (low degree variant)		
Taylor Degree (d)	{1}	{3, 5}
Expansion Point (x_0)	{−3}	{−10}
BPMax		
Degree (p)	{1, 3, 5, 7}	{1, 3, 5, 7, 9}
Scaling (c)	{1, 3, 5, 7, 20}	{1, 3, 5, 7, 9, 20, 40, 60}
Optimization		
Epochs (CGF-softmax)	{5}	{15}
Epochs (BPMax)	{20}	{20}
Batch Size	{32, 64}	{64, 128}
Learning Rate	{9e-6, 1e-5, 3e-5, 5e-5}	{1e-5, 5e-5, 1e-4}
Regularization & Strategy		
Drop Path Rate	N/A	{0.0, 0.1, 0.2}
Attention-only Tuning	{Off}	{On, Off}
Knowledge Distillation		
Teacher Model	{Self}	{Self, ViT-Large (85.84%)}

J Alternative Training Pipeline: Replace Softmax Before Supervised Fine-Tuning

When evaluating large language models (LLMs) like Llama-3.2-1B on specific downstream tasks, applying Supervised Fine-Tuning (SFT) is standard practice. In our main experiments (Figures 1 and 2), the "exact softmax" baselines were established by performing SFT directly on the pre-trained models downloaded from Hugging Face (HF). Subsequently, to ensure a fair evaluation of the softmax replacement methods, we replaced the softmax function in these already fine-tuned models and conducted an additional Knowledge Distillation (KD) phase (as detailed in Appendix I) to create the BPMax and CGF-softmax models. While CGF-softmax inherently requires a significantly lower training cost than BPMax during this KD phase, this section introduces an alternative pipeline that removes the *additional* adaptation training cost.

In this alternative pipeline, we replace the exact softmax with CGF-softmax immediately on the base pre-trained model, *prior* to any task-specific tuning. We then proceed directly with a standard Low-Rank Adaptation (LoRA) (Hu et al., 2022) SFT phase. As detailed in Table 8, this unified approach achieves an accuracy within $1\,\%$ of the exact softmax baseline using the exact same number of

training epochs. In settings that already include task-specific SFT, this pipeline avoids a separate KD adaptation stage. Consequently, adapting the model to CGF-softmax incurs virtually zero extra training overhead.

Table 8: Accuracy and training configuration comparison for Llama-3.2-1B downstream adaptation.

Method	Training Settings				Evaluation Accuracy		
	Base Model	Softmax	Training Method	Epochs	Clinc150	Banking77	SST-2
Exact softmax baseline	HF Pre-trained	Exact	SFT	3	86.67%	92.99%	93.11%
CGF-softmax pipeline	Exact softmax baseline	CGF-softmax	KD	5	87.02%	92.86%	92.43%
BPMax pipeline	Exact softmax baseline	BPMax	KD	20	81.40%	91.13%	87.15%
Alternative pipeline	**HF Pre-trained**	**CGF-softmax**	**SFT (LoRA)**	**3**	**87.09%**	**92.04%**	**93.00%**

K End-to-end Latency Evaluation Details

Model adaptation. Cerium evaluates BERT-Base on the GLUE RTE task and reports an accuracy of 69.3% with its original softmax implementation. Using a fine-tuning setup similar to that in Appendix I, we fine-tune BERT-Base on RTE for eight epochs with a learning rate of 2×10^{-5} to obtain model weights compatible with CGF-softmax. The resulting model achieves an accuracy of 68.59%. We then port these weights to the Cerium implementation and replace only its softmax operation with CGF-softmax, while leaving all other model operations unchanged.

CGF-softmax implementation. For the exponential evaluation, we use the same approximation as the original Cerium implementation,

$$\exp(x) \approx \left(1 + \frac{x}{2^6}\right)^{2^6}. \tag{38}$$

To compute the row-wise mean and variance required by CGF-softmax, we directly reuse Cerium's row-wise summation routine originally used to form the denominator of softmax after exponentiation. This routine first aggregates all elements within each row using ciphertext rotations and additions. The aggregated values are then multiplied by a binary plaintext mask consisting of zeros and ones to retain the required slots. A second sequence of rotations and additions broadcasts each retained row-wise sum back to the positions of all elements in the corresponding row. We apply this same routine to compute the row-wise sums required for the mean and variance.

Multiplicative depth and level consumption. The CGF-softmax implementation consumes a total multiplicative depth of nine. In the original Cerium implementation, the softmax computation starts at Level 16 after bootstrapping and, after its nonlinear operations and internal bootstrapping, produces its output at Level 8. CGF-softmax instead produces its output at Level 7 without the internal bootstrapping operations required by the original softmax. Consequently, the subsequent multiplication with the value matrix and output-projection matrix is executed one level lower than in the original implementation. Except for this level difference, the remaining BERT-Base computation is identical between the two implementations.

Table 9: Single-GPU latency breakdown for encrypted BERT-Base inference on an NVIDIA RTX PRO 6000 Blackwell GPU. Latencies are measured over 100 iterations and reported in seconds as mean $\pm$ standard deviation.

Operation	Component	Original	CGF-softmax
Softmax	Non-bootstrapping	3.08 ± 0.00	0.31 ± 0.00
	Bootstrapping	14.76 ± 0.01	–
Others	Non-bootstrapping	19.38 ± 0.01	18.93 ± 0.02
	Bootstrapping	12.25 ± 0.01	12.26 ± 0.02
Total		$\mathbf{49.48 \pm 0.03}$	$\mathbf{31.51 \pm 0.04}$

Latency breakdown. Table 9 reports the latency breakdown for the single-GPU configuration. The original softmax requires $3.08\,\mathrm{s}$ for non-bootstrapping operations and an additional $14.76\,\mathrm{s}$ for bootstrapping, resulting in a total softmax latency of $17.84\,\mathrm{s}$. CGF-softmax reduces this cost to $0.31\,\mathrm{s}$ and requires no bootstrapping within the softmax operation. Meanwhile, the latency of all remaining operations changes only marginally from $31.63\,\mathrm{s}$ to $31.19\,\mathrm{s}$. Therefore, the end-to-end latency reduction from $49.48\,\mathrm{s}$ to $31.51\,\mathrm{s}$ is primarily driven by the elimination of bootstrapping within softmax. With two GPUs, CGF-softmax similarly reduces the end-to-end latency from $31.96\,\mathrm{s}$ to $19.30\,\mathrm{s}$, demonstrating consistent latency improvements in the multi-GPU setting.