PowerSoftmax: Towards Secure LLM Inference over Encrypted Data

Itamar Zimerman **Allon Adir** **Ehud Aharoni** **Matan Avitan**

Moran Baruch **Nir Drucker** **Jenny Lerner** **Ramy Masalha**

Reut Meiri **Omri Soceanu**

IBM Research

Abstract

Modern cryptographic methods for implementing privacy-preserving LLMs such as homomorphic encryption (HE) require the LLMs to have a polynomial form. Forming such a representation is challenging because transformers include non-polynomial components, such as Softmax and layer normalization. Previous approaches have either directly approximated pre-trained models with large-degree polynomials, which are less efficient over HE, or replaced non-polynomial components with easier-to-approximate primitives before training, e.g., Softmax with pointwise attention. The latter approach might introduce scalability challenges. We present a new HE-friendly variant of self-attention that offers a stable form for training and is easy to approximate with polynomials for secure inference. Our work introduces the first polynomial LLMs over a billion parameters, exceeding the size of previous models by more than tenfold. The resulting models demonstrate reasoning and in-context learning (ICL) capabilities comparable to standard transformers of the same size, representing a breakthrough in the field. Finally, we provide a detailed latency breakdown for each computation over encrypted data, paving the way for further optimization, and explore the differences in inductive bias between models relying on our HE-friendly variant and standard transformers.

Proceedings of the 29[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

1 INTRODUCTION

Privacy-preserving machine learning (PPML) solutions and in particular privacy-preserving LLMs (Yan et al., 2024; Yao et al., 2024) aim to provide confidentiality guarantees for user data, the model owner, or both. One prominent cryptographic primitive for achieving this is homomorphic encryption (HE), as it allows computations to be performed on encrypted data without revealing any information to the (potentially untrusted) computing environment. Furthermore, HE enables non-interactive computations, which increases the usability of these solutions.

However, modern HE schemes like CKKS (Cheon et al., 2017) face a significant challenge of only supporting polynomial computations on encrypted data. This limitation complicates the deployment of DL models , particularly for LLMs, which depend on non-polynomial functions like Softmax in self-attention. To overcome this, existing approaches have adapted these non-polynomial operations into polynomial forms using techniques such as unique polynomial approximation (Lee et al., 2021) or fine-tuning procedures (Baruch et al., 2022). While these methods have enabled the execution of FFNs, CNNs (Lee et al., 2022), and small transformers (Zimerman et al., 2024)over HE, they often struggle with stability and sensitivity issues (Goyal et al., 2020), preventing an effective scale-up.

We take a different approach. Rather than modifying existing transformers to fit within the constraints of HE, we revisit the core design principles of the transformer architecture (Vaswani et al., 2017) through the lens of the CKKS constraints. Concretely, we ask:

Are there HE-friendly operators that can replicate the key design principles of self-attention?

We find a positive answer by introducing a power-based variant of self-attention that is more amenable to polynomial representation. Models with this variant

maintain comparable performance to Softmax-based transformers across several benchmarks and preserve the core design characteristics of self-attention. We also present variants that include length-agnostic approximations or improved numerical stability. The entire mechanism offers a more HE-friendly and effective transformer solution than previous approaches, enabling our method to scale efficiently to LLMs with 32 layers and 1.4 billion parameters.

Our main contributions: (i) We propose a HE-friendly self-attention variant tailored specifically for HE environments. This variant minimizes the usage of non-polynomial operations while maintaining the core principles of attention mechanisms. Additionally, we extend this approach by introducing a numerically stable training method and a length-agnostic computation strategy for inference. As a result, our model enables secure inference at scale and is more efficient than existing methods. (ii) We leverage this technique to develop the first polynomial LLM that exhibits reasoning and ICL capabilities, as well as the largest polynomial model trained to date, encompassing 32 transformer layers and approximately a billion parameters, and a polynomial variant of RoBERTa (iii) We validate that our model operates precisely over HE, and also provide ablation studies and profiling of latency breakdowns over encrypted data, paving the way for further improvements.

2 BACKGROUND

Homomorphic Encryption HE is a form of encryption that enables processing of encrypted data without decrypting it (Gentry, 2009), so that the results achieved from processing data inputs while encrypted are similar to the results of applying the same computation after decryption. Some HE schemes (Brakerski et al., 2014; Fan and Vercauteren, 2012) are exact, meaning that the value of the decrypted ciphertext is exactly the result of the arithmetic operation, while some like CKKS (Cheon et al., 2017) are approximate and introduce a tiny amount of noise (η) to the decrypted values. Formally, an HE scheme encryption operation $\mathrm{E} : \mathbb{R}_1 \to \mathbb{R}_2$ takes a plaintext from a ring $\mathbb{R}_1(+, *)$ and transforms it into a ciphertext in a ring $\mathbb{R}_2(\oplus, \odot)$ (and the opposite holds for decryption $\mathrm{D} : \mathbb{R}_2 \to \mathbb{R}_1$). This all occurs while also maintaining the following properties for an input $x, y \in \mathbb{R}_1$: (i) $\mathrm{D}(\mathrm{E}(x)) = x + \eta$, (ii) $\mathrm{D}(\mathrm{E}(x) \oplus \mathrm{E}(y)) = x + y + \eta$, and (iii) $\mathrm{D}(\mathrm{E}(x) \odot \mathrm{E}(y)) = x * y + \eta$.

Polynomial DL Models Deep learning models rely heavily on non-polynomial activation functions like ReLU, sigmoid, and tanh to introduce non-linearity, which enhances model expressiveness. However, over most HE schemes, operations must have a polynomial form. Prior work has reported that polynomial DNNs tend to face instability as the network grows (Zhou et al. (2019); Goyal et al. (2020); Chrysos et al. (2020); Gottemukkula (2020)). Thus, maintaining an accurate and stable network when using polynomial approximations is challenging.

There are two primary approaches for polynomial approximation: post-training approximation (PTA) and approximation-aware training (AAT). In PTA, the approximation is applied to a pre-trained network without modifying the model architecture and parameters ((Lee et al., 2021; Ao and Boddeti, 2024; Ju et al., 2023; Zhang et al., 2024b; Avitan et al., 2025)). This approach saves the costly training process by providing a precise approximation for each computation using high-degree polynomials.

In contrast, AAT aims to reduce the number of required approximation polynomials in the network or to minimize their degree (Gilad-Bachrach et al., 2016; Lee et al., 2023; Baruch et al., 2022, 2025; Ao and Boddeti, 2024; Drucker and Zimerman, 2023; Zimerman et al., 2024). Doing so can improve both latency and precision under HE, as higher-degree polynomials increase the *multiplicative depth*–the number of sequential multiplications required–leading to higher computational overhead, greater resource consumption, and an increase in the accumulated noise. Typically, this is achieved by modifying the network architecture. For instance, early studies substituted the ReLU activation function with quadratic activations (Gilad-Bachrach et al., 2016; Baruch et al., 2022).

To reduce polynomials' degrees in large models, such as ResNet152 on ImageNet and transformers, while still achieving accurate approximation, recent works ((Baruch et al., 2025) and (Zimerman et al., 2024)) have suggested using the training process to minimize the input range to the non-polynomial layers. This is done by adding a **range-loss term** to the original loss, encouraging the model to operate within a range where lower-degree approximations are accurate enough.

Polynomial Transformers To enjoy the non-interactive property of HE-based solutions, this paper only considers fully polynomial models. While other secure alternatives (Chen et al., 2022; Ding et al., 2023; Liang et al., 2024; Gupta et al., 2023; Zheng et al., 2023) exist, they require interaction with the user to process non-polynomial layers, incurring extra communication overhead and potential vulnerability to cryptographic attacks (Akavia and Vald, 2021). In contrast, HE enables non-interactive computation in untrusted environments without additional communication. To this end, the non-polynomial operations in transformers, namely Softmax, LayerNorm,

and GELU, need to be replaced or approximated.

The first work to present a fully polynomial transformer was by (Zimerman et al., 2024), who used the AAT approach and substituted Softmax with a scaled- ReLU that is easier to approximate. They also used the range-loss term during training to reduce the polynomial degree required for accurate approximation of ReLU and LayerNorm. They demonstrated a 100M-parameter polynomial transformer pretrained on WikiText-103 for secure classification using HE.

Alternatively, (Zhang et al., 2024b) used the PTA approach. They introduced a polynomial transformer by directly approximating the numerator, denominator, and division separately, without dedicated training modifications. However, as described in Sec. 5.3, this approach has disadvantages in terms of latency.

In this work, we scale up the AAT for transformers approach, by replacing Softmax with a polynomial-friendly alternative that closely replicates its behavior. This enhancement allows us to improve model performance and scalability, enabling the deployment of 1.4B-parameters LLMs under HE, while maintaining the model's performance. After training, we approximate the non-polynomial operations using methods detailed in App. C, converting the trained model into a polynomial form for secure inference.

3 PROBLEM SETTING

We consider secure inference for LLMs over HE, where a semi-honest server runs inference on behalf of a data-owner whose query remains encrypted throughout. Optionally, model weights may also be encrypted, depending on the trust assumptions between the server and the model owner; our approach supports both cases transparently (see App. H for details).

The core technical goal is a transformer that relies *exclusively* on polynomial operations, enabling direct HE deployment, while matching the performance of standard transformers at billion-parameter scale. This is challenging on two fronts: polynomial networks are prone to instability even at modest scales (Zhou et al., 2019; Goyal et al., 2020; Zhang et al., 2024a), and HE computational cost grows with polynomial degree, making high-degree solutions impractical. These pressures pull in opposite directions, as expressivity demands higher-degree polynomials while efficiency demands lower ones, and navigating this tension is the central challenge we address.

4 METHOD

The self-attention mechanism is defined by:

$$\text{Self-Attention}(Q,K,V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V\,, \qquad (1)$$

which is inherently non-polynomial because it includes division and exponential operations. Furthermore, for numerical stability it is common to compute the Softmax function using the *log-sum-exp* trick, which adds non-polynomial operations. For example, it involves calculating the maximum absolute values of each row of QK^T. The latter operation involves high-degree polynomials that in HE environments may introduce significant noise. Instead of directly approximating the maximum, division, and exponential functions individually (as done in the Nexus protocol (Zhang et al., 2024b)), our objective is to develop a more polynomial-friendly and HE-compatible Softmax variant for transformers. Such a mechanism not only reduces the overall computational complexity, particularly in terms of multiplication depth, but also supports scaling polynomial transformers to models with billions of parameters and deeper architectures.

4.1 HE-Friendly Attention

To design an HE-friendly variant of Softmax-based attention, we start by distilling its properties that correlate with its performance: (i) Normalization of the attention scores ensures they are bounded in $[0,1]$, with their sum equal to 1, similar to probabilities; (ii) exponential scaling of attention scores, such that it amplifies the differences between higher and lower scores; and (iii) monotonic increasing and order-preserving behavior, meaning that higher input values yield higher output values, while preserving the relative order of the input values. Building on these properties, we introduce the following attention variant:

$$\text{HE-Friendly Attn}(Q,K,V) = \text{PowerSoftmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$\text{PowerSoftmax}(x)_j = \frac{{x_j}^p}{\sum_i x_i^p}\,, \qquad (2)$$

where we replaced the $\text{Softmax}(x)_j = e^{x_j}/\sum_i e^{x_i}$ function with PowerSoftmax, for some positive even p. Equation 2 describes a variant that satisfies (i), but does not accurately retain properties (ii) and (iii), as the variant performs *polynomial scaling* instead of *exponential scaling* (both have superlinear trends), and because it is not strictly monotonic increasing. Nevertheless, for suitable values of p, the polynomial scaling can mimic the trends of exponential scaling relatively well, as shown in Fig. 1. Additionally, instead of maintaining the order and strictly increasing monotonically, our variant preserves *the order of the norms* and is increasing monotonically for positive values.

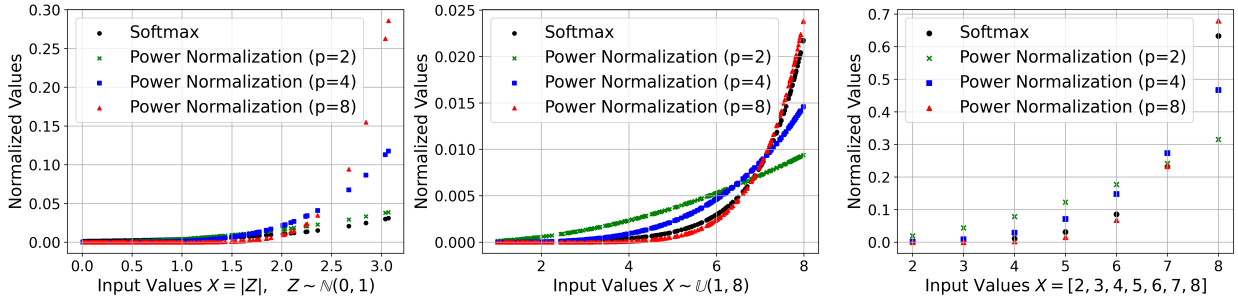

Figure 1: **Comparison of** Softmax **and** PowerSoftmax on normally distributed values on the left, uniformly distributed values in the middle, and evenly spaced values on the right. As can be seen, the scaling trends are relatively similar.

To highlight the similarities and differences between both attention mechanisms in Eqs. 1 and 2, we introduce a generalization of the Softmax function within transformers, using an elementwise activation function $\sigma : \mathbb{R} \to \mathbb{R}$ followed by proportional normalization $\mathbb{N} : \mathbb{R}^L \to \mathbb{R}^L$:

$$\text{Generalized Self-Attn}(Q, K, V) = \mathbb{N}\left(\sigma\left(\frac{QK^T}{\sqrt{d_k}}\right)\right)V \tag{3}$$

$$\mathbb{N}(\mathbf{x})_j = \frac{|\mathbf{x}_j|}{\|\mathbf{x}\|_1}. \tag{4}$$

In this formulation, Softmax is obtained by setting σ as $\sigma_e(x) = \exp(x)$, while our variant is defined by using $\sigma_p(x) = x^p$ for σ using a positive even p. Additionally, standard LLM practices often involve using attention masks. Thus, a masked HE-friendly variant is included in App. I.

4.2 $\frac{1}{\epsilon^2}$-Lipschitz Division for Softmax Approximation

A key challenge in approximating Softmax or Eq. 2 with polynomials is the behavior of the inverse term $1/x$, which grows rapidly near zero, i.e., $\lim_{x\to 0^+} \frac{1}{x} = \infty$. While Softmax deals with summation over strictly positive exponents, this property does not hold for PowerSoftmax, where the denominator can potentially reach zero. To address this, we propose the $\frac{1}{\epsilon^2}$-*Lipschitz division for* PowerSoftmax, modifying the denominator of $\mathbb{N}$ before training as:

$$\frac{1}{\epsilon^2}\text{-Lipschitz HE-Friendly Attn}(Q, K, V) = \tag{5}$$

$$\mathbb{N}_\epsilon\left(\sigma_p\left(\frac{QK^T}{\sqrt{d_k}}\right)\right)V, \quad \mathbb{N}_\epsilon(\mathbf{x})_j = \frac{|\mathbf{x}_j|}{\epsilon + \|\mathbf{x}\|_1}.$$

Here, ϵ (e.g., $1e-3$) ensures the denominator is bounded away from zero, preventing discontinuities and ensuring $\lim_{x\to 0^+} \frac{1}{x+\epsilon} = \frac{1}{\epsilon}$. This introduces a single non-polynomial division, which is $\frac{1}{\epsilon^2}$-Lipschitz continuity function, making the polynomial approximation more tractable. Importantly, unlike the common use of ϵ for numerical stability in division, our approach focuses on much larger values of ϵ to reduce the multiplication depth required for approximation, making the approximation problem significantly easier for secure inference over HE.

4.3 Stable Variant for Training

By examining the i-th row of the unnormalized attention scores $S_i = \left[\frac{1}{\sqrt{d_k}}QK^T\right]_i$, it is clear that Eq. 2 can lead to training instability when applying PowerSoftmax, as when $|S_{i,j}| > 1$, $|S_{i,j}|^p$ can become very large, causing overflow, and when $|S_{i,j}| < 1$, $|S_{i,j}|^p$ can become very small, leading to underflow. In transformers, a similar problem occurs with the traditional Softmax, which is mitigated using the *log-sum-exp trick* to scale the values of $|S_i|$ within a manageable range. Inspired by this, we propose a more stable version of our PowerSoftmax variant:

$$\text{Stable PowerSoftmax}(\mathbf{x})_j := \text{PowerSoftmax}\left(\frac{\mathbf{x}}{c}\right)_j, \tag{6}$$

where $c = \|\mathbf{x}\|_\infty + \delta$ and δ is a small positive number introduced to avoid division by 0. This method leverages the fact that PowerSoftmax is invariant to division of its input by a constant $c > 0$ (similar to Softmax which is invariant under the subtraction). By selecting c such that $\forall j |S_{i,j}| < 1$, we (i) ensure that the input values stay within a range where floating-point precision is more reliable ($0 < |S_{i,j}| < 1$), and (ii) stretch (or shrink) the values of x to have a similar scale across different coordinates, preventing the loss of significant digits during division. Fig. 2 (middle) illustrates our HE-friendly training variant, built on top of Eqs. 5 and 6, compared to the original attention.

4.4 Length-Agnostic Range for Polynomial Division

The only non-polynomial operation in Eq. 2 is division, which can be approximated effectively in a

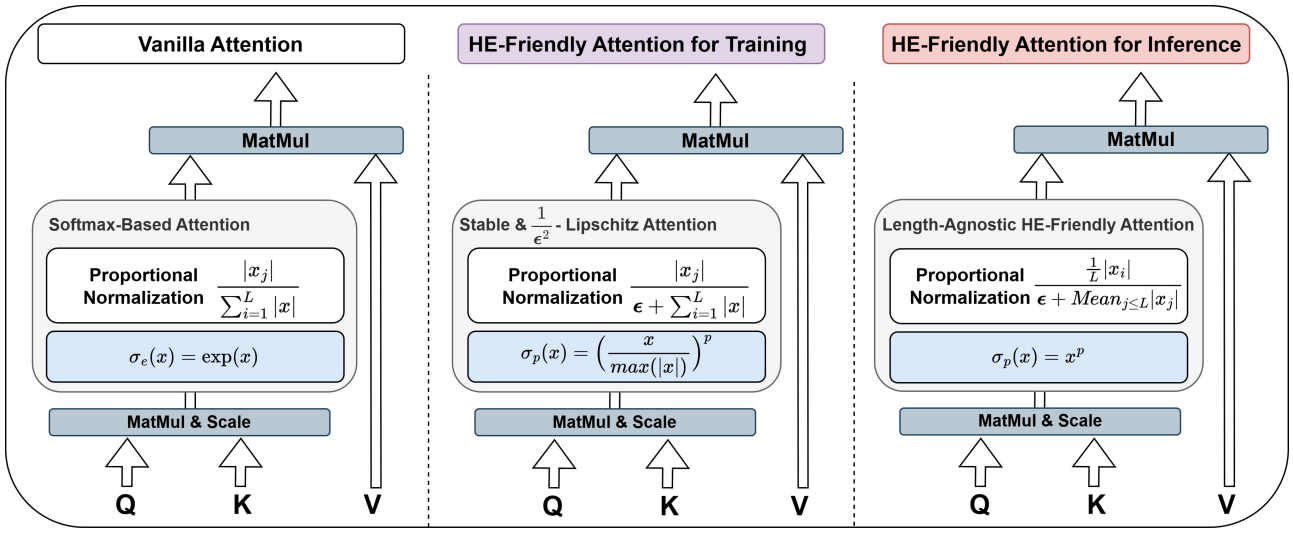

Figure 2: **Our Attention Variants:** (Left) the Softmax-based attention mechanism using the generalized attention formulation (Eq. 3). (Middle) Our variant for training (purple), builds on the stable variant from Eq. 6 and the Lipschitz division from Eq. 5. (Right) During secure inference with the polynomial model (red), we use a length-agnostic approximation for division, as described in Eq. 7.

bounded domain using the Goldschmidt algorithm (Goldschmidt, 1964). However, in our attention variant, we need to approximate the function $\frac{1}{x}$, where x is the sum of the scores raised to the power of p, which is unbounded and increases linearly with the sequence length L. Thus, applying Goldschmidt's algorithm naively would struggle to precisely approximate division for both short and long sentences and would require relatively high-degree polynomials due to the extremely large domain range. To address this problem, we propose a length-agnostic HE-friendly attention:

$$\text{Length-Agnostic PowerSoftmax}(\mathbf{x})_j = \tag{7}$$

$$\frac{\frac{1}{L}x_j^p}{\text{Mean}_{i\leq L}\, x_i^p} = \frac{\left(\frac{x_j}{L'}\right)^p}{\text{Mean}_{i\leq L}\, x_i^p}\,.$$

This variant leverages the fact that the sequence length L is not a secret, so $\frac{1}{L}$ can be computed directly without approximation (or pre-computed by the client). The resulting approximation operates on the mean of the attention scores rather than their sum. Notably, if the attention scores have mean μ and variance σ^2, the asymptotic behavior of both approaches as $L \to \infty$ can be described as follows (by the law of large numbers):

$$\text{Mean}\,\sigma_p\left(\frac{1}{\sqrt{d_k}}QK^T\right) \to \mu \quad \sum \sigma_p\left(\frac{1}{\sqrt{d_k}}QK^T\right) \to \infty\,. \tag{8}$$

This shows that our length-agnostic variant does not become harder to approximate as L increases, enabling a more flexible and precise polynomial approximation. Fig. 2 (right) compares this variant with the original attention.

4.5 A Recipe for Polynomial LLM

Alg. 1 illustrates the entire process, which is divided into three key stages: **(i) Architectural modification:** We begin by modifying the original transformer architecture to use an HE-friendly attention variant (Eq. 6). This modified model is then trained from scratch using the same hyperparameters as the vanilla transformer. **(ii) Range minimization:** In the second stage, we apply a supplementary training procedure as outlined in (Baruch et al., 2025) to ensure that the model operates within HE-friendly constraints. Specifically, we adjust the model's weights so that each non-polynomial component operates only within specific, restricted input domains. This is achieved by adding a regularization loss function that minimizes the range of inputs to non-polynomial layers. For activations and LayerNorm layers, we directly apply the method from (Zimerman et al., 2024).

Additionally, for the HE-friendly attention mechanism, we introduce a tailored loss term defined as:

$$\mathbb{L}_{\text{PowerSoftmax}} := \sum_{n=1}^{N_L} \max_{c\in C}\left\{|z|_{n,c}^i\right\}\,, \tag{9}$$

where we denote the number of attention layers by N_L and the set of heads by C. Additionally, we denote the input at layer n to the PowerSoftmax layer, at head $c \in C$, when the model processes the x_i example by $z_{n,c}^i$. This loss serves two main purposes: First, it minimizes the upper bound of the denominator in the HE-friendly attention, making the approximation problem more tractable. Second, we observed

that when the input norm to the HE-friendly attention is not too high, the stabilize factor defined in Eq. 6 can be omitted, eliminating the need for additional division approximations. **(iii) Polynomial replacement:** In the final stage, each non-polynomial layer is replaced with its polynomial approximation, resulting in a fully polynomial model. App. C provides further details on the approximations used. These approximations are designed to be highly accurate for the HE-friendly weights obtained from the previous stages.

Continual Training A significant limitation of Step 1 in Algorithm 1, compared to PTA methods, is the need for retraining, which can be expensive for large transformers trained on extensive datasets. To mitigate this, we propose a complementary procedure to convert standard pre-trained attention layers into PowerSoftmax layers via a short fine-tuning step. Since both attention variants share the same trainable parameters and perform similar (though not identical) computations (as shown in Fig. 1), we initialize the weights of our attention variant from a vanilla pre-trained reference model. Fine-tuning the resulting model reduces the performance gap between the two variants, enabling us to take advantage of the significant computational investment made in these models.

Wrap-up Finally, we present a unified solution that refines the attention mechanism for FHE constraints. It adopts distinct forms for training and secure inference to tackle two main challenges: ensuring stable large-scale training (via a stable variant with non-polynomial division) and enabling efficient secure inference (through Lipschitz and length-agnostic variants) by minimizing multiplication depth. Furthermore, we demonstrate how this mechanism can be continually pre-trained, thus overcoming limitations of the AAT approach and making it truly scalable.

5 EXPERIMENTS

We now present an empirical evaluation of our method. Sect. 5.1 introduces our polynomial LLMs and reports results on both encrypted and unencrypted data in zero-shot and fine-tuned settings. Sect. 5.2 offers a comprehensive set of ablation studies, providing empirical justifications for the key design decisions of our method, and Sect. 5.3 presents comparisons of our method and other SoTA methods in the domain. Finally, Sect. 5.4 compares the attention matrices generated by the standard Softmax with those produced by our HE-friendly variant, analyzing the differences between these matrices. The experimental setup is detailed in App. A.

Algorithm 1: Polynomial Transformer Construction

Input: A vanilla transformer architecture and hyper-parameters for training.

Output: A polynomial transformer for secure inference.

1. **Architectural modification and pre-training**: Modify the transformer architecture via Eqs. 6 and 5 (stable and Lipschitz HE-friendly variant) and train the new architecture from scratch with the same hyper-parameters.
2. **Range minimization**: Minimize the input range to the GELU, LayerNorm, and PowerSoftmax layers via the loss function defined in Eq. 9.
3. **Polynomial replacement**: Replace the inverse function in HE-friendly attention and the inverse square root in LayerNorm with polynomial approximations obtained from the Goldschmidt method. Replace activations with suitable polynomial approximations (see App. C). Incorporate the length-agnostic approximation (Eq. 7).

5.1 Polynomial LLMs

We experimented with polynomial variants of a causal transformer (GPT) and a bidirectional model.

Causal Transformer For a GPT model, we built upon the Pythia (Biderman et al., 2023) family of models, adapting their training procedures, evaluation methodologies, and hyperparameters. Specifically, we trained two models for next-token prediction (NTP) on the Pile (Gao et al., 2020): a small model with 70M and a large model with 1.4B parameters, using continual pretraining (Sect. 4.5).

We evaluated these models using the popular lm-evaluation-harness framework. Table 1 shows that our models achieve performance comparable to the original models for 5-shot and 0-shot settings. These results mark a significant advancement, as no prior work has introduced polynomial LLMs with demonstrated **ICL or reasoning capabilities**. This is particularly evident on reasoning benchmarks such as the ARC, where our models perform competitively.

Bidirectional Transformer For the bidirectional model, we tested our approach on RoBERTa (Liu, 2019). Starting with a Softmax-based pre-trained transformer, we applied the HE-friendly adaptation using the method described in Sect. 4.5 through continual pre-training on the OpenWebText corpus (Gokaslan and Cohen, 2019). Then, we fine-tuned

Table 1: Comparison of zero-shot and five-shot results between vanilla transformer and our poly. variant across different model sizes. Original models trained on Pile. Results of non-polynomial models copied from (Biderman et al., 2023).

Dataset	Zero-shot				5-shot			
	1.4B		70M		1.4B		70M	
	Orig.	Poly.	Orig.	Poly.	Orig.	Poly.	Orig.	Poly.
Lambada O. Acc	0.610	0.607	0.192	0.258	0.568	0.487	0.134	0.181
PIQA	0.720	0.710	0.598	0.592	0.725	0.720	0.582	0.597
WinoGrande	0.566	0.562	0.492	0.503	0.570	0.568	0.499	0.505
WSC	0.442	0.395	0.365	0.365	0.365	0.548	0.365	0.452
ARC-Easy	0.617	0.602	0.385	0.420	0.633	0.613	0.383	0.387
ARC-Challenge	0.272	0.265	0.162	0.185	0.276	0.277	0.178	0.183
SciQ	0.865	0.873	0.606	0.716	0.926	0.907	0.598	0.718
LogiQA	0.221	0.217	0.235	0.210	0.230	0.222	0.250	0.238

our model on three datasets from the GLUE benchmark (Wang, 2018) separately, adapting RoBERTa's fine-tuning process, and finally approximating the non-polynomial components. The results are depicted in Tab. 2, and compared with the work of Zhang et al. (2024a). The full configuration is detailed in App. A.2. These results indicate a degradation of approximately 1% compared to the original RoBERTa.

Table 2: Downstream GLUE results for polynomial RoBERTa-Base. Results from (Zhang et al., 2024b) are denoted by $^{\diamond}$.

Model	Dataset		
	SST-2	QNLI	MNLI
RoBERTa	94.80	92.80	87.60
Poly-RoBERTa	93.35	91.62	86.93
Nexus (BERT) $^{\diamond}$	92.11	89.90	N.A

Experiments over HE To demonstrate the feasibility of our method under FHE, we conducted an experiment designed to show that the accuracy of both encrypted and plaintext inference is similar. For this, we implemented polynomial Pythia 70M using HElayers 1.5.4 (Aharoni et al., 2023a), and evaluated the model over 100 samples twice. We observed a maximal MSE loss of 0.005 between the encrypted and non-encrypted inference outputs, without modifying the maximal (predicted) value in 99% of the cases. While the scale of the experiment was limited to Pythia 70M, we chose this model because the encrypted inference time takes 93 seconds per sample on a single A100 GPU. For full details of the workflow under HE, see App. G. Please note that we conducted experiments over 128 tokens, following previous work (Zhang et al., 2024b; Cho et al., 2024). However, practical scenarios and LLMs often process much longer sequences with thousands of tokens. In these scenarios, the proportional cost of the Softmax compared to other components drastically increases, as it is the only component whose complexity scales quadratically rather than linearly with sequence length. This rise in the proportional cost of Softmax makes our method much more efficient for long-context, as other methods like (Zhang et al., 2024b; Cho et al., 2024) require computing the maximal value over each row of the attention matrix, resulting in a gap that increases with context length.

5.2 Ablation Studies

We perform the following series of ablation studies:

PowerSoftmax Attention We first compare Softmax and PowerSoftmax outside the context of HE, showing that in addition to being an HE-friendly variant, PowerSoftmax also exhibits similar scaling trends as Softmax. Figures 3 and 9 (in App. E) present comparative visualizations of training curves for various model sizes and datasets (including Pile, Wikitext-103, Text-8, Tiny-Imagenet, CIFAR-100 and CIFAR-10) across both NLP and vision domains, respectively. Although Softmax generally achieves better results, it is evident that by the end of training, most of the gap between the models is reduced, and the scaling laws of the models are relatively similar.

Stability To assess the contribution of our numerically stable variant, we conduct dedicated experiments. In Fig. 4, we provide training curves for models with 32 layers and a hidden dimension size of 1024, trained on 10% of the Wikitext-103 dataset. We compare two PowerSoftmax-based transformers with the same training procedure, one with (blue) and one without (red), the stable variant from Eq. 6. As an additional baseline, we trained a vanilla transformer (black). As shown, the stable variant consistently outperforms the PowerSoftmax baseline, closing a third of the gap between the PowerSoftmax and the Softmax baseline. Additionally, we observe that in more challenging regimes, such as training on the full dataset

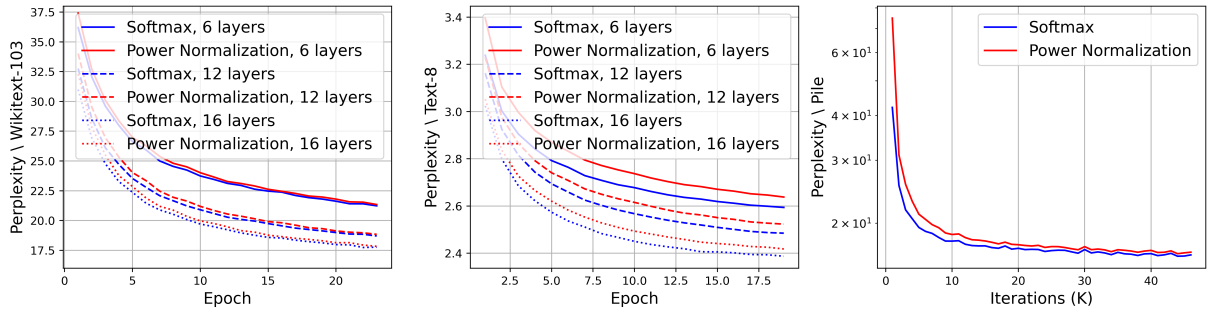

Figure 3: **Training curves for NTP:** Comparison of test perplexity for transformers with Softmax and PowerSoftmax when trained over several datasets including Pile, Wikitext-103, and Text-8.

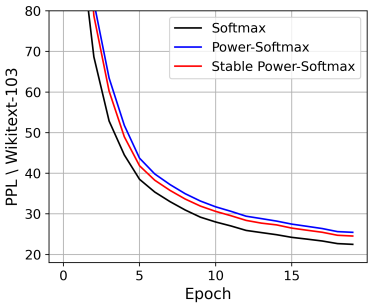

Figure 4: **The Significance of the stable variant.** Training curves for NTP on Wikitext for large models. The stable variant (red) consistently outperforms the vanilla PowerSoftmax (blue).

or other datasets, the stable variant is much more robust to optimization issues and less sensitive for hyperpramtaer tuning.

ϵ-bounded division for Softmax The HE-friendly attention variant from Eq. 5 proposes adding ϵ to make the approximation problem of division easier, resulting in an approximation of a $\frac{1}{\epsilon^2}$-Lipschitz continuous function. Fig. 5 empirically supports this evidence by showing that the approximation error obtained by the Goldsmith method decreases as ϵ increases. Additionally, Fig. 7 in App. B shows that higher ϵ values improve the training dynamics.

5.3 Comparisons with SotA Methods

To the best of our knowledge, only two prior efforts have successfully presented fully polynomial transformers executed over FHE: (i) Zimerman et al. (2024), who employ the AAT approach, and (ii) (Zhang et al., 2024a), who introduced NEXUS and focus on the PTA regime. We begin by noting that our method exhibits superior scaling properties compared to both methods. This is evidenced by the fact that both methods concentrated on relatively simple text classification tasks, such as those found in the GLUE benchmark, with or without pre-training. In contrast, our models tackle much more complex tasks, including those that require *reasoning and ICL capabilities*, which are associated with LLMs.

Additionally, when operating over encrypted data, our model is significantly more efficient than both methods. Specifically, Nexus incorporates three high-degree polynomial approximations per attention layer (for the exponential, division, and maximum functions), whereas our approach requires only a single non-polynomial division. Furthermore, Zimerman et al. (2024) exhibits substantially worse scaling properties for large models, and we were unable to successfully train deep transformers with 32 layers using their method. One possible explanation is that their pointwise attention lacks score normalization, resulting in training instability. A comparison is provided in Fig. 6 (App. A.2). Although both methods include a single non-polynomial operation per attention head, ours is far more efficient for long contexts: their method applies an activation to each element of the attention matrix, yielding L^2 deep polynomial evaluations per head, whereas ours applies division once per row, yielding only L deep polynomials and fewer HE bootstrap operations. For additional empirical analysis, see App. G.

Beyond fully homomorphic encryption (FHE), several multi-party computation (MPC)-based secure LLM inference protocols exist, for example, (Xue et al., 2025; Dong et al., 2025), but they typically require gigabytes of communication overhead, making deployment impractical compared to FHE-centric approaches.

5.4 Analyzing Attention Matrices

PowerSoftmax introduces an important hyperparameter p that differentiates it from the traditional Softmax function. To better understand its mechanistic behavior, we examine how the attention matrices evolve with varying values of p. Our analysis reveals that as p increases, the resulting attention matrices become more

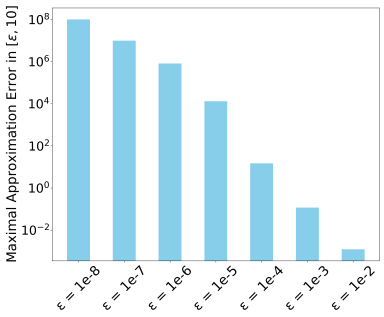

Figure 5: Attention mean distance for different transformer variants.

localized as depicted in Fig. 13. For instance, by comparing the first column (PowerSoftmax with $p = 4$) with the third column ($p = 12$), we observe a significantly stronger diagonal in the latter, whereas the $p = 4$ model displays a more uniform attention distribution. Additionally, we empirically confirm this pattern by analyzing the average of the mean attention distance (Vig and Belinkov, 2019) per model (i.e., averaged across all the layers and heads) as illustrated in Fig. 12. Moreover, we observe that later layers tend to exhibit more longer-distance relationships compared to earlier layers in both PowerSoftmax and Softmax. This finding is consistent with previous research (Vig and Belinkov, 2019). Additional analysis and visualizations can be found in in Appendix J.

6 CONCLUSION AND LIMITATIONS

We presented a method for training polynomial LLMs with approximately 1.4 billion parameters, significantly larger than those employed in previous works. For that, we introduced an HE-friendly alternative to self-attention, which we demonstrate performs comparably to the original model. This variant allows us to present the first polynomial LLM with zero-shot and reasoning capabilities. Despite the promising results, a full evaluation of the auto-regressive generative abilities of our models in both sequential decoding over plain and encrypted environments has not yet been conducted. For future work, we plan to investigate these aspects further and explore techniques to reduce the model's latency when operating on encrypted data.

References

Allon Adir, Ehud Aharoni, Nir Drucker, Ronen Levy, Hayim Shaul, and Omri Soceanu. *Homomorphic Encryption for Data Science (HE4DS)*. Springer, 2024. doi:https://doi.org/10.1007/978-3-031-65494-7.

Ehud Aharoni, Allon Adir, Moran Baruch, Nir Drucker, Gilad Ezov, Ariel Farkash, Lev Greenberg, Ramy Masalha, Guy Moshkowich, Dov Murik, et al. HElayers: A tile tensors framework for large neural networks on encrypted data. *PoPETs*, 2023a. doi:10.56553/popets-2023-0020.

Ehud Aharoni, Nir Drucker, and Hayim Shaul. Tutorial-HEPack4ML '23: Advanced HE Packing Methods with Applications to ML. In *Proceedings of the 2023 Tutorial on Advanced HE Packing Methods with Applications to ML*, Tutorial-HEPack4ML '23, page 1–2, New York, NY, USA, 2023b. Association for Computing Machinery. ISBN 9798400702679. doi:10.1145/3605774.3625525.

Adi Akavia and Margarita Vald. On the privacy of protocols based on cpa-secure homomorphic encryption. *IACR Cryptol. ePrint Arch.*, 2021:803, 2021. URL `https://eprint.iacr.org/2021/803`.

Alex Andonian, Quentin Anthony, Stella Biderman, Sid Black, Preetham Gali, Leo Gao, Eric Hallahan, Josh Levy-Kramer, Connor Leahy, Lucas Nestler, Kip Parker, Michael Pieler, Jason Phang, Shivanshu Purohit, Hailey Schoelkopf, Dashiell Stander, Tri Songz, Curt Tigges, Benjamin Thérien, Phil Wang, and Samuel Weinbach. Gpt-neox: Large scale autoregressive language modeling in pytorch, 9 2023. URL `https://www.github.com/eleutherai/gpt-neox`.

Wei Ao and Vishnu Naresh Boddeti. AutoFHE: Automated adaption of CNNs for efficient evaluation over FHE. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 2173–2190, Philadelphia, PA, August 2024. USENIX Association. ISBN 978-1-939133-44-1. URL `https://www.usenix.org/conference/usenixsecurity24/presentation/ao`.

Matan Avitan, Moran Baruch, Nir Drucker, Itamar Zimerman, and Yoav Goldberg. Efficient decoding methods for language models on encrypted data. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 1777–1794, 2025.

Moran Baruch, Nir Drucker, Lev Greenberg, and Guy Moshkowich. A Methodology for Training Homomorphic Encryption Friendly Neural Networks. In *Applied Cryptography and Network Security Workshops*, pages 536–553, Cham, 2022. Springer International Publishing. ISBN 978-3-031-16815-4. doi:10.1007/978-3-031-16815-4_29.

Moran Baruch, Nir Drucker, Gilad Ezov, Yoav Goldberg, Eyal Kushnir, Jenny Lerner, Omri Soceanu, and Itamar Zimerman. Polynomial Adaptation of Large-Scale CNNs for Homomorphic Encryption-Based Secure Inference. In Shlomi Dolev, Michael

Elhadad, Mirosław Kutyłowski, and Giuseppe Persiano, editors, *Cyber Security, Cryptology, and Machine Learning*, pages 3–25, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-76934-4. doi:10.1007/978-3-031-76934-4_1.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, Usvsn Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. Pythia: A suite for analyzing large language models across training and scaling. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 2397–2430. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/biderman23a.html.

Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) Fully Homomorphic Encryption without Bootstrapping. *ACM Trans. Comput. Theory*, 6(3), July 2014. ISSN 1942-3454. doi:10.1145/2633600.

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. *arXiv preprint arXiv:2206.00216*, 2022. URL https://arxiv.org/abs/2206.00216.

Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 409–437. Springer, 2017. doi:10.1007/978-3-319-70694-8_15.

Wonhee Cho, Guillaume Hanrot, Taeseong Kim, Minje Park, and Damien Stehlé. Fast and accurate homomorphic softmax evaluation. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, page 4391–4404, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400706363. doi:10.1145/3658644.3670369.

Grigorios G Chrysos, Stylianos Moschoglou, Giorgos Bouritsas, Yannis Panagakis, Jiankang Deng, and Stefanos Zafeiriou. P-nets: Deep polynomial neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7325–7335, 2020. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Chrysos_P-nets_Deep_Polynomial_Neural_Networks_CVPR_2020_paper.html.

Yuanchao Ding, Hua Guo, Yewei Guan, Weixin Liu, Jiarong Huo, Zhenyu Guan, and Xiyong Zhang. East: Efficient and accurate secure transformer framework for inference. *arXiv preprint arXiv:2308.09923*, 2023. URL https://arxiv.org/abs/2308.09923.

Ye Dong, Wen-jie Lu, Yancheng Zheng, Haoqi Wu, Derun Zhao, Jin Tan, Zhicong Huang, Cheng Hong, Tao Wei, Wen-Guang Chen, et al. Puma: Secure inference of llama-7b in five minutes. *Security and Safety*, 4:2025014, 2025.

Nir Drucker and Itamar Zimerman. Efficient skip connections realization for secure inference on encrypted data. In Shlomi Dolev, Ehud Gudes, and Pascal Paillier, editors, *Cyber Security, Cryptology, and Machine Learning*, pages 65–73, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-34671-2. doi:10.1007/978-3-031-34671-2_5.

Junfeng Fan and Frederik Vercauteren. Somewhat Practical Fully Homomorphic Encryption. *Proceedings of the 15th international conference on Practice and Theory in Public Key Cryptography*, pages 1–16, 2012. URL https://eprint.iacr.org/2012/144.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020. URL https://arxiv.org/abs/2101.00027.

Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, Palo Alto, CA, 2009. URL https://crypto.stanford.edu/craig/craig-thesis.pdf.

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 *International conference on machine learning*, pages 201–210. PMLR, 2016. URL http://proceedings.mlr.press/v48/gilad-bachrach16.pdf.

Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Robert E Goldschmidt. *Applications of division by convergence*. PhD thesis, Massachusetts Institute of Technology, 1964. URL https://dspace.mit.edu/bitstream/handle/1721.1/11113/34136725-MIT.pdf.

Vikas Gottemukkula. Polynomial activation functions. *OpenReview*, 2020. URL https://openreview.net/forum?id=rkxsgkHKvH.

Mohit Goyal, Rajan Goyal, and Brejesh Lall. Improved polynomial neural networks with normalised activa-

tions. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020. doi:10.1109/IJCNN48605.2020.9207535.

Kanav Gupta, Neha Jawalkar, Ananta Mukherjee, Nishanth Chandran, Divya Gupta, Ashish Panwar, and Rahul Sharma. SIGMA: Secure GPT inference with function secret sharing. *Cryptology ePrint Archive*, 2023. URL https://eprint.iacr.org/2023/1269.

Shai Halevi and Victor Shoup. Faster homomorphic linear transformations in helib. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, pages 93–120, Cham, 2018. Springer International Publishing. doi:10.1007/978-3-319-96884-1_4.

Xiaoqian Jiang, Miran Kim, Kristin Lauter, and Yongsoo Song. Secure outsourced matrix computation and application to neural networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 1209–1222, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356930. doi:10.1145/3243734.3243837.

Jae Hyung Ju, Jaiyoung Park, Jongmin Kim, Donghwan Kim, and Jung Ho Ahn. Neujeans: Private neural network inference with joint optimization of convolution and bootstrapping. *arXiv preprint arXiv:2312.04356*, 2023. URL https://arxiv.org/abs/2312.04356.

Eunsang Lee, Joon-Woo Lee, Junghyun Lee, Young-Sik Kim, Yongjune Kim, Jong-Seon No, and Woosuk Choi. Low-complexity deep convolutional neural networks on fully homomorphic encryption using multiplexed parallel convolutions. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 12403–12422. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/lee22e.html.

Junghyun Lee, Eunsang Lee, Joon-Woo Lee, Yongjune Kim, Young-Sik Kim, and Jong-Seon No. Precise approximation of convolutional neural networks for homomorphically encrypted data. *arXiv preprint arXiv:2105.10879*, 2021. URL https://arxiv.org/abs/2105.10879.

Junghyun Lee, Eunsang Lee, Young-Sik Kim, Yongwoo Lee, Joon-Woo Lee, Yongjune Kim, and Jong-Seon No. Optimizing layerwise polynomial approximation for efficient private inference on fully homomorphic encryption: A dynamic programming approach. *arXiv preprint arXiv:2310.10349*, 2023. URL https://arxiv.org/abs/2310.10349.

Zi Liang, Pinghui Wang, Ruofei Zhang, Nuo Xu, Shuo Zhang, Lifeng Xing, Haitao Bai, and Ziyang Zhou. MERGE: Fast private text generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(18):19884–19892, Mar. 2024. doi:10.1609/aaai.v38i18.29964.

Yinhan Liu. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019. URL https://arxiv.org/abs/1907.11692.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. URL https://arxiv.org/abs/1706.03762.

Jesse Vig and Yonatan Belinkov. Analyzing the structure of attention in a transformer language model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, Florence, Italy, August 2019. Association for Computational Linguistics. doi:10.18653/v1/W19-4808. URL https://aclanthology.org/W19-4808.

Alex Wang. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Yujie Xue, Lin Liu, Yuchuan Luo, Bing Sun, and Shaojing Fu. Privmllm: Efficient three-party multimodal large language model secure inference supported prompt privacy. In *2025 IEEE 33rd International Conference on Network Protocols (ICNP)*, pages 1–24, 2025. doi:10.1109/ICNP65844.2025.11192338.

Biwei Yan, Kun Li, Minghui Xu, Yueyan Dong, Yue Zhang, Zhaochun Ren, and Xiuzheng Cheng. On protecting the data privacy of large language models (LLMs): A survey. *arXiv preprint arXiv:2403.05156*, 2024. URL https://arxiv.org/abs/2403.05156.

Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing*, 4(2):100211, 2024. ISSN 2667-2952. doi:https://doi.org/10.1016/j.hcc.2024.100211.

Chi Zhang, Man Ho Au, and Siu Ming Yiu. Neural networks with (low-precision) polynomial approximations: New insights and techniques for accuracy im-

provement. *arXiv preprint arXiv:2402.11224*, 2024a. URL https://arxiv.org/abs/2402.11224.

Jiawen Zhang, Jian Liu, Xinpeng Yang, Yinghao Wang, Kejia Chen, Xiaoyang Hou, Kui Ren, and Xiaohu Yang. Secure transformer inference made non-interactive. *Cryptology ePrint Archive*, 2024b. URL https://eprint.iacr.org/2024/136.

Mengxin Zheng, Qian Lou, and Lei Jiang. Primer: Fast private transformer inference on encrypted data. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2023. doi:10.1109/DAC56929.2023.10247719.

Jun Zhou, Huimin Qian, Xinbiao Lu, Zhaoxia Duan, Haoqian Huang, and Zhen Shao. Polynomial activation neural networks: Modeling, stability analysis and coverage bp-training. *Neurocomputing*, 359:227–240, 2019. ISSN 0925-2312. doi:https://doi.org/10.1016/j.neucom.2019.06.004.

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 Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 62803–62814. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/zimerman24a.html.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. Yes
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. Not Applicable
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. Yes

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. Yes
   (b) Complete proofs of all theoretical results. Yes
   (c) Clear explanations of any assumptions. Yes

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). Yes
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). Yes
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). Yes
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). Yes

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. Yes
   (b) The license information of the assets, if applicable. Not Applicable
   (c) New assets either in the supplemental material or as a URL, if applicable. Not Applicable
   (d) Information about consent from data providers/curators. Not Applicable
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. Not Applicable

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. Not Applicable
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. Not Applicable
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. Not Applicable

A EXPERIMENTAL SETUP AND HYPER-PARAMETERS

All training experiments were conducted on public datasets using the PyTorch framework. Results were averaged over three random seeds, with experiments running on two A100 80GB GPUs for a maximum of two days, except for those involving the Pile dataset, which were run for up to three days on eight A100 40GB GPUs.

A.1 GPT

We used the framework of neox-gpt[1] (Andonian et al., 2023) with its configuration of Pythia to train the 70M and 1.4B models. For this process. The replacement process is done as follows:

1. Load a checkpoint of the pre-trained model.
2. Replace Softmax with PowerSoftmax, with $p = 4$, and employ continual pre-training over the Pile dataset for 100 iterations.
3. Finetune the model with range-loss to minimize c and the input to GELU. This process takes around 17K iterations.
4. Apply polynomial approximation.

Table 3 shows the specific hyperparameters used for this process.

Table 3: HE-related configuration for Pythia 1.4B, 70M, and RoBERTa models

Parameter	**GPT 1.4B**	**GPT 70M**	**RoBERTa-Base**
Sum power weights epsilon	$1e{-}4$	0.001	$1e{-}4$
PowerSoftmax loss weight (c)	$1e{-}4$	$1e{-}4$	0.01
GELU loss weight	0.001	$1e{-}4$	0
Learning rate	$4e{-}5$	$1e{-}4$	$1e{-}4$

A.2 RoBERTa

We employed the RoBERTa framework [2] (Ott et al., 2019) and configuration to train and fine-tune the base model with 125M parameters for three GLUE tasks: SST-2, QNLI, and MNLI. The process was carried out as follows:

1. Load a checkpoint of the pre-trained base model.
2. Replace Softmax with PowerSoftmax, with $p = 6$, and continually pre-train the model on the OpenWebText dataset for 1250 iterations.
3. Fine-tune the model individually for each of the three GLUE tasks for up to 10 epochs. This fine-tuning followed the procedure described in the original RoBERTa paper, except for substituting the Tanh activation function in the classification head with a Sigmoid, which we found to perform better under HE.
4. Perform an additional fine-tuning step using range-loss with PowerSoftmax loss weight for 10 epochs. The GELU ranges were narrow enough and did not require tuning.
5. Apply polynomial approximation.

We reported accuracy results in Tab. 2. See Table 3 for the specific hyperparameters.

Additionally, we train RoBERTa models with 12 layers from scratch over the Wikitext-103 benchmark for three types of attention: (i) Softmax (black), (ii) our PowerSoftmax (blue), and (iii) the Scaled-ReLU (red) attention

[1] `https://github.com/EleutherAI/gpt-neox`

[2] `https://github.com/facebookresearch/fairseq/blob/main/examples/roberta`

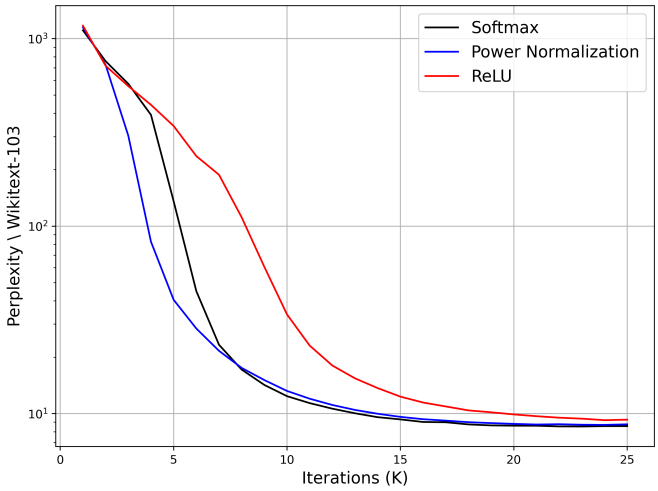

Figure 6: Comparison of training curves for 12-layer RoBERTa models with different attention mechanisms on the Wikitext-103 benchmark. The PowerSoftmax variant (blue) converges faster than Softmax (black), while the Scaled-ReLU baseline (red) underperforms. Curves are averaged over three seeds.

baseline of Zimerman et al. (2024), all using the same training procedure and hyperparameters optimized for the vanilla Softmax-based transformer. Training curves are averaged over three seeds and presented in Fig. 6. As shown, the Scaled ReLU variant is not competitive with the variants that employ proportional normalization. While Softmax achieves better final results, it converges slightly slower than the PowerSoftmax variant. With the implementation of early stopping, the models achieved average perplexity of 8.48 for Softmax, 8.69 for PowerSoftmax, with the Scaled ReLU lagging behind with 9.12.

B ADDITIONAL ABLATION STUDIES

To gain a clearer understanding of the impact of ϵ in PowerSoftmax-based attention models, we trained several models using different values of ϵ. As shown in Figure 7, our variants demonstrate robustness across various ϵ values in terms of training dynamics. However, Figure 5 shows that for larger values of ϵ, the resulting approximation function for division becomes easier, and we consider these settings (as an example $\epsilon = 1e-2$) to be preferred.

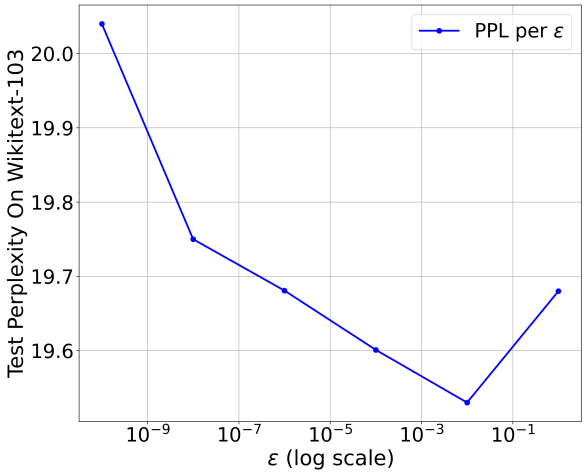

Figure 7: The impact of different values of ϵ on training dynamics of PowerSoftmax-based models

C OUR POLYNOMIAL APPROXIMATIONS

Our PowerSoftmax-based transformers utilize three polynomial approximations. For the division in PowerSoftmax and the $\frac{1}{\sqrt{x}}$ function in LayerNorm, we apply the Goldschmidt approximation, following previous work in the domain (Zimerman et al., 2024; Zhang et al., 2024a). For the GELU approximation, we use the following identity to reduce the problem to approximating the Sigmoid function, which has been extensively explored in previous research in the HE domain.

$$GELU(x) = x \cdot \text{Sigmoid}(1.702 \cdot x). \tag{10}$$

D ANALYSIS OF THE PARAMETER P

In this section, we provide a detailed analysis of the parameter P and its role in shaping training dynamics. Our empirical study demonstrates that setting $4 \leq P \leq 8$ consistently yields the best results across various tasks and datasets. A representative example is illustrated in Figure 8, which shows training curves of several PowerSoftmax-based language models trained for next-token prediction on WikiText-103. The figure depicts perplexity across epochs for different P values.

It can be observed that when P is set too high, training instabilities arise. For instance, models with $P > 8$ exhibit oscillations in perplexity, indicating numerical issues. While this instability can be mitigated using our stable variant, the results do not outperform those achieved with $4 \leq P \leq 8$.

Conversely, when P is too low (e.g., $P = 2$), the results are sub-optimal. One possible explanation is that the super-linear trend is overly conservative, limiting the model's capacity to capture complex relationships in the data adequately.

Thus, to maintain both performance and computational efficiency with low multiplication depth, we select $P = 4$ for all models.

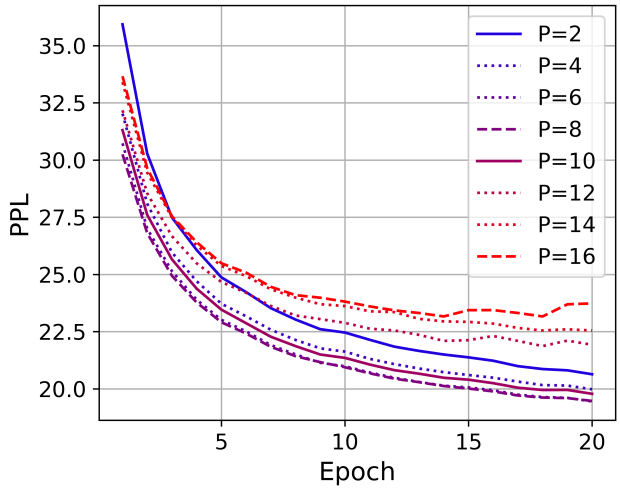

Figure 8: The impact of P during training.

E ADDITIONAL EMPIRICAL ANALYSIS

In addition to the analysis in Figure 3, we further explore the differences between Softmax and PowerSoftmax in vision by conducting experiments on Tiny-ImageNet, CIFAR-100, and CIFAR-10. The results are reported in Figure 9, showing that by the end of training, the models achieve similar performance.

To provide a clear overview of the final performance of the fully trained models, we include Tab. 4 summarizing the results across six datasets and various model sizes. The table consolidates data from Fig. 3 and Fig. 9,

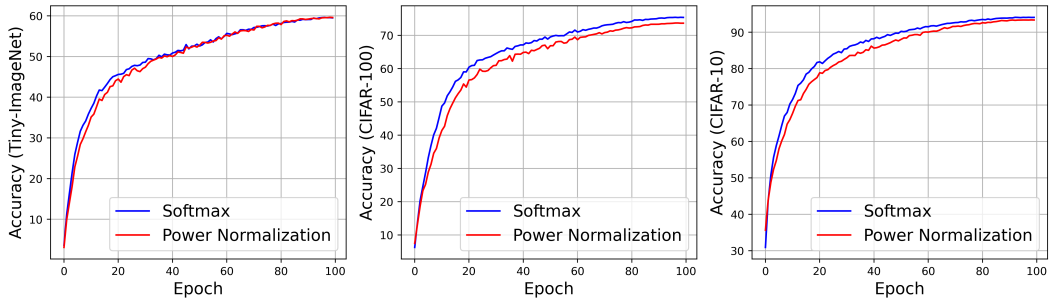

Figure 9: **Results on vision tasks.** Training curves for ViT variants with PowerSoftmax (red) and the Softmax baseline (blue). On the left, results are presented for Tiny-ImageNet and on the middle and right for CIFAR-100 and CIFAR-10 accordingly.

showcasing perplexity for language modeling tasks and accuracy for image classification tasks. This summary highlights the consistent performance of our PowerSoftmax variant compared to the baseline Softmax models, across different configurations and datasets.

Table 4: **Comparison of fully-trained models with** Softmax **and** PowerSoftmax **variants.** Results include perplexity (PPL, lower is better, ↓) for language modeling datasets and accuracy (denoted by 'Top-1') higher is better, ↑) for image classification datasets.

Dataset	**Metric**	**Softmax**	**PowerSoftmax**
WikiText-103 (6 layers)	PPL (↓)	21.18	21.27
WikiText-103 (12 layers)	PPL (↓)	18.60	18.76
WikiText-103 (16 layers)	PPL (↓)	17.64	17.81
Text-8 (6 layers)	PPL (↓)	2.585	2.630
Text-8 (12 layers)	PPL (↓)	2.477	2.515
Text-8 (16 layers)	PPL (↓)	2.383	2.413
Pile	PPL (↓)	16.05	16.51
Tiny-ImageNet	Top-1 (↑)	59.48	59.58
CIFAR-100	Top-1 (↑)	75.39	73.66
CIFAR-10	Top-1 (↑)	94.07	93.36

F IMPACT STATEMENT

Our research introduces the first polynomial LLM, enabling HE-based secure inference with transformers with billion parameters over encrypted data and through encrypted weights, and an HE-friendly transformer architecture. This advancement contributes to privacy-preserving deep learning, offering significant implications for data-sensitive sectors like healthcare and finance. This work aligns with the ethical need for responsible AI development by enhancing data privacy.

G EXPERIMENTS OVER HE

Our experiments used HElayers 1.5.4 (Aharoni et al., 2023a), configured for CKKS with 128-bit security and polynomial degree of 2^{16}. The underlying CKKS library was HEaaN, using its FGb parameter set with the following specific values: $log(QP) = 1555$, $N = 2^{16}$, $L = 9$, $h = 192$, $\lambda = 128$.

Figures 10 and 11 illustrate the packing and chain-index flow of secure inference on the Pythia 70M model compiled with HElayers under FHE. For brevity, we included only the major nodes, such as plaintext matrix multiplication (PMM), ciphertext matrix multiplication (CMM), layer normalization, and the rotary positional embedding (RoPE) layers. The outputs and inputs of nodes include a tuple of information consisting of the tile tensor shape (packing method), the number of tiles (ciphertexts) at this point, and the chain index (ci) of the

ciphertexts. Blue arrows indicate direct input/output data, while orange dashed arrows represent skipped trivial low-latency computation nodes in the figure. Whenever bootstrapping was required, a green arrow indicates the number of ciphertexts that needed bootstrapping. We distinguish between sequential (Seq) and parallel (Par) bootstrapping as follows: By writing *4 Seq Bootstraps*, we mean that the same input tiles were consumed through their multiplication depth 4 times, requiring 4 bootstraps at this node. An example of such a scenario is performing the Goldschmidt method with 21 iterations (i.e., a multiplication depth of 22), which requires $22/9 \leq 3$ sequential bootstraps. In contrast, by writing *4 Par Bootstraps*, we refer to bootstrapping 8 ciphertexts in parallel by using complex plaintext numbers (and not evaluating 4 bootstraps in parallel using threads, as we use a single-threaded GPU).

We use the tile tensor language (Aharoni et al., 2023a) to describe the packing method used by HElayers, ensuring consistency with HElayers logs and because we believe it makes the flow easier to read and understand. A detailed description of the tile tensor language is beyond the scope of this paper, and we refer the reader to (Adir et al., 2024)[Chapters 7-9], for a complete explanation. Alternatively, a good tutorial on tile tensors is provided in (Aharoni et al., 2023b).

For plaintext matrix multiplication, HElayers uses the diagonalization method from (Halevi and Shoup, 2018), extended to the matrix-matrix case, as described in (Adir et al., 2024). For ciphertext matrix multiplication, HElayers uses the method from (Jiang et al., 2018), which requires a multiplication depth of 3 per operation.

The latency of running one secure instance was 93 seconds. The PMMs took 28 seconds (30%), multiplying by W_q, W_k, and W_v requiring approximately 0.32 seconds each for the 18 PMM operations (total: 5.814 seconds). The (h24h/4h2h) PMMs took about 1.88 and 1.242 seconds, respectively (total: 18.732 seconds). The remaining time was spent on the 6 linear PMM operations. The CMM operations took 17 seconds (18%), GELU and the Goldschmidt Inverse-SQRT took 0.65 seconds each (total: 7.7 seconds or 8% of the total latency), and PowerSoftmax took 1.475 seconds each (total: 8.82 seconds or 9.5%). Other bootstrap operations took 18.563 seconds (19.4%), while the remaining 15 seconds out of the 93 seconds (16%) were spent equally on lower-latency operations such as addition, subtraction, rescaling, and repacking. Among the most time-consuming FHE operations, bootstrap took 33 seconds (34%), encoding took 25 seconds (27%), and rotation took 13.623 seconds (14%).

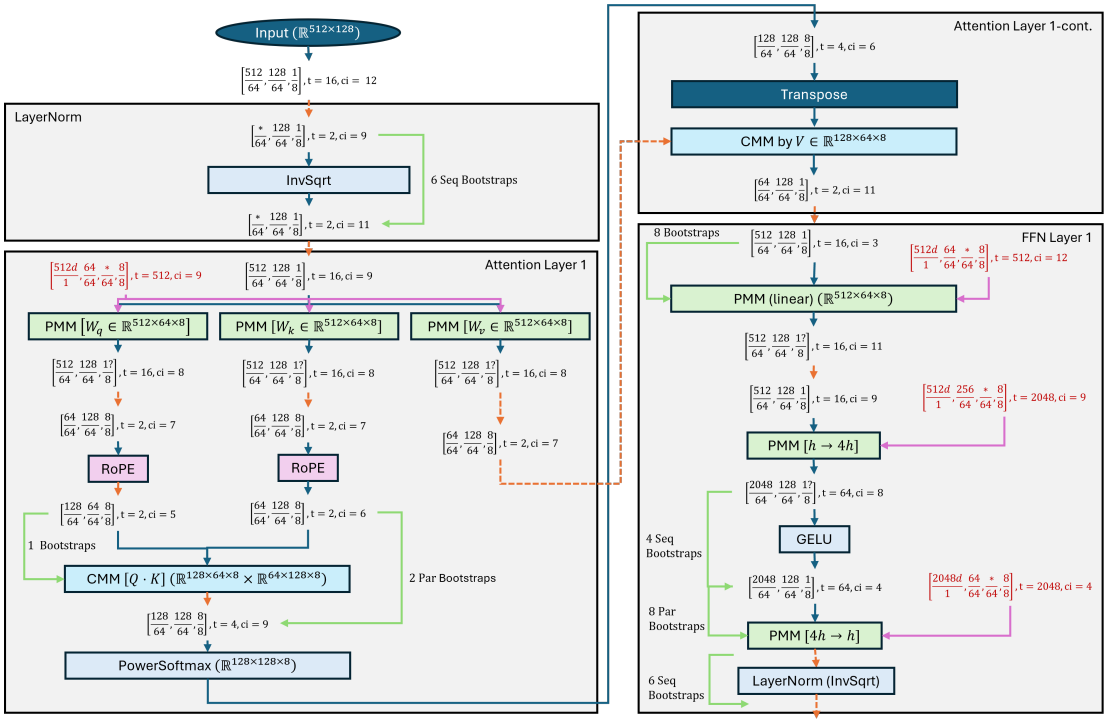

Figure 10: An illustration of the HE-encrypted Pythia 70M model up to the end of the first transformer layer. See the text for details and Figure 11 for the rest of the layers.

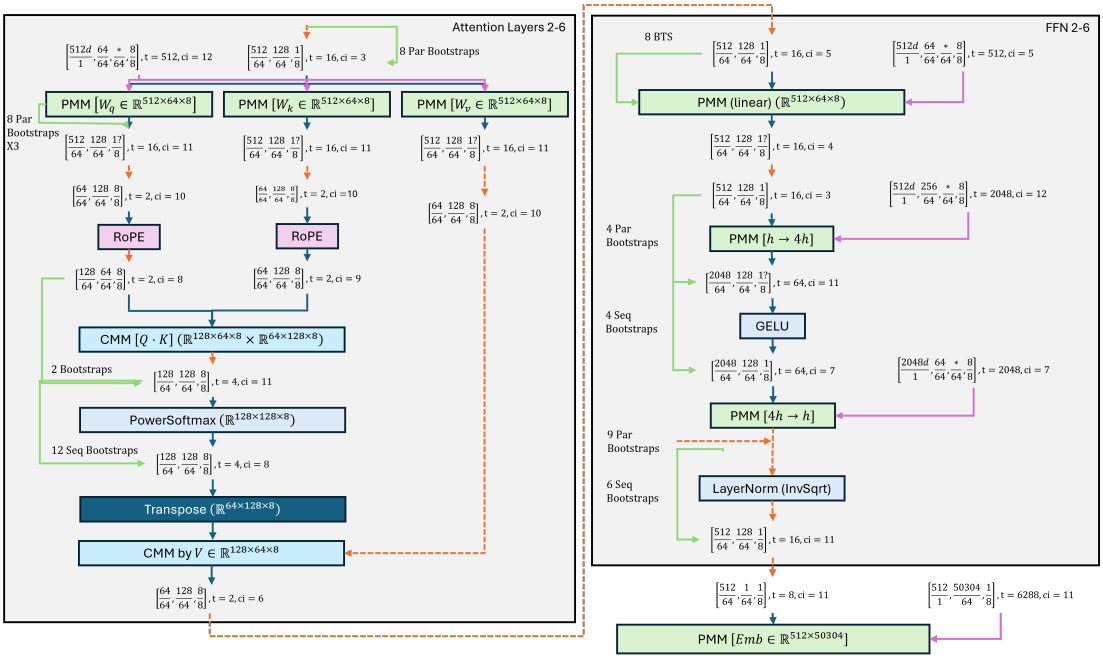

Figure 11: An illustration of the last five transformer layers of the HE-encrypted Pythia 70M model. See the text for details.

Comparing PowerSoftmax with Sota The latency reported by HElayers on one A100 80GB GPU for our PowerSoftmax over 32 layers of 128 tokens (128×128) took only 16 seconds (and 0.6 seconds for 1×32768 input). In contrast, prior SotA – NEXUS (Zhang et al., 2024b)[Table IV] reported that Softmax using 4 (instead of 1) A100 GPUs over 128×128 blocks for 12 (instead of 32) layers took 1.15 seconds * 32 (batch size) = 36.8 seconds, $\times 2.3$ slower. Moreover, following the publication of this paper on arXiv, another study (Cho et al., 2024) addressing Softmax was published. This paper introduces a new Softmax approximation. Compared to (Cho et al., 2024), our method reduces multiplication depth by at least 23% when replacing Softmax with our PowerSoftmax in fine-tuned models. Moreover, the latency reported by (Cho et al., 2024) on an NVIDIA RTX-6000 for $128 \times 128 \times 32$(layers) $+ 1 \times 32768$ inputs was 90 seconds, slower than when using PowerSoftmax.

To further extend our empirical analysis of PowerSoftmax over HE, we conducted additional dedicated experiments isolating the impact of the algorithm while eliminating the effects of different hardware, other aspects of model architecture, and HE libraries. All models were run under HE layers with the parameters described above, using a single A100 GPU. We chose to focus on the following baseline methods, as to the best of our knowledge, these are the only methods that propose algorithms for softmax that have been shown to work with fully polynomial models at relatively large scale: (i) Nexus, by Zhang et al. (2024b), which uses an 8-degree Taylor approximation for the exponent in softmax and Goldschmidt's approximation for division without specifying the number of iterations (we use 21, as in PowerSoftmax); and (ii) the method proposed by Cho et al. (2024) (see Alg. 2 in their work), which employs the normalize-and-square strategy for softmax evaluation, resulting in a multiplication depth of 48. Results are described in able 5 shows that in the tested scenarios, our method reduces latency by a factor of 20 compared to the square strategy, and by a factor of 9.7 compared to the Taylor and Goldschmidt approaches, both on Pythia 70M with 1024 tokens. Similar trends are observed in other regimes, such as shorter contexts and different models, highlighting the effectiveness of our method.

H THREAT MODEL

We consider a two-party FHE scenario where a semi-honest server runs inference and a data-owner submits encrypted queries. Whether the server holds the model weights encrypted or in plaintext depends on the trust relationship with the model owner; our solution is orthogonal to this choice, affecting only latency. We refer the reader to (Adir et al., 2024)[Chapter 3] for a formal treatment of this security model.

Model	#Heads	#Tokens	#Layers	Taylor and Goldschmidt (i)	Square strategy (ii)	PowerSoftmax
Pythia 70M	8	128 / 1024	6	3.6 / 83.9	7.42 / 173.4	2.7 / 8.6
Pythia 1.4B	16	128 / 1024	24	19.4 / 662.3	42.1 / 1371.6	11.1 / 51.8
Llama 7B	32	128	32	46.1	84.05	16.6

Table 5: Comparison of computation cost (in seconds) for three different polynomial alternatives to Softmax, lower is better.

I MASKED HE-FRIENDLY VARIANT

Attention masks are a well-known technique used to manipulate self-attention by determining which tokens can attend to each other. Traditional LLMs leverage a binary mask M for various applications. A notable example is the causal mask, employed for training LLMs via NTP, a popular self-supervised learning scheme. These standard masking mechanisms are specifically designed for Softmax-based self-attention (masked values were represented by $-\infty$ and used as an additive term) and should be reformulated for HE-Friendly Attention. Masking is applied via an element-wise product, denoted by $\odot$, as follows:

$$\text{Masked HE-Friendly Attn}(Q,K,V) = \left(\frac{QK^T \odot M}{\sqrt{d_k}}\right), \quad M_{i,j} \in [0,1]. \tag{11}$$

J ADDITIONAL POLYNOMIAL ATTENTION VISUALIZATION

In Fig. 13 and Fig. 14, we present a visual analysis of attention matrices obtained from both the vanilla Softmax-based models and the corresponding polynomial HE-friendly variants across different layers. Fig. 13 depicts the attention matrices averaged over 3 seeds, all attention heads at a layer, and 1,000 examples. Additionally, to provide a comprehensive view of the attention matrices, Fig. 14 contains random samples of attention matrices. All models rely on a BERT-like 12-layer causal model with a context length of 512, trained on Wikitext-103 for next-token prediction with the same training procedure. We use examples from the test set of Wikitext-103 as input samples.

Our analysis reveals that as p increases, the resulting attention matrices become more localized as depicted in Fig. 13. For instance, by comparing the first column (PowerSoftmax with $p = 4$) with the third column ($p = 12$), we observe a significantly stronger diagonal in the latter, whereas the $p = 4$ model displays a more uniform attention distribution. Additionally, we empirically confirm this pattern by analyzing the average of the mean attention distance (Vig and Belinkov, 2019) per model (i.e., averaged across all the layers and heads) as illustrated in Fig. 12.

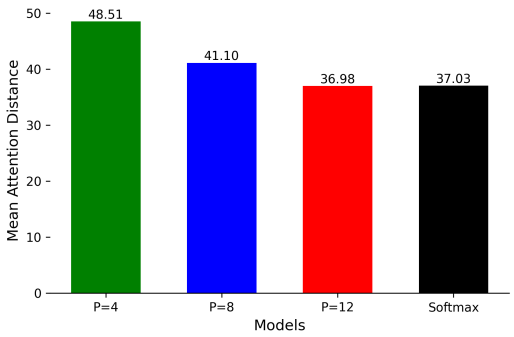

Figure 12: Attention mean distance for different transformer variants.

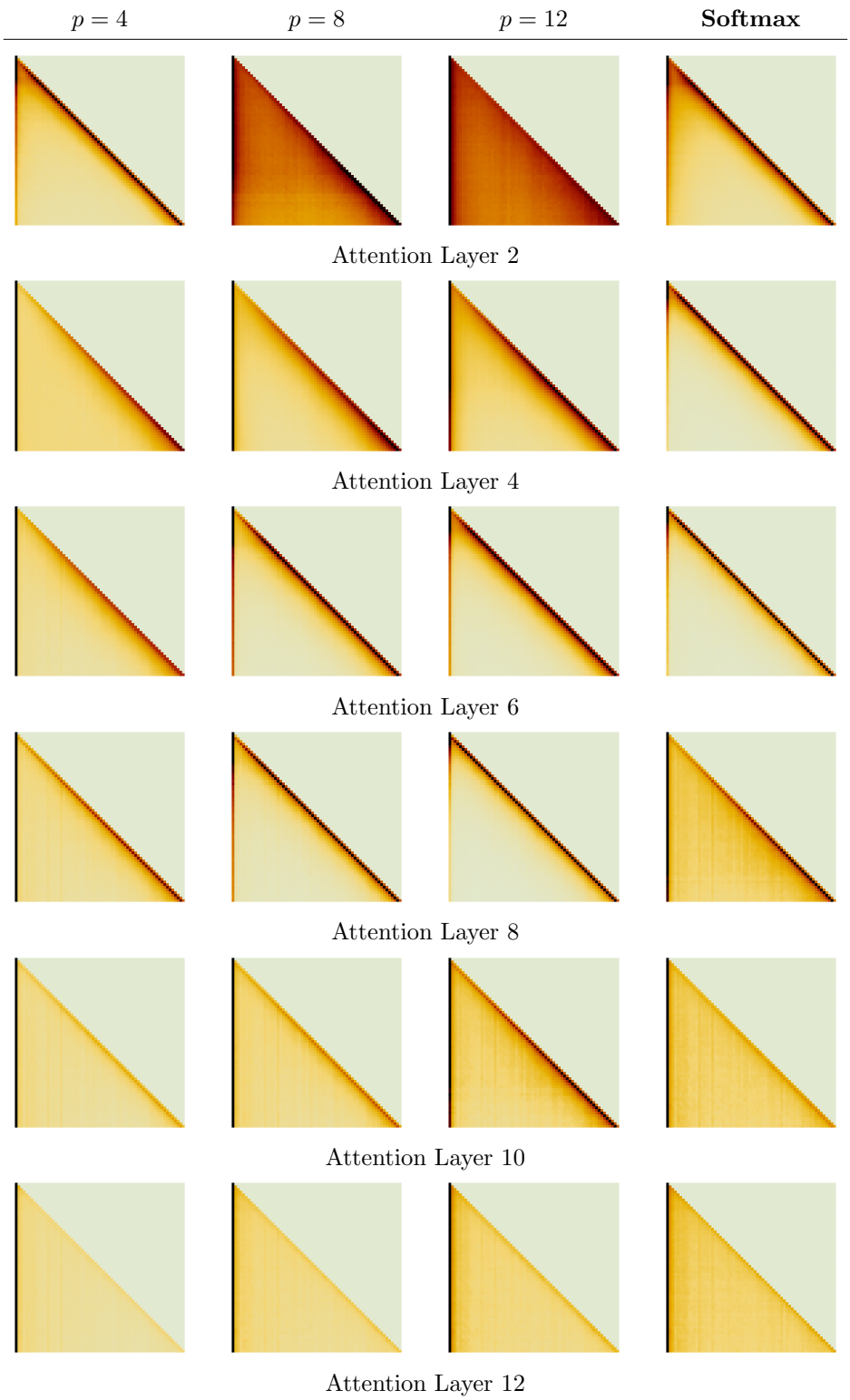

Figure 13: **Visualisation of polynomial average attention matrices:** Models with $P = 4$ (first column) generate more local attention matrices, with reduced mass near the diagonal compared to models with $P = 8$ or $P = 12$, particularly in layers 4-10. In all models, the final layers (rows at the bottom) display more global attention patterns than the middle layers.

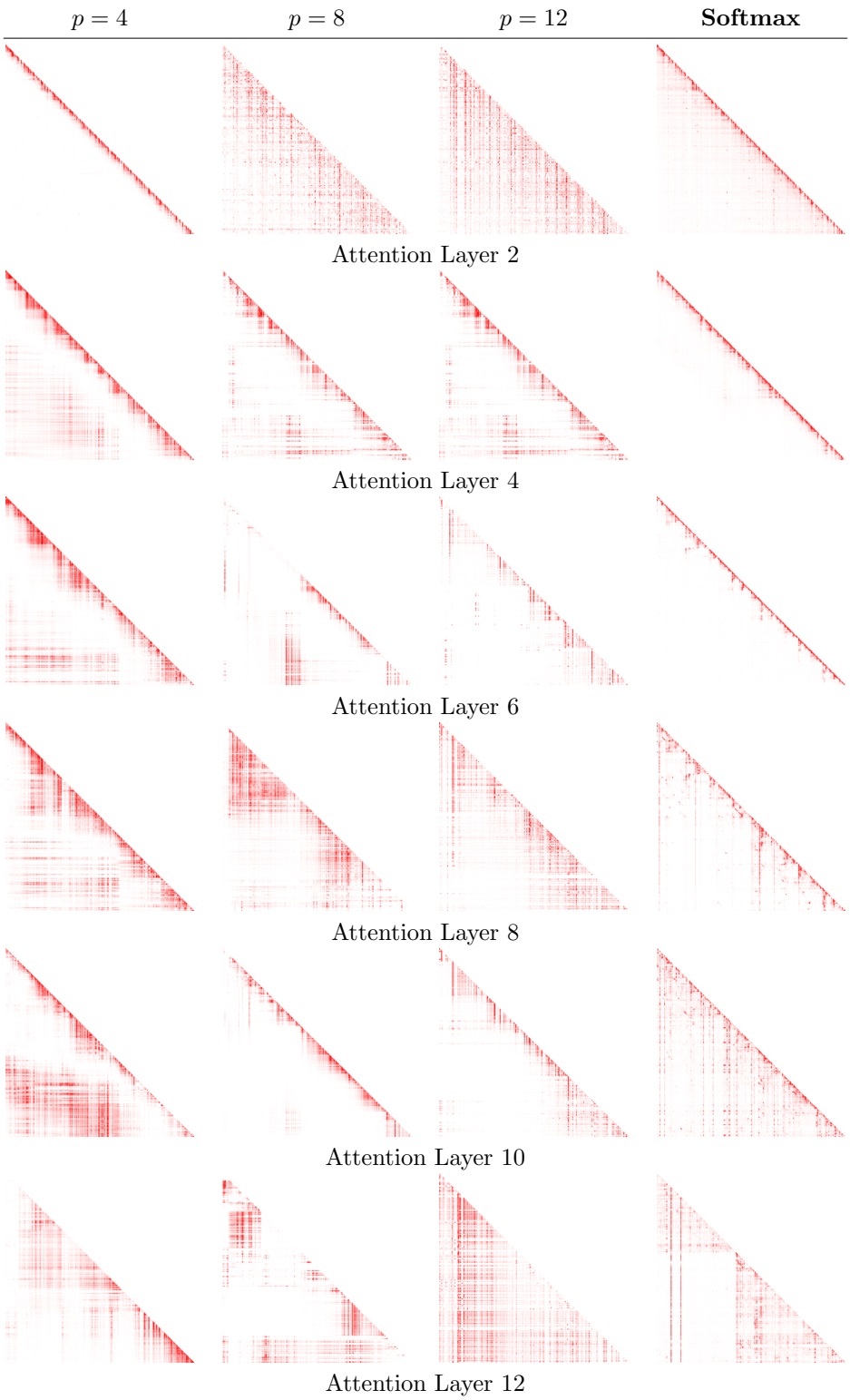

Figure 14: **Visualisation of random samples of polynomial attention matrices:** Although the attention matrices are noisy and a small number of samples may not capture the full distribution trend, the PowerSoftmax-based models (first three columns) show behavior similar to the original Softmax (last column). Notably, our attention layers can dynamically adjust focus across different parts of the input, allowing attention heads to freely learn both local and global patterns.