# Converting Transformers to Polynomial Form for Secure Inference Over Homomorphic Encryption

**Itamar Zimerman** [1 2] **Moran Baruch** [1 3] **Nir Drucker** [1] **Gilad Ezov** [1] **Omri Soceanu** [1] **Lior Wolf** [2]

## Abstract

Designing privacy-preserving DL solutions is a major challenge within the AI community. Homomorphic Encryption (HE) has emerged as one of the most promising approaches in this realm, enabling the decoupling of knowledge between a model owner and a data owner. Despite extensive research and application of this technology, primarily in CNNs, applying HE on transformer models has been challenging because of the difficulties in converting these models into a polynomial form. We break new ground by introducing the first polynomial transformer, providing the first demonstration of secure inference over HE with full transformers. This includes a transformer architecture tailored for HE, alongside a novel method for converting operators to their polynomial equivalent. This innovation enables us to perform secure inference on LMs and ViTs with several datasts and tasks. Our techniques yield results comparable to traditional models, bridging the performance gap with transformers of similar scale and underscoring the viability of HE for state-of-the-art applications. Finally, we assess the stability of our models and conduct a series of ablations to quantify the contribution of each model component. Our code is publicly available.

https://github.com/IBM/PolyTransformer

## 1. Introduction

Privacy-Preserving Machine Learning (PPML) has become a prominent field, ensuring that valuable insights can be inferred from data without compromising individual privacy. HE stands out within this domain, offering the capability to compute on encrypted data in a **non-interactive** manner, thereby safeguarding sensitive information during local analysis. However, modern HE schemes such as CKKS (Cheon et al., 2017) support computation over encrypted input only when the computations are represented by polynomial functions. This limitation poses a unique challenge for DL applications. For example, GELU and Softmax are non-polynomial and, therefore, must be adapted into an equivalent polynomial form. Understanding the theoretical limitation is important when considering applications of HE in real-world scenarios.

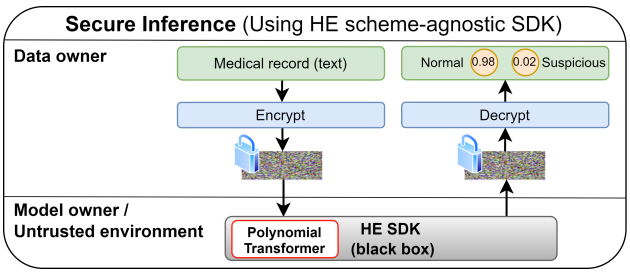


Figure 1: **Schematic Overview of Secure Inference Using HE**: The diagram depicts the sequence of steps where the data owner sends an encrypted sensitive data sample, to the model owner or an untrusted environment. Within this environment, an HE framework (SDK) employs the polynomial transformer to process encrypted data, ensuring no access to the sensitive information, doing so in a **non-interactive** way. After computation, the encrypted result, is returned to the data owner, who decrypts it to retrieve the final classification outcomes. As such, the privacy of the underlying data is maintained. As such, the privacy of the underlying data is maintained throughout the entire process.

Fig. 1 depict a typical HE scenario: A model owner provides a trained model. A data owner encrypts its data using HE and sends it to the model owner for inference. The model owner processes the encrypted data, where the input and output remain confidential and cannot be learned by the model owner. The encrypted result is then returned to the data owner for decryption and interpretation.

As of now, the adaptation process to align NNs with the HE requirements has been primarily focused on polyno-

[1]IBM Research, Israel [2]Tel-Aviv University [3]Bar-Ilan University. Correspondence to: Itamar Zimerman <Itamar.Zimerman@ibm.com>.

*Proceedings of the 41st International Conference on Machine Learning*, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

mial CNNs such as AlexNet (Krizhevsky et al., 2012), ResNet (He et al., 2016), and ConvNext (Liu et al., 2022). This focus is evident in various previous works, e.g. (Baruch et al., 2023; Aharoni et al., 2023; Lee et al., 2022b; Baruch et al., 2022; Dathathri et al., 2019; Gilad Bachrach et al., 2016). These models commonly use $\mathrm{ReLU}$ and $\mathrm{GELU}$, which are relatively straightforward for polynomial approximation. However, transformers (Vaswani et al., 2017) remain a notable exception due to their inherent non-polynomial operations such as Softmax – which, unlike $\mathrm{ReLU}$, requires division by an exponent and presents a more challenging case for polynomial approximation. Another limitation is their large structural size, for example, one study (Zhou et al., 2019) showed that polynomial NNs tend to be less stable with increasing capacity.

**Our contributions.**

1. We introduce a novel approach to adapt transformers to be compatible with modern HE schemes for secure inference. Our method involves a series of simplifications and approximations, that allows us to introduce the **first practical polynomial transformer models** that retain competitive performance across both language modeling and image classification.
2. We demonstrate the feasibility of employing transformers under HE while bridging the performance gap with their non-encrypted counterparts of similar scale.
3. We offer stability analyses and ablation studies for a comprehensive understanding of the trade-offs and potential of polynomial transformers in privacy-preserving settings, paving the way for future innovations in PPML.
4. We provide techniques for polynomial and HE-friendly alternatives to the layer normalization and self-attention layers. Such techniques can enhance existing cryptographic protocols including improving client-aided solutions, or enable the development of a wider variety of polynomial models, extending beyond transformers.

## 2. Background

**Homomorphic Encryption (HE)** is a cryptographic technique that allows computations to be performed on encrypted data without decryption (Gentry, 2009) The HE system has an encryption operation $\mathrm{Enc} : \mathcal{R}_1 \to \mathcal{R}_2$ that encrypts plaintext input from the ring $\mathcal{R}_1(+, *)$ into ciphertexts in the ring $\mathcal{R}_2(\oplus, \odot)$ and an associated decryption operation $\mathrm{Dec} : \mathcal{R}_2 \to \mathcal{R}_1$. An HE scheme is correct if for every valid input $x, y \in \mathcal{R}_1$:

$$\mathrm{Dec}(\mathrm{Enc}(x)) = x \tag{1}$$

$$\mathrm{Dec}(\mathrm{Enc}(x) \oplus \mathrm{Enc}(y)) = x + y \tag{2}$$

$$\mathrm{Dec}(\mathrm{Enc}(x) \odot \mathrm{Enc}(y)) = x * y \tag{3}$$

It is approximately correct if for some small $\epsilon > 0$ determined by the key, $|x - \mathrm{Dec}(\mathrm{Enc}(x))| \leq \epsilon$, and similarly modifying Equations 2 and 3.

**Polynomial DL Models** Producing polynomial networks with high accuracy is challenging, and several theoretical intuitions and proofs were proposed. For example, Zhou et al. (2019) proved that under some conditions polynomial FFNs are unstable, concluding that the likelihood of instability in a polynomial network increases with its complexity, specifically as depth and width grow. Goyal et al. (2020) suggested that the problem with poly-activations is that the gradients and outputs are unbounded and can be arbitrarily large, unlike other activations such as ReLU and GELU. They also pointed out that in deeper networks $f_{(d,l)}$ with $l$ layers and $d$-degree polynomial activations, the gradients explode exponentially in the degree of the entire network, since for input $x > 1, \lim_{x\to\infty} f_{(n.l)}(x)/x = \infty$. Additionally, Chrysos et al. (2020); Goyal et al. (2020); Gottemukkula (2020) attempted to implement deep polynomial networks but faced optimization instability. They resolved the issue by incorporating non-polynomial components into their models.

Recent works focus on converting Deep CNNs into polynomials. The method in (Baruch et al., 2023) introduced a technique to stabilize polynomial models by adding a loss term that minimizes the input range to the non-polynomial layers. Using this approach, they successfully produced low-degree polynomial versions of ResNet-152 and ConvNeXt on ImageNet. In addition, (Lee et al., 2021) approximated $\mathrm{ReLU}$ using a composition of three polynomials to precisely approximate $\mathrm{ReLU}$, achieving high-degree polynomial models. To the best of our knowledge, these works represent some of the deepest polynomial models to date. However, none of these or any other works have tackled the problem of polynomial transformers.

**Polynomial Approximation** Polynomial networks are commonly obtained by approximating the non-polynomial functions of pre-trained networks, e.g., (Lee et al., 2022d; Takabi et al., 2019; Mohassel & Zhang, 2017; Hesamifard et al., 2017; Lee et al., 2021), or by substituting ReLU during or after a dedicated training process (Baruch et al., 2022; 2023). The Remez algorithm (Remez, 1934; Pachón & Trefethen, 2009; Egidi et al., 2020) is commonly used for finding the optimal polynomial approximation of a function $f(x)$ in a certain degree within a pre-defined range $[a, b]$, assuming a uniform distribution of $x$. Alternatively, iterative methods such as the Newton–Raphson method (Raphson, 1702) offer another polynomial approximation approach. Specifically, (Panda, 2022) focused on approximating $\frac{1}{\sqrt{x}}$ in the interval $[a, b]$, by dividing the interval into sub-intervals and approximating over each via Newton–Raphson method, before aggregating the results with

another polynomial. However, the input range to the non-polynomial layers $[a, b]$ can be extremely large, which results in poor and no practical approximations. This paper employs polynomials for $\mathrm{ReLU}$, as defined in (Lee et al., 2021), for layer normalization (inverse square root) from (Panda, 2022) and for $\mathrm{GELU}$ using polynomials derived from the Remez algorithm.

**Transformers in PPML** The integration of transformers into PPML solutions has become significantly prominent, highlighting the relevance of both fields. In recent years, a variety of secure interactive protocols have been introduced to enable secure inference of transformer models (Chen et al., 2022; Ding et al., 2023; Liu & Liu, 2023; Liang et al., 2023; Gupta et al., 2023; Zheng et al., 2023; Zeng et al., 2023; Hao et al., 2022). These methods leverage mechanisms such as shared secrets to compute non-polynomial operations like $\mathrm{GELU}$, $\mathrm{Softmax}$, and $\mathrm{LayerNorm}$. However, such approaches increase communication overhead and the potential for vulnerability to man-in-the-middle attacks. Our method addresses these concerns by enabling computation in untrusted environments, eliminating the need for additional communication, and thus preserving a non-interactive stance. By offering polynomial alternatives for non-polynomial operations in transformers, our method not only enhances existing protocols, but also eliminates the need for client involvement in the computation process of non-polynomial operations. One alternative approach involves applying secure inference on text embeddings extracted from an unsecured transformer via a HE-based classification model (Lee et al., 2022c; 2023). However, this method addresses a significantly narrower threat model.

## 3. Problem Settings

We begin by clearly defining the motivation and problem settings before discussing our methodology. Our objective is to develop a transformer model that uses only polynomial operations and performs well on downstream tasks. By utilizing these polynomial-based transformers, we aim to enable secure inference within the HE framework. Note that this paper does not cover secure training. One might think that replacing non-polynomial operations in the network with polynomial alternatives or approximated polynomials, either before or after training, could simply solve the problem. However, polynomial networks are unstable, and instability issues can arise during training or when non-polynomial operations are replaced, especially in deep networks (as analyzed from both theoretical and empirical perspectives in (Zhou et al., 2019; Goyal et al., 2020)). Therefore, standard methods for creating polynomial DNNs for secure inference involve several architectural modifications and unique training procedures, including training of non-polynomial DNNs as an intermediate step (Baruch et al., 2023; 2022; Ao & Boddeti, 2023). Furthermore, in general, as higher-degree polynomials are used, the running time for secure inference increases drastically. Hence, a common challenge is to reduce both the polynomial degree of each operation and the overall multiplication depth of the model.

## 4. Method

Addressing the problem defined in Section 3, our methodology begins by identifying non-polynomial components in the transformer model: (i) The $\mathrm{Softmax}$ function in the attention is non-polynomial, involving exponentiation and division; (ii) layer normalization ($\mathrm{LayerNorm}$), which normalizes features by dividing them by their standard deviation, also contains the square root function, adding to the non-polynomial complexity; and (iii) activation functions, which are traditionally non-polynomial but have been substituted with polynomial alternatives in prior research (Lee et al., 2021; Baruch et al., 2023; Lee et al., 2022a).

From earlier stages of our research, we found that directly approximating the $\mathrm{Softmax}$ and $\mathrm{LayerNorm}$ by polynomials within each transformer block is challenging due to: (a) the inherent complexity of these functions, which involve multiple non-polynomial operations—specifically, division and exponentiation for $\mathrm{Softmax}$, and division along with square root for $\mathrm{LayerNorm}$, and (b) the nature of these functions as vector-based rather than scalar-based, unlike neural activation functions, which were approximated by polynomials previously. In light of these complexities, our work seeks to develop HE-friendly alternatives to $\mathrm{Softmax}$-based attention (in Section 4.1) and $\mathrm{LayerNorm}$ (in Section 4.2) that are easier to approximate by a polynomial, ideally employing operations that are polynomial in nature, as well as those that can be precisely approximated by polynomials. Our entire pipeline for polynomial adaptation is detailed in Section 4.3.

### 4.1. HE-Friendly Attention

To circumvent the use of $\mathrm{Softmax}$, we employ a pointwise activation-based attention, denoted as $\mathrm{Attn}_\sigma$ as an alternative. This $\mathrm{Softmax}$-free mechanism can be formalized as:

$$\mathrm{Attn}_\sigma(Q, K, V) = \sigma\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{4}$$

where $\sigma$ acts as an element-wise activation function, differing from the vector-wise $\mathrm{Softmax}$ function used in standard attention mechanisms. This modification simplifies the transformation into a polynomial form, effectively reducing the problem from polynomial attention to the well-studied problem of handling polynomial activation functions. This mechanism enhances model accuracy (Ma et al., 2022) and reduces latency (Hua et al., 2022; So et al., 2021), which are

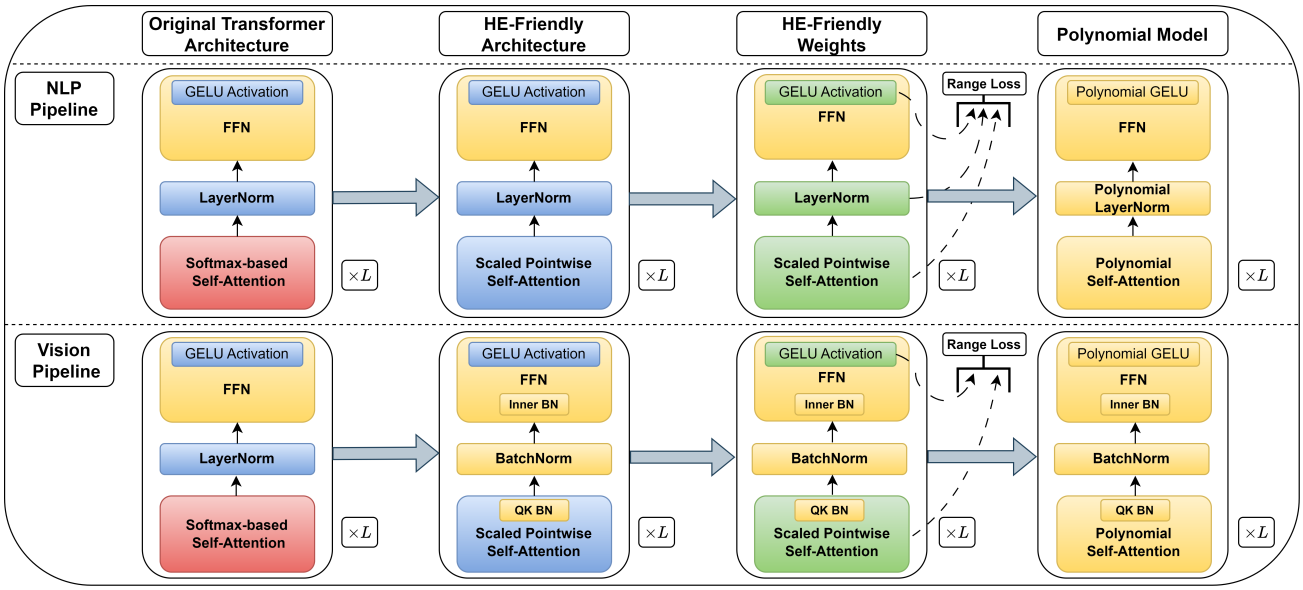


Figure 2: **Method:** The NLP pipeline is depicted at the top, and the vision pipeline shown at the button. **Yellow** marks polynomial components, while **red** indicates components that cannot be efficiently adapted to polynomial form, and therefore replaced by components that can be replaced by a polynomial via range-minimization tuning (**blue**). **Green** components can be easily replaced by polynomials, given their current weights. Only the two central columns include training, while the rest include component replacement. Skip-connections are omitted for clarity.

benefits outside the scope of HE. However, those implementations have combined it with additional techniques, such as gated attention and extra unique normalization. These additions result in an increase in the overall multiplication depth and require the development of additional HE-friendly alternatives. In our work, we omit these supplementary components altogether. In their absence, the standalone mechanism becomes less stable, and we use length scaling to overcome this instability:

$$\text{Attn}_{\sigma\text{-scale}}(Q,K,V) = \frac{1}{\text{S}(L)}\sigma\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{5}$$

where $L$ represents the sequence length, and $S$ is a scaling function defined as either $\frac{1}{\sqrt{L}}$ or $\frac{1}{L}$. Thus, the stabilizing effect of the $\mathrm{Softmax}$ is replaced by a scaling factor that compensates for the instability introduced by the alternative attention mechanism. Additionally, as the multiplication of the attention matrix with the values matrix involves summation over $L$ elements, it is logical to normalize by this factor. We investigate various scaling functions, applied both pre- and post-activation, and can be implemented by modifying the $\sigma$ activation function:

$$\text{Pre-act scaling:} \quad \hat{\sigma}(x) = \sigma(S(L)x) \tag{6}$$

$$\text{Post-act scaling:} \quad \hat{\sigma}(x) = S(L)\sigma(x) \tag{7}$$

$$\text{Pre- and post-scaling:} \quad \hat{\sigma}(x) = S(L)\sigma(S(L)x) \tag{8}$$

The selection of scaling functions is determined by empirical considerations and stability analysis.

**Choosing the Attention Activation** Naturally, we initially experimented with polynomials as activations. We started with polynomials used in previous HE-literature, ranging from a simple quadratic activation ($x^2$) (Gilad Bachrach et al., 2016) to high-degree polynomials (Lee et al., 2021) that approximate $\mathrm{ReLU}$ and other standard activations. However, these polynomial activations were found to be unstable during training. To address this, we first trained our model using standard non-polynomial activations and then applied an additional training phase to convert these activations into polynomials, similar to (Baruch et al., 2023). This approach balanced performance in the initial training phase with the precision of polynomial approximation in the later phase.

**Reformulate Attention Mask** Traditional practices, such as those employed in the Swin transformer or in training LLMs via self-supervised learning, manipulate self-attention via masking to determine which tokens can attend to each other. These standard mask mechanisms are specifically designed for the Softmax-based self-attention and should be reformulated for pointwise attention:

$$\text{Attn}_{\sigma}(Q,K,V) = \left(\sigma\left(\frac{QK^T}{\sqrt{d_k}}\right)\odot M\right)V \tag{9}$$

where $M$ is the binary mask. This mechanism is agnostic to any type of pointwise activation $\sigma$.

### 4.2. HE-Friendly Normalization

To enhance training stability, transformers rely on $\mathrm{LayerNorm}$, which is formulated as follows:

$$\mathrm{LayerNorm}(x) = \frac{x-\mu}{\sqrt{\sigma^2}} \cdot \gamma + \beta \tag{10}$$

where $x$ is the input vector, $\mu$ is the mean of $x$, $\sigma^2$ is the variance, and $\gamma$ and $\beta$ are learnable parameters. Computing $\mathrm{LayerNorm}$ over HE requires calculating the inverse square root, which is not a polynomial operation. A common practice in designing DNNs for secure inference over HE is to replace $\mathrm{LayerNorm}$ with $\mathrm{BatchNorm}$, as it can be implemented by a straightforward constant affine transformation at inference time. Therefore, we attempted to train transformers using $\sigma$-attention and $\mathrm{BatchNorm}$. We observed that these models were highly unstable, performing poorly on vision tasks and failing to converge in NLP tasks. Consequently, we adopt two distinct approaches for vision and NLP tasks.

**Normalization for Vision Transformers** For ViTs, to improve performance and mitigate training instability, we add two components: (i) Additional $\mathrm{BatchNorm}$ in the MLP, which is proposed in (Yao et al., 2021) as a stabilizer for ViT training, and (ii) additional $\mathrm{BatchNorm}$ within the $\sigma$-attention, which normalizes values across different attention heads, since we observe that those are the sources of instability. The resulting $\sigma$-attention variant is:

$$\frac{1}{\mathrm{S}(L)}\sigma\left(\text{BatchNorm2D}\left(\frac{QK^T}{\sqrt{d_k}}\right)\right)V \tag{11}$$

**Normalization for NLP Transformers** For NLP, models with $\sigma$-attention and $\mathrm{BatchNorm}$ completely failed, even when augmented by stabilizing factors from the literature, such as (Wang et al., 2022). Consequently, we had to confront the challenge of approximating $\mathrm{LayerNorm}$ by polynomials, which entails approximating the inverse square root function. Empirically, we found that the values of the variance in trained transformers (with $\sigma$-attention) ranged between 1 and $10^9$, causing approximation challenges due to the extremely large domain. To solve this problem, we first focus on narrowing the domain of the variance, which then makes it easier to approximate the inverse square root over this restricted domain. The method is similar to (Baruch et al., 2023), which introduces an additional loss function that encourages the model to minimize the range of the input to activation layers. We apply this technique on the variance at each layer via the following objective:

$$\mathbb{L}_{\text{Var-Minimization}} := \sum_{n=1}^{N_L} \max_{c\in C, x_i \in X} \left\{\mathrm{var}^i_{n,c}\right\} \tag{12}$$

where we denote the number of layers by $N_L$, the set of channels by $C$, and the train dataset by $X := [x_1, x_2, ..]$. Furthermore, we denote the variance at layer $n$ and channel $c \in C$, when the model processes the $x_i$ example by $\mathrm{var}^i_{n,c}$. For reasons of efficiency, we compute the loss over each batch rather than the whole training set $X$.

By extending this method to operate on layer normalization (LayerNorm) instead of activations, we succeed in reducing the variance range to a smaller domain. This reduction makes it feasible to use well-known approximations, such as the technique described in (Baruch et al., 2023), for the inverse square root.

### 4.3. A Recipe for a Polynomial Transformer

Figure 2 illustrates the entire method, which comprises three stages: (i) First, we modify the architecture from the original transformer architecture (first column) to a **HE-friendly architecture** (second column), namely, an architecture that can eventually be converted into a polynomial form. Then we train the modified model from scratch with the same hyperparameters. (ii) In the second stage, we perform a supplementary training procedure to obtain a model with **HE-friendly weights**, which means that each non-polynomial component will only operate on specific and restricted domains. To do so, we add a loss function that minimizes the range of inputs to non-polynomial layers. For the activations (standard activations and attention-activations), we directly apply the method from (Baruch et al., 2023), which defines the range loss for activations. For the $\mathrm{LayerNorm}$ layers, we use the loss defined in Eq. 12. The whole training objective $\mathbb{L}$ is defined by:

$$\alpha\mathbb{L}_{\text{Range Minimization}} + \beta\mathbb{L}_{\text{Var-Minimization}} + \mathbb{L}_{\text{original}} \tag{13}$$

where $\alpha$ and $\beta$ are hyperparameters. (iii) Finally, each non-polynomial layer is directly replaced with its polynomial approximation, resulting in a **polynomial model**. Appendix A contains details on the approximation we used. Those approximations are accurate for the HE-friendly architecture & weights obtained from earlier stages.

## 5. Experiments

We evaluate the polynomial models generated by our method in Section 5.1, focusing on language modeling with several benchmarks such as the Wikitext-103 dataset and image classification using standard benchmarks, including Tiny-ImageNet and CIFAR-10. Section 5.2 justifies our methodological choices, specifically the use of scaled $\sigma$-attention and an additional training phase designed to manipulate the input values of non-polynomial layers. Furthermore, that section contains several ablation studies to assess the impact of each method component on the overall performance degradation. Finally, we discusses the accuracy and latency implications of applying our models over HE. The experimental setup is detailed in Appendix B.

### 5.1. Polynomial Models

**Polynomial Language Modeling** We evaluated our BERT-like transformer model for language modeling as our NLP task. Specifically, we trained on Wikitext-103, text-8, and enwik8 with a self-supervised scheme for Next Token Prediction (NTP). The results in Table 1 show that after architectural and training modifications, we achieved a polynomial model with competitive perplexity. In particular, for the wikitext-103 benchmark, the perplexity increased by 0.91 compared to a vanilla transformer of the same size, from 18.98 to 19.89 for a 6-layer transformer (53.3M parameters), and by 2.02 from 16.89 to 18.91 for a 12-layer model (95.8M parameters). Considering that at least 80% of the gap between the vanilla transformer and our corresponding polynomial model is caused in the last stage where polynomial approximations are used (0.74 for 6 layers model and 1.76 for 12 layers), we hypothesize that more accurate polynomials can mitigate most of the performance gap.

| Dataset | Depth | Original | P | P+RM | Poly |
|---|---|---|---|---|---|
| wiki103 | 6 | 18.98 | 19.07 | 19.15 | 19.89 |
| wiki103 | 12 | 16.89 | 16.98 | 17.15 | 18.91 |
| text-8 | 6 | 2.416 | 2.419 | 2.433 | 2.435 |
| text-8 | 12 | 2.395 | 2.400 | 2.404 | 2.421 |
| enwik8 | 6 | 2.330 | 2.349 | 2.350 | 2.367 |
| enwik8 | 12 | 2.211 | 2.226 | 2.234 | 2.281 |

Table 1: **NLP Results:** Perplexity results of a polynomial BERT-like transformer on the Wikitext-103, text8, and enwik8 benchmarks. 'Depth' indicates the number of transformer layers. 'Original' denotes the perplexity of the vanilla Softmax-based transformer of equivalent size. 'P' represents models utilizing scaled $\sigma$-attention, while 'P+RM' shows perplexity at the end of the range minimization training. 'Poly' details the final performance after substituting $\mathrm{LayerNorm}$ and activation functions with polynomial approximations.

**Polynomial Image Classification** We evaluated our vision models on two image classification benchmarks: Tiny-ImageNet and CIFAR-100. The results, presented in Table 2, indicate that our vision models, which are converted to polynomial form by our methods, remain competitive. Specifically, for ViT on CIFAR-100, the original ViT (denoted as 'O') achieved a score of 73.4%, whereas our HE-friendly alternative (P+BN+QK+A), achieved a score of 71.1%. The HE-friendly alternative employs $\mathrm{BatchNorm}$ as the normalization layer, includes additional stabilizers described in 4.2, and is based on scaled-$\sigma$ attention. After applying our range-aware training procedure, the accuracy of our model decreased marginally by 0.1% to 71.0%, and it further decreased to 70.8% after approximating non-polynomial components. For the Swin Transformer on Tiny-ImageNet, the original model achieved 59.4%, and transitioning to the HE-friendly architecture resulted in a performance decrease of 0.3% to 59.1%. After employing range-aware training to obtain HE-friendly weights, the performance further degraded by 0.2% to 58.9%, while the final performance of the polynomial model remained the same. In conclusion, the performance gap between the polynomial models and the original architectures is less than 4%, demonstrating the practicality of our methods in this domain.

| Model | Dataset | O | P+BN+QK+A | RM | Poly |
|---|---|---|---|---|---|
| ViT | CIFAR-100 | 73.4 | 71.1 | 71.0 | 70.8 |
| Swin | Tiny-ImgNet | 59.4 | 59.1 | 58.9 | 58.9 |

Table 2: **Vision Results:** Test accuracy results of a polynomial ViT. 'O' represents the original vanilla model, 'P+BN+QK+A' represents scaled-$\sigma$ attention-based ViT trained with $\mathrm{BatchNorm}$ as the normalization function instead of $\mathrm{LayerNorm}$, and contains the additional stabilizers described in 4.2. 'RM' refers to the accuracy at the end of the range minimization training, and 'Poly' details the final performance after substituting polynomial approximations.

### 5.2. Model Analysis

**Scaled-$\sigma$ attention** We began our analysis by empirically assessing the performance differences between the vanilla transformer and the scaled-$\sigma$ attention. We conducted experiments in five regimes: language modeling on wikitext-103 for (i) next token prediction (NTP) and (ii) denoising (MLM), (iii) language modeling on the text-8 dataset for NTP, and two additional image classification tasks using (iv) CIFAR-10 with the ViT backbone and (v) Tiny-ImageNet with the Swin backbone. Across all regimes, we employed the same hyperparameters that were optimized for the vanilla transformer, which can be found in Table 8 and Table 7 (see Appendix B). The training curves are presented in Figure 3, and indicate that the models with scaled-$\sigma$ attention perform comparably to the baseline.

To justify our design choices regarding length scaling, we performed analyses in both the NLP and vision domains, comparing several variants of scaled-$\sigma$ attention models. These variants included models without length scaling and models with length scaling, employing the two functions $S(L) = \frac{1}{\sqrt{(L)}}$ and $S(L) = \frac{1}{L}$, applied at different positions relative to the activation — either before, after, or both.

For the NLP tasks, consistent with our previous experiments, we employed a 6-layer BERT-like transformer as a baseline and evaluated the variants on the Wikitext-103 dataset. We experimented with two types of activation functions: GELU

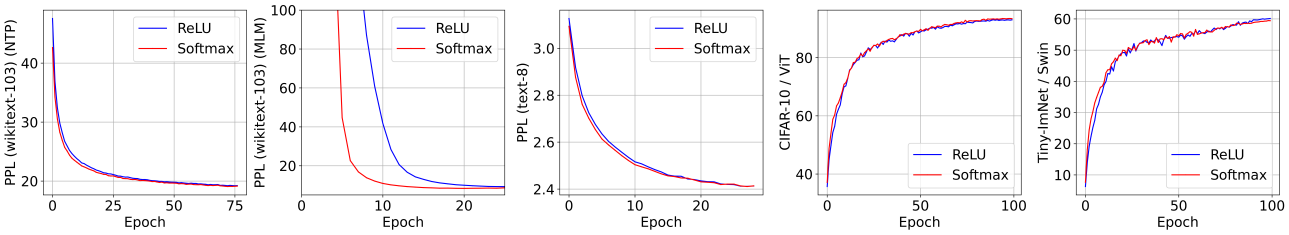


Figure 3: Scaled-$\sigma$ attention is comparable with Softmax attention. On each graph, we present the test accuracy for vision tasks, or perplexity for language modeling tasks.

and Squared ReLU. Notably, without length scaling, we observed that the model's weights explode in the initial training epochs. This phenomenon confirms the necessity of integrating length scaling with $\sigma$-attention. Furthermore, variants utilizing both pre-scaling and post-scaling with a scaling function of $S(L) = \frac{1}{L}$ failed to converge and eventually collapsed. For the other scaling functions, the results are depicted in Figure 4. Evidently, employing post-length scaling with the scaling function $S(L) = \frac{1}{\sqrt{L}}$ provides the best performance for both types of attention activation.

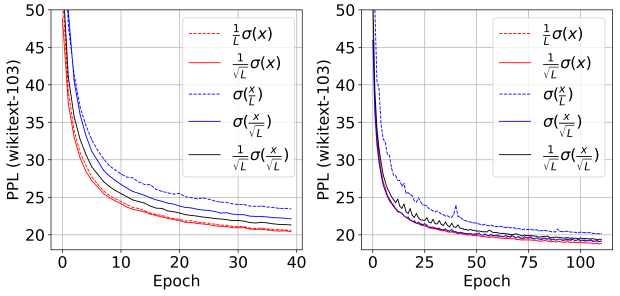


Figure 4: **Pointwise Transformers Require Scaling in NLP.** (left) GELU attention. (right) Squared ReLU attention. All experiments without scaling, or with both pre-scale and post-scale by length ($\frac{1}{L}\sigma(\frac{x}{L})$) collapse early in the training, even for lower learning rate.

For vision tasks, we replicated the settings detailed in Section 5.1 with the ViT backbone. Test accuracy results with and without length scaling for attention activations are presented in Table 3 for models evaluated on both the CIFAR-10 and CIFAR-100 benchmarks. These experiments were conducted with BatchNorm, as this was the type of normalization we use for secure ViT (see Section 4.2). We employ post-activation length scaling with a scaling function $S(L) = \frac{1}{\sqrt{L}}$, which we found to be optimal for scaled-$\sigma$ based ViT. The findings underscore the significance of length scaling, which substantially enhances performance. Specifically, length scaling improves the GELU-attention models, increasing their accuracy by 2.46% from 68.41% to 70.87% on CIFAR-100, and lifting their performance by 11.14% from 81.17% to 92.31% on CIFAR-10.

| Dataset | Vanilla | GELU | Scaled-GELU |
|---|---|---|---|
| CIFAR-100 | 72.08 | 68.41 | 70.87 |
| CIFAR-10 | 92.70 | 81.17 | 92.31 |

Table 3: **Pointwise ViT Require Scaling:** Experiments on both CIFAR-100 and CIFAR-10 with ViT. For both benchmarks, we compare vanilla attention, GELU-attention, and scaled-GELU attention.

| $\sigma$ | O | B | B+QK | B+QK+A | B+QK+A+S |
|---|---|---|---|---|---|
| G. | 59.4 | 39.0 | 49.8 | 58.6 | 59.1 |
| R. | 59.4 | 38.4 | 50.1 | 56.2 | 58.6 |

Table 4: **Stabilize BatchNorm-based $\sigma$-ViT:** Experiments on both Tiny-ImageNet with $\sigma$-attention-based Swin transformer. We compare models with GELU-attention (G.) and ReLU-attention (R.). 'O' denotes the accuracy of the original Swin with Softmax and LayerNorm. 'B' denote the accuracy of a $\sigma$-attention-based Swin with BatchNorm. The remaining three columns represent the accuracy achieved by incorporating the first technique, the first two techniques combined, and all of the techniques, respectively.

**Range Minimization** Our novel training method narrows the variance range at each LayerNorm layer and the input to the activation layers, including both attention activations and MLP activations. To demonstrate the practicality and effectiveness of this method, we visualize in Figure 5 the maximal and mean-variance values at each layer, as well as the maximal and minimal values at the input of each activation. Both are measured at the end of each epoch to demonstrate progress during training.

The graph clearly illustrates that prior to implementing the additional training phase, variance values were limited by 3300. This number was reduced to 300 during range training. Furthermore, it is evident that the activation range has

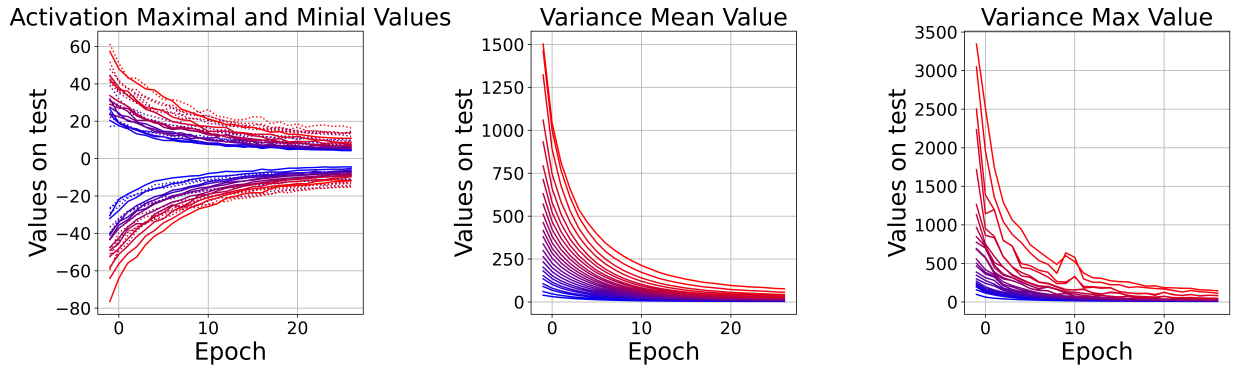


Figure 5: **Range Minimization:** The impact of applying range regularization loss (Eq. 13) to constrain the input range of non-polynomial layers. Each curve corresponds to an individual non-polynomial layer. The **left** panel displays maximal and minimal values recorded at the input of the activation layers throughout the test set for each layer, where dashed lines denote activations in the MLP and solid lines represent attention activations. The **middle and right** panels, respectively display the mean and maximum values of the variance observed throughout the test set for each layer. The colors of the curves progress from blue to red to denote the sequence of layers: blue for the first layer, and red for the last, where intermediate layers are colored by interpolating blue and red based on their sequence position. The x-axes represent the epochs.

been shortened from 70 to 20. These reductions greatly facilitate the approximation problem, since the error of the approximation increases with the domain width of the function being approximated, as reported in (Baruch et al., 2023). Additionally, this allows for the use of relatively low-degree polynomials, which significantly reduce the overall multiplication depth and, consequently, decrease the model's latency during secure inference. It is also noteworthy that the values of both the variance and activation norms tend to rise in the deeper layers, posing a greater challenge for approximating these layers.

**Robustness to Context Length and Headcount**

To further explore the robustness of our method with respect to variations in context length and headcount, we conducted additional experiments. Specifically, we evaluated our model configurations with 12 and 16 attention heads in Table 6, and context lengths of 256 and 1024 tokens in Table 5, across both 6-layer and 12-layer architectures. In both tables, 'Original' denotes the perplexity of the vanilla Softmax-based transformer of equivalent size, 'ReLU' represents models utilizing scaled $\sigma$-attention, while 'P+RM' shows perplexity at the end of the range minimization training. Finally, 'Poly' details the score of a fully polynomial model. All results are averaged over 3 seeds and contain perplexity scores of the model that was trained and evaluated on Wikitext-103. These tables demonstrate that regardless of the configuration, our method achieves only a marginal reduction in performance. These results emphasize the robustness of our approach across various context sizes and headcounts.

Table 5: Perplexity Across Various Context Lengths

| Length | Depth | Original | ReLU | P+RM | Poly |
|---|---|---|---|---|---|
| 256 | 6 | 20.07 | 21.87 | 22.04 | 22.38 |
| 512 | 6 | 18.98 | 19.07 | 19.15 | 19.89 |
| 1024 | 6 | 17.85 | 17.96 | 18.22 | 18.59 |
| 256 | 12 | 17.41 | 17.55 | 17.94 | 19.17 |
| 512 | 12 | 16.89 | 16.98 | 17.15 | 18.91 |
| 1024 | 12 | 16.08 | 16.62 | 16.93 | 18.29 |

Table 6: Perplexity Across Different Headcounts

| Heads | Depth | Softmax | ReLU | P+RM | Poly |
|---|---|---|---|---|---|
| 8 | 12 | 16.89 | 16.98 | 17.15 | 18.91 |
| 12 | 12 | 16.76 | 16.96 | 17.13 | 18.73 |
| 16 | 12 | 17.04 | 17.30 | 17.46 | 19.04 |

**Stabilized HE-Friendly ViT** Although $\sigma$-transformers that normalize with BatchNorm rather than LayerNorm do not collapse and provide some non-trivial accuracy in the image classification regime, their performance still falls short of that of standard ViTs with Softmax attention and LayerNorm. To bridge this gap, Sections 4.2 and 4.1 propose three techniques: (i) adding an additional BatchNorm layer to the MLP in each ViT block (A), (ii) normalizing the attention matrix across the attention heads via BatchNorm 2D (QK), and (iii) implementing length scaling (S). In Table 4, we ablate the contributions of each of these methods. These experiments were conducted using the Swin Transformer backbone on the Tiny ImageNet dataset. The results indicate that the incorporation of BatchNorm instead LayerNorm (denoted by B) initially decreases performance compared to the original Softmax-based Swin (denoted by O), with GELU-attention (G.) and ReLU-attention (R.) accuracy dropping to 39.0% and 38.4%, respectively. However,

normalizing the attention matrix across the attention heads with BatchNorm 2D (B+QK) significantly improves accuracy. The subsequent addition of an extra BatchNorm layer (B+QK+A) further enhances performance, nearly matching the original Swin's accuracy. Finally, the implementation of length scaling (B+QK+A+S) improves $\sigma$-attention-based models and closed the gap with the original Swin.

**Accuracy over FHE** To validate that our polynomial models can be precisely computed over FHE, we fine-tuned our 6-layer and 12-layer polynomial transformers, which were pre-trained on the Wikitext-103 dataset, for financial news text classification (Muchinguri, 2022) involving three classes. After fine-tuning, we achieved 72% and 74% accuracy on plaintext, respectively. Then, we tested the models on all 506 encrypted examples in the test set via HElayers (Aharoni et al., 2023). The results showed exactly the same predictions for both encrypted and plaintext examples in 99.5% of the cases, resulting in a similar level of accuracy.

**Performance under FHE** We run HElayers (Aharoni et al., 2023) version 1.5.4 as our HE SDK and set the underlying HE library to HEaaN. The concrete HE parameters were set as follows: We used ciphertexts with $2^{15}$ coefficients, a multiplication depth of 12, fractional part precision of 42, and integer part precision of 18. This context allows us to use up to 9 multiplications before bootstrapping is required. The security parameters were set to provide a solution with 128-bit security. Our hardware involved a computing system that used both CPU and GPU capabilities. The CPU component was an AMD EPYC 7763 64-core processor comprising 32 cores and 32 threads, along with 200 GB of RAM allocated to the processes under evaluation. Complementing this, we utilized an NVIDIA A100-SXM4-80GB GPU with 80GB of memory, which was integral for performing certain parts of the computation. This configuration was designed to exploit the combined processing power of CPU and GPU, ensuring efficient performance for our computational tasks. Under these settings, for a BERT-like transformer with 6 layers and 53.3M parameters (see hyperparameters in Table 7, Appendix B), secure inference over 128 tokens takes 211.15 seconds. In Fig. 6 we visualize a pie chart illustrating the distribution of computation times across different components during secure inference.

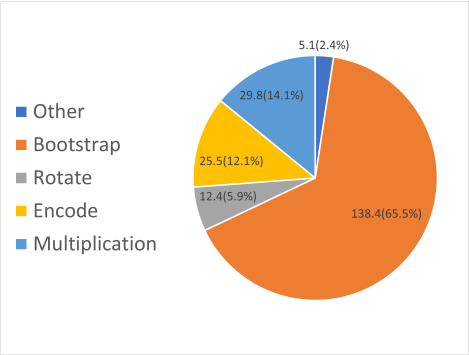


Figure 6: **Breakdown of Runtime for Secure Inference Over HE.** The chart details the time spent on various computational tasks, measured in seconds.

## 6. Conclusion

This paper presents an effective and innovative approach to converting transformers into a polynomial form via the scaled-$\sigma$ attention mechanism and a specialized training procedure that produces HE-friendly weights. Our techniques are the first to propose polynomial alternatives to the self-attention and LayerNorm layers, allowing the deployment of secure inference with transformers for the first time. This advancement significantly extends the potential of the HE-based DL models.

## 7. Limitations

While our approach marks a significant advancement in applying DL over HE, it comes with certain limitations. Firstly, our method does not directly approximate the Softmax function using polynomials. Instead, it employs an alternative architecture and a complementary training procedure. This approach might not fully capture the entire range of behaviors exhibited by the traditional Softmax function in various contexts, potentially affecting the model's performance in specific scenarios. Secondly, the scalability and consistency of our scaled-$\sigma$ attention mechanism as a universal replacement for standard self-attention have not yet been fully established. These aspects should be thoroughly explored in future research, particularly through the use of larger models and datasets, as well as additional modalities such as speech, to ascertain the robustness and versatility of our method across a diverse range of AI applications.

## Acknowledgments

We thank HElayers (Aharoni et al., 2023) developers, particularly Ehud Aharoni and Ramy Masalha, for their technical support and guidance regarding using the HElayers library.

## Impact Statement

Our research introduces the first polynomial transformer, enabling HE-based secure inference with transformers over encrypted data and through encrypted weights. 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.

## References


Aharoni, E., Adir, A., Baruch, M., Drucker, N., Ezov, G., Farkash, A., Greenberg, L., Masalha, R., Moshkowich, G., Murik, D., et al. HElayers: A tile tensors framework for large neural networks on encrypted data. *PoPETs*, 2023. doi: 10.56553/popets-2023-0020.

Ao, W. and Boddeti, V. N. Autofhe: Automated adaption of cnns for efficient evaluation over fhe. *arXiv preprint arXiv:2310.08012*, 2023.

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

Baruch, M., Drucker, N., Ezov, G., Kushnir, E., Lerner, J., Soceanu, O., and Zimerman, I. Sensitive tuning of large scale cnns for e2e secure prediction using homomorphic encryption. *arXiv preprint arXiv:2304.14836*, 2023.

Chen, T., Bao, H., Huang, S., Dong, L., Jiao, B., Jiang, D., Zhou, H., Li, J., and Wei, F. The-x: Privacy-preserving transformer inference with homomorphic encryption. *arXiv preprint arXiv:2206.00216*, 2022.

Cheon, J. H., Kim, A., Kim, M., and Song, Y. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 409–437. Springer, 2017. doi: 10.1007/978-3-319-70694-8_15.

Chrysos, G. G., Moschoglou, S., Bouritsas, G., Panagakis, Y., Deng, J., and Zafeiriou, S. P-nets: Deep polynomial neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7325–7335, 2020.

Dathathri, R., Saarikivi, O., Chen, H., Laine, K., Lauter, K., Maleki, S., Musuvathi, M., and Mytkowicz, T. Chet: An optimizing compiler for fully-homomorphic neural-network inferencing. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2019, pp. 142–156, New York, NY, USA, 2019. ISBN 9781450367127. doi: 10.1145/3314221.3314628.

Ding, Y., Guo, H., Guan, Y., Liu, W., Huo, J., Guan, Z., and Zhang, X. East: Efficient and accurate secure transformer framework for inference. *arXiv preprint arXiv:2308.09923*, 2023.

Egidi, N., Fatone, L., and Misici, L. A New Remez-Type Algorithm for Best Polynomial Approximation. In Sergeyev, Y. D. and Kvasov, D. E. (eds.), *Numerical Computations: Theory and Algorithms*, pp. 56–69, Cham, 2020. Springer International Publishing. doi: 10.1007/978-3-030-39081-5_7.

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

Gilad Bachrach, R., Dowlin, N., Laine, K., Lauter, K., Naehrig, M., and Wernsing, J. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, pp. 201–210, 2016. URL http://proceedings.mlr.press/v48/gilad-bachrach16.pdf.

Gottemukkula, V. Polynomial activation functions. 2020.

Goyal, M., Goyal, R., and Lall, B. Improved polynomial neural networks with normalised activations. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2020.

Gupta, K., Jawalkar, N., Mukherjee, A., Chandran, N., Gupta, D., Panwar, A., and Sharma, R. Sigma: Secure gpt inference with function secret sharing. *Cryptology ePrint Archive*, 2023.

Hao, M., Li, H., Chen, H., Xing, P., Xu, G., and Zhang, T. Iron: Private inference on transformers. *Advances in Neural Information Processing Systems*, 35:15718–15731, 2022.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hesamifard, E., Takabi, H., and Ghasemi, M. Cryptodl: Deep neural networks over encrypted data, 2017. URL https://arxiv.org/abs/1711.05189.

Hua, W., Dai, Z., Liu, H., and Le, Q. Transformer quality in linear time. In *International Conference on Machine Learning*, pp. 9099–9117. PMLR, 2022.

Krizhevsky, A., Sutskever, I., and Hinton, G. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012. doi: 10.1145/3065386.

Lee, E., Lee, J.-W., Lee, J., Kim, Y.-S., Kim, Y., No, J.-S., and Choi, W. Low-complexity deep convolutional neural networks on fully homomorphic encryption using multiplexed parallel convolutions. In *International Conference on Machine Learning*, pp. 12403–12422. PMLR, 2022a.

Lee, E., Lee, J.-W., Lee, J., Kim, Y.-S., Kim, Y., No, J.-S., and Choi, W. Low-complexity deep convolutional neural networks on fully homomorphic encryption using multiplexed parallel convolutions. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 12403–12422. PMLR, 17–23 Jul 2022b. URL https://proceedings.mlr.press/v162/lee22e.html.

Lee, G., Kim, M., Park, J. H., Hwang, S.-w., and Cheon, J. H. Privacy-preserving text classification on bert embeddings with homomorphic encryption. *arXiv preprint arXiv:2210.02574*, 2022c.

Lee, J., Lee, E., Lee, J.-W., Kim, Y., Kim, Y.-S., and No, J.-S. Precise approximation of convolutional neural networks for homomorphically encrypted data. *arXiv preprint arXiv:2105.10879*, 2021. URL https://arxiv.org/abs/2105.10879.

Lee, J.-W., Kang, H., Lee, Y., Choi, W., Eom, J., Deryabin, M., Lee, E., Lee, J., Yoo, D., Kim, Y.-S., and No, J.-S. Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. *IEEE Access*, 10:30039–30054, 2022d. doi: 10.1109/ACCESS.2022.3159694.

Lee, S., Lee, G., Kim, J. W., Shin, J., and Lee, M.-K. Hetal: Efficient privacy-preserving transfer learning with homomorphic encryption. 2023.

Liang, Z., Wang, P., Zhang, R., Xu, N., and Zhang, S. Merge: Fast private text generation. *arXiv preprint arXiv:2305.15769*, 2023.

Liu, X. and Liu, Z. Llms can understand encrypted prompt: Towards privacy-computing friendly transformers. *arXiv preprint arXiv:2305.18396*, 2023.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11976–11986, 2022.

Ma, X., Zhou, C., Kong, X., He, J., Gui, L., Neubig, G., May, J., and Zettlemoyer, L. Mega: moving average equipped gated attention. *arXiv preprint arXiv:2209.10655*, 2022.

Mohassel, P. and Zhang, Y. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 19–38, 2017. doi: 10.1109/SP.2017.12.

Muchinguri, N. Financial news classification dataset. https://huggingface.co/datasets/nickmuchi/financial-classification, 2022. Accessed: 2024-05-26.

Pachón, R. and Trefethen, L. N. Barycentric-remez algorithms for best polynomial approximation in the chebfun system. *BIT Numerical Mathematics*, 49(4):721–741, 2009. doi: https://doi.org/10.1007/s10543-009-0240-1.

Panda, S. Polynomial approximation of inverse sqrt function for fhe. In *International Symposium on Cyber Security, Cryptology, and Machine Learning*, pp. 366–376. Springer, 2022.

Raphson, J. *Analysis aequationum universalis*. Typis TB prostant venales apud A. and I. Churchill, 1702.

Remez, E. Y. Sur la détermination des polynômes d'approximation de degré donnée. *Comm. Soc. Math. Kharkov*, 10(196):41–63, 1934.

So, D. R., Mańke, W., Liu, H., Dai, Z., Shazeer, N., and Le, Q. V. Primer: Searching for efficient transformers for language modeling. *arXiv preprint arXiv:2109.08668*, 2021.

Takabi, D., Podschwadt, R., Druce, J., Wu, C., and Procopio, K. Privacy preserving neural network inference on encrypted data with gpus, 2019. URL https://10.48550/ARXIV.1911.11377.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, J., Wu, J., and Huang, L. Understanding the failure of batch normalization for transformers in nlp. *Advances in Neural Information Processing Systems*, 35:37617–37630, 2022.

Yao, Z., Cao, Y., Lin, Y., Liu, Z., Zhang, Z., and Hu, H. Leveraging batch normalization for vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 413–422, 2021.

Zeng, W., Li, M., Xiong, W., Tong, T., Lu, W.-j., Tan, J., Wang, R., and Huang, R. Mpcvit: Searching for accurate and efficient mpc-friendly vision transformer with heterogeneous attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5052–5063, 2023.

Zheng, M., Lou, Q., and Jiang, L. Primer: Fast private transformer inference on encrypted data. *arXiv preprint arXiv:2303.13679*, 2023.

Zhou, J., Qian, H., Lu, X., Duan, Z., Huang, H., and Shao, Z. Polynomial activation neural networks: Modeling, stability analysis and coverage bp-training. *Neurocomputing*, 359:227–240, 2019.

## A. Our Polynomial Approximations

After modifying the architecture and training using the methods described in Sec. 4, we obtain model architecture and weights that are HE-friendly. At this point, simple and well-known polynomial approximations can be applied to convert the model to a polynomial form. For the GELU and ReLU functions, which are used for attention activations and the MLP activations, we utilize the approximations from (Lee et al., 2021) for ReLU, and polynomials generated with the Remez algorithm for GELU. To approximate the layer normalization, we use polynomials from (Panda, 2022). The polynomials and their approximation errors are presented in Figure 7.

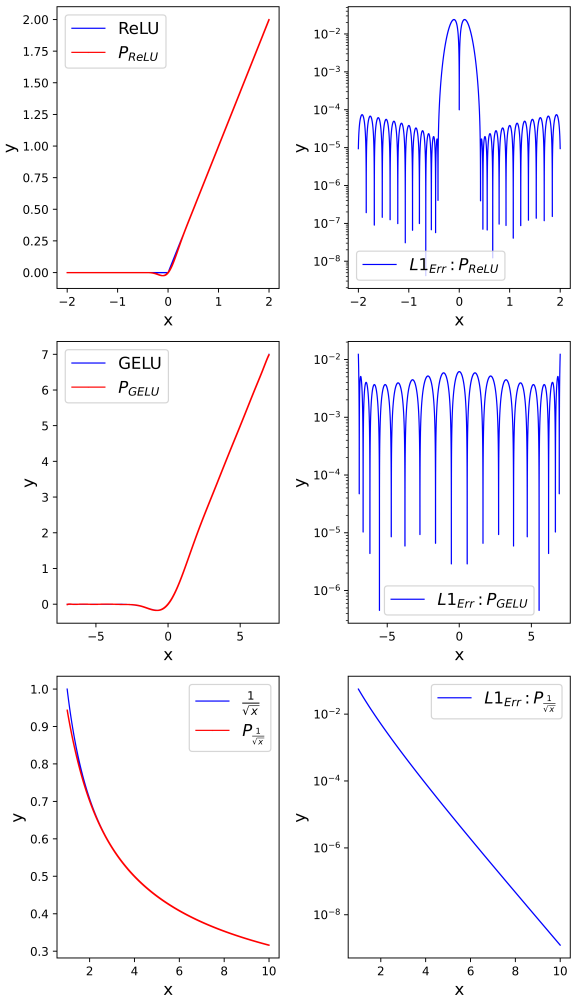


Figure 7: **Our Polynomial Approximations:** (Left): Polynomial obtained by various well-known methods. (Right): L1 approximation error for each approximation presented on a logarithmic scale. At the top, we present polynomials and L1-error for ReLU; in the middle, for GELU, and at the bottom, for $\frac{1}{\sqrt{x}}$.

## B. Experimental Setup

**Training setup** All training experiments were performed on public datasets using a single A100 80GB GPU for a maximum of two days. All experiments were conducted using PyTorch, employing half-precision floating-point format. Results were averaged over three seeds, and all hyperparameters are detailed in Table 7 and Table 8.

| Parameter | Value |
|---|---|
| Model-width | 768 |
| Number of heads | 8 |
| $\sigma$-activation | ReLU |
| Context-length (training) | 512 |
| Batch-size | 256 |
| Optimizer | AdamW |
| Momentum | $\beta_1, \beta_2 = 0.9, 0.999$ |
| Base learning rate | $5e-4$ |
| Weight decay | 0.0 |
| Dropout | 0.1 |
| Training epochs | 100 |
| Learning rate schedule | cosine decay |
| Warmup epochs | 5 |
| Warmup schedule | linear |

Table 7: Hyperparameters for Wikitext-103

Table 8: Hyperparameters for ViT and Swin

| Parameter | ViT | Swin |
|---|---|---|
| Model-width | 192 | 96 |
| Model-depth | 9 | [2, 6, 4] |
| Number of heads | 12 | [3, 6, 12] |
| Trainable parameters | 2.7M | 7.13M |
| Window-size | - | 4 |
| Label smoothing | 0.1 | |
| $\sigma$-activation | GELU | |
| Patch-size | $4 \times 4$ | |
| Batch-size | 128 | |
| Optimizer | AdamW | |
| Momentum | $\beta_1, \beta_2 = 0.9, 0.999$ | |
| Base learning rate | $1e-3$ | |
| Weight decay | 5e-2 | |
| Dropout | 0.1 | |
| Training epochs | 100 | |
| Learning rate schedule | cosine decay | |
| Warmup epochs | 10 | |
| Warmup schedule | linear | |
| Random erasing probability | 0.25 | |