PAF-FHE: Low-Cost Accurate Non-Polynomial Operator Polynomial Approximation in Fully Homomorphic Encryption Based ML Inference

Jingtian Dang  
Carnegie Mellon University

Jianming Tong (jianming.tong@gatech.edu)  
Georgia Institute of Technology

Anupam Golder  
Georgia Institute of Technology

Arijit Raychowdhury  
Georgia Institute of Technology

Cong Hao  
Georgia Institute of Technology

Tushar Krishna  
Georgia Institute of Technology

Research Article

Keywords: Fully Homomorphic Encryption, Non-Polynomial Operators Approximation, Polynomial Approximation Function, Privacy-Preserving Machine Learning

Posted Date: May 11th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-2910088/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License. Read Full License

Additional Declarations: No competing interests reported.
PAF-FHE: Low-Cost Accurate Non-Polynomial Operator Polynomial Approximation in Fully Homomorphic Encryption Based ML Inference

Jingtian Dang¹†, Jianming Tong²*, Anupam Golder³, Arijit Raychowdhury³, Cong Hao³, Tushar Krishna³

¹Electrical and Electronics Engineering, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, 15213, Pennsylvania, U.S.
²School of Computer Science, Georgia Institute of Technology, North Avenue, Atlanta, 30332, Georgia, U.S.
³School of Electrical and Computer Engineering, Georgia Institute of Technology, North Avenue, Atlanta, 30332, Georgia, U.S.

*Corresponding author(s). E-mail(s): jianming.tong@gatech.edu;
Contributing authors: dangjingtian@cmu.edu; anupamgolder@gatech.edu; arijit.raychowdhury@ece.gatech.edu; callie.hao@gatech.edu; tushar@ece.gatech.edu;
†These authors contributed equally to this work.

Abstract

Machine learning (ML) is getting more pervasive. Wide adoption of ML in healthcare, facial recognition, and blockchain involves private and sensitive data. One of the most promising candidates for inference on encrypted data, termed Fully Homomorphic Encryption (FHE), preserves the privacy of both data and the ML model. However, it slows down plaintext inference by six magnitudes, with a root cause of replacing non-polynomial operators with latency-prohibitive 27-degree Polynomial Approximated Function (PAF).

While prior research has investigated low-degree PAFs, naive stochastic gradient descent (SGD) training fails to converge on PAFs with degrees higher than 5, leading to limited accuracy compared to the state-of-the-art 27-degree PAF. Therefore, we propose four training techniques to enable convergence in the post-approximation model using PAFs with an arbitrary degree, including (1) Dynamic Scaling (DS) and Static Scaling (SS) to enable minimal approximation error during approximation, (2) Coefficient Tuning (CT) to obtain a good initial coefficient value for each PAF, (3) Progressive Approximation (PA) to simply the two-variable regression optimization problem into single-variable for fast
and easy convergence, and (4) Alternate Training (AT) to retraining the post-replacement PAFs and other linear layers in a decoupled divide-and-conquer manner. A combination of DS/SS, CT, PA, and AT enables the exploration of accuracy-latency space for FHE-domain ReLU replacement. Leveraging the proposed techniques, we propose a systematic approach (PAF-FHE) to enable low-degree PAF to demonstrate the same accuracy as SotA high-degree PAFs. We evaluated PAFs with various degrees on different models and variant datasets, and PAF-FHE consistently enables low-degree PAF to achieve higher accuracy than SotA PAFs. Specifically, for ResNet-18 under the ImageNet-1k dataset, our spotted optimal 12-degree PAF reduces 56% latency compared to the SotA 27-degree PAF with the same post-replacement accuracy (69.4%). While as for VGG-19 under the CiFar-10 dataset, optimal 12-degree PAF achieves even 0.84% higher accuracy with 72% latency saving. Our code is open-sourced at: https://github.com/TorchFHE/PAF-FHE

**Keywords:** Fully Homomorphic Encryption, Non-Polynomial Operators Approximation, Polynomial Approximation Function, Privacy-Preserving Machine Learning

1 Introduction

As ML becomes more pervasive in fields such as healthcare [16], facial recognition [19], and blockchain [27], concerns regarding privacy leakage of private and sensitive data have arisen. Fully Homomorphic Encryption (FHE) provides a solution to these concerns by allowing for ML inference on encrypted data while preserving the privacy of both the data and models. However, FHE-based ML inference comes with a significant latency overhead, with FHE-based computations taking six orders of magnitude longer than plaintext computations. This latency is primarily due to non-polynomial operators (e.g. ReLU, MaxPooling, and Softmax, etc.), which are computation-intensive under FHE and dominate approximately half of the total latency. Therefore, a major research problem is how to efficiently process non-polynomial operators in FHE.

1.1 Challenges

Non-polynomial operators pose a challenge in Fully Homomorphic Encryption (FHE) due to the lack of native support. To overcome this, previous research explored (1) the hybrid scheme, which offloads non-polynomial operators to other secure schemes with a secure data transfer to communicate data among schemes, and (2) PAF approximation, which replaces non-polynomial operators with Polynomial Approximated Function (PAF).

The hybrid scheme is infeasible in practice because a plethora of prior arts illustrate the prohibitive communication overheads for transferring data securely among different schemes [5, 15, 20]. Despite its promise, PAF approximation is non-trivial because it requires striking a balance between accuracy and latency. A high-accuracy approximation requires high-degree PAFs with a prohibitively long chain of multiplications with bootstrapping. An example includes a SotA 27-degree PAF [10, 13]. On the other hand, a low-latency approximation suffers from severe accuracy degradation, allowing only a subset of non-polynomial operators to be replaced with PAFs. In such cases, there are still some non-polynomial operators being
offloaded to other schemes to reduce the accuracy drop, resulting in prohibitive communication overheads in practice [5, 15, 20]. These approaches are suboptimal and highlight the need for exploring alternative approaches.

To address this challenge, previous research has explored replacing non-polynomial operators with low-latency low-degree Polynomial Approximated Functions (PAFs) followed by fine-tuning of coefficients. This approach aims to find the optimal degree with minimal accuracy degradation in the entire PAF design space. However, existing techniques fail to converge for PAFs with degrees higher than 5, and PAFs with degrees lower than 5 still suffer from severe accuracy degradation. To explore the optimal degree with minimal accuracy degradation in the entire design space, the key challenge is to improve training techniques to enable the convergence of PAF with arbitrary degrees.

1.2 Our Contributions

To tackle the aforementioned dilemma, this paper proposes four key training techniques to enable the convergence of the post-replacement model using PAFs with arbitrary degrees. We propose the first-ever systematic method to adopt proposed training techniques to replace all non-polynomial operators with minimal-degree PAFs without sacrificing accuracy (to the best of our knowledge, the comparison with prior arts is shown in Tab. 1) with our contributions as follows.

- To achieve minimal approximation error, we propose Dynamic Scaling (DS) for training and Static Scaling (SS) for the inference that enables the minimal approximation error during training and avoids infinite approximation error during inference to ensure training stability.
- To decide how many coefficients are needed in PAF, we propose Coefficient Tuning (CT) that fine-tunes initial coefficients before replacing the non-polynomial operators to improve the post-replacement accuracy by $1.04 \times \sim 2.38 \times$ for PAFs of varying degrees.
- To replace non-polynomial operators, we propose Progressive Approximation (PA) to simplify variables of regression problem in approximation to only PAFs coefficients, resulting in an easy-converge convex question to enable exploration of the full accuracy-latency tradeoff space.
- To fine-tune post-replacement PAF coefficients, we propose Alternate Training (AT) to decouple and train weights and coefficients alternately, resulting in progressive accuracy climbing by $1.009 \sim 1.037 \times$ of retraining post-replacement ML model.
- We developed a systematic scheduler to automatically explore the accuracy-latency tradeoff space for multiple models on different datasets. PAF-FHE consistently enables low-degree PAFs to demonstrate higher accuracy than high-degree SotA PAFs. For example, PAF-FHE spots the sweet-point PAF with 12 degrees, which can achieve the same 69.4% validation accuracy with 72% reduction in latency for in ResNet-18 inference on ImageNet 1k, compared to the SotA 27-degree PAF [13].
Table 1: Comparison of proposed strategies with SotA

<table>
<thead>
<tr>
<th></th>
<th>Low Communication Overhead</th>
<th>Low Accuracy Degradation</th>
<th>Low Latency Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>SafeNet [15], CryptoGCN [20]</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CryptoNet [5], CryptoDL [7], LoLa [13], CHE [26]</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>F1 [24], CraterLake [25], BTS [10]</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>HEAX [22], Delphi [17], Gazelle [8], Cheetah [21]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SHE [14]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAF–FHE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Secure Inference - Linear Operators Only

Fig. 1: Overview of the FHE-base ML inference where original non-polynomial operators are replaced by Polynomial Approximated Activation (PAF).

2 Technical Background

2.1 Non-polynomial Operators in FHE-based ML Inference

FHE is an asymmetric encryption scheme that enables ciphertext-based computation with Cheon-Kin-Kim-Song (CKKS) [3] as the most commonly used FHE scheme for machine learning inference due to its superior efficiency in approximate computation compared to other schemes such as BGV, BFV, and TFHE [22]. Under the CKKS scheme, only polynomial operators are allowed as shown in Fig. 1 such that all non-polynomial operators including both ReLU and MaxPooling must be replaced by PAF (Approximation) or offloaded to other schemes (Hybrid Scheme). Prior works have also shown that approximation offers superior performance and lower overhead compared to the hybrid scheme [15].

2.2 Polynomial Approximated Function (PAF)

High-degree polynomials are theoretically capable of approximating arbitrary functions, but a direct approximation of non-polynomial operators like ReLU or MaxPooling using PAFs can result in severe approximation errors [12]. Prior research has shown that it is more effective to approximate the \( \text{sign}(x) \) function \(^1\) and then construct ReLU and Max using it, such as with \( \frac{(x+\text{sign}(x)x)}{2} \) and \( \frac{(x+y)+(x-y)\text{sign}(x-y)}{2} \).

Lower latency and higher accuracy (low accuracy degradation) are two fundamental goals of replacing non-polynomial operators with PAFs. Latency is determined by the degree of the polynomial, as FHE-based multiplication dominates latency. Accuracy degradation arises from the difference between the PAF and the original non-polynomial operators, as PAF cannot 100% precisely approximate targeted non-polynomial functions. Such differences can vary depending on the degree of the PAF, the cascaded format of the PAF, and the coefficient values of the PAF.

\(^1\) \( \text{sign}(x) \) outputs 1 if \( x \) is positive, −1 if \( x \) is negative, and 0 for zero

4
Table 2: Baseline PAF and corresponding multiplication depth

<table>
<thead>
<tr>
<th>Form</th>
<th>Degree</th>
<th>Multiplication Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha = 10$</td>
<td>27</td>
<td>29</td>
</tr>
<tr>
<td>$f_1^2 \circ g_1^2$</td>
<td>14</td>
<td>12</td>
</tr>
<tr>
<td>$\alpha = 7$</td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>$f_2 \circ g_3$</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>$f_2 \circ g_2$</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>$f_1 \circ g_2$</td>
<td>8</td>
<td>7</td>
</tr>
</tbody>
</table>

We adopt cascaded polynomial PAFs throughout the paper because they achieve lower approximation error than single polynomials within the same maximal degree [12, 13]. Tab. 2 shows the selected PAFs with the minimal multiplication depth under different degrees constraints. We use the notation $f^n$ to indicate a serial nested function call of the same polynomial $f$ for $n$ times, for example, $f^2 = f(f(x))$.

2.3 PAF Approximation Input Range

The effectiveness of PAFs in approximating non-polynomial operators depends on their input range, which determines the extent to which the PAF can approximate the non-polynomial operator’s output accurately. For example, setting the approximation input range as $[-1, 1]$ for a PAF indicates that it gives relatively more accurate approximations of non-polynomial operators’ output when the input falls in $[-1, 1]$ than outside this range. However, determining an appropriate input range can be challenging because a narrow input range can lead to severe approximation loss for input value outside the input range and directly lead to training divergence. In contrast, a broader range may result in a large average approximation error across the entire input range, because of the limited representation capability of PAFs to capture negligible differences among input values under a broad range. Finding the optimal approximation input range is still an open question.

3 Problem Statement

3.1 Polynomial Approximation Problem Statement

The essential problem for approximating non-polynomial operators with Polynomial Approximation Function (PAF) is the regression problem shown in Equ. 1. Specifically, the goal is to find a vector of coefficients $a = a_0, a_1, a_2, \cdots, a_D$ such that the cumulative error of replacing non-polynomial function $R(x_i)$ by a $D$-degree PAFs is minimal given input data as $x_i(i = 0, \cdots, N)$.

$$\min_\alpha f(a) = \frac{1}{N} \sum_{i=0}^{N} (R(x_i) - a \cdot x)^2$$

where $x = x_i^0, x_i^1, \cdots, x_i^D$, and $D$ stands for the maximal degree of PAF. $N$ represents the total number of input data.

The Equ. 1 is convex such that the stochastic gradient descent could guarantee finding the optimal solution $a^*$ given initial value as $a_0$. The error could converge in the speed order as shown in Equ. 2 after training for $T$ epochs [1]. And a good initialization value of $a$ could reduce the total error and thus improve the convergence speed.

$$f(a_T) - f(a^*) = O\left(\frac{||a_0 - a^*||^2}{\sqrt{T}}\right)$$

where $a_T$ indicates the value of $a$ after training for $T$ epochs.
3.2 post-replacement Model Retraining

The post-replacement neural network replaces non-polynomial operators with PAFs. Such a replacement changes the original model structure, leading to changed data distribution of $x_i$ for all layers after the replacement points. Therefore, original coefficients trained using the original model structure may perform worse under new data distribution, leading to accuracy degradation. Therefore, the PAF replacement forces retraining to fine-tune coefficients to learn new input data.

4 Proposed Method

4.1 Overview

Training methods adopted by prior arts lead to coefficients divergence when training PAF with a degree higher than 5, forbidding exploration using PAFs with higher degrees to enhance overall accuracy. In this section, we thus analyze critical reasons behind the divergence of existing training algorithms and propose corresponding four solution techniques to guarantee training convergence while replacing non-polynomial operators with PAFs of arbitrary degrees. Then we propose a systematic training schedule strategy to apply all techniques automatically.

4.2 Dynamic Scaling (DS) and Static Scaling (SS)

In Section 2.3, we discuss the challenge of setting an appropriate input range for PAFs. Previous studies have employed a large input range, such as $[-50, 50]$, to prevent infinite errors for values outside the range [13]. However, we contend that this approach is suboptimal because it results in a high average error across the entire input range, which can impede training convergence and lead to significant accuracy degradation (e.g., dropping from 69% to 10% for ResNet-18 on ImageNet-1k).

To address such a challenge, we choose Dynamic Scaling (DS) for training to scale data range to $[-1, 1]$. DS selects the scale in the granularity of the batch. For every input data batch, scale is selected as the input value with the largest absolute value, and then all data get scaled down to $[-1, 1]$ such that each input value could get the maximal difference without falling outside the range during training. However, the post-training model should be deployed in FHE, and DS cannot be used as FHE doesn’t support MAX function. Therefore, we propose Static Scaling (SS) to fix scale in layer-wise granularity. For example, for each layer, the scale is chosen as the value with the largest absolute value within the entire training input data at this layer.

---

2Such a scale determination relies on the similarity between training data and validation data. In our evaluation, either higher or smaller scale results in lower accuracy for both VGG-19 (CIFAR-10 dataset) and ResNet-18 (ImageNet-1k dataset).
4.3 Coefficients Tuning (CT)

Equ. 2 indicates that a good initialization can reduce the overall training time, facilitating faster convergence, which has been ignored by prior arts. These studies have typically initialized PAFs with the same coefficients using traditional regression algorithms. Such initialization is suboptimal because it overlooks the differences between different non-polynomial operators.

Therefore, we propose Coefficients Tuning (CT) as a technique to use profiled data distribution to obtain a closer-to-optimum initialization point and then use PAF with different initialization points to replace different non-polynomial operators, as shown in Fig. 2. The CT process involves the following steps: (1) obtain the coefficients of PAFs using traditional regression methods on the full range of input data; (2) profile the input data distribution for each non-polynomial operator; (3) tuned PAF coefficients to minimize overall approximation errors on profiled distributions to obtain the post-CT PAFs; (4) replace original non-polynomial operators with the post-CT PAFs.

Intuitively, CT tunes PAFs to achieve higher accuracy and less approximation error over a reduced smaller input range, which corresponds to the highest-probability range in the data distribution. Besides, CT reduces training time because the initial value of coefficients $a_0$ are closer to the optimal value $a^*$ as indicated by Equ. 2.

4.4 Progressive Approximation (PA)

Replacing all non-polynomial operators in the given ML model with PAFs simultaneously, as done by prior arts, is a suboptimal approach. This is because approximation errors of early replacements propagate to input data for all layers behind the replacement point, making the regression (Equ. 1) non-convex by varying both $a$ and $x_i$ and causing training divergence. Instead, we propose the Progressive Approximation (PA) approach, which replaces each non-polynomial operator one by one. In such cases, as for each replacement point, $x_i$ statistically gets fixed with only $a$ keeping variant, ensuring a simple convex regression problem shown in Equ. 1, which is easily optimizable by SGD.

Specifically, the PA replaces the non-polynomial operator with PAF, one layer at a time, followed by fine-tuning of PAF coefficients until accuracy convergence as shown in Fig. 2. In the first step of PA, the first ReLU is replaced with a PAF, and the replaced PAF gets retrained.
Replacing the non-polynomial operators with PAFs requires model retraining, as coefficients trained for the original model are no longer optimal for the new post-replacement model. However, multiple previous works retrain the post-replacement models as a whole to tune both PAF coefficients and the parameters of other linear operators. Such a training scheme, however, deteriorates accuracy, indicating a failure of convergence. This is because modifications to the coefficients of PAFs can have a significant impact on the final inference results, unlike changes to convolution weights, which may not affect the results at all. Such a finding is consistent with the model structure that all data need to go through PAF while some weights can be pruned without any impact on overall accuracy.

Therefore, the training of PAFs coefficients should be decoupled from the training of the parameters of other linear operators under even different training hyperparameters. We thus propose Alternate Training (AT) to train PAF coefficients and parameters of other layers separately in an alternate manner. Specifically, AT involves training PAF coefficients first...
During training, PAF coefficients get frozen and we train parameters of other linear operators in AT step 2. Note that the training in different AT steps may use different hyperparameters because of the changing parameter sensitivity. The alternate training repeats until the accuracy finally converges, resulting in an accuracy climb.

Intuitively, AT considers the sensitivity difference between PAF parameters and parameters of linear operators and decouples the two training processes to avoid training interference.

4.6 Systematic Training Scheduler

Different orders of applying DS/SS, CT, PA, and AT can significantly affect the final validation accuracy. Besides, existing tricks including dropout to avoid overfitting and Stochastic Weights Averaging (SWA) also need to be considered to achieve faster convergence. We thus develop a scheduler to change the training configurations and apply different training techniques as shown in Fig. 5, which takes the predefined PAF format and ML model as input. And it keeps track of the index non-polynomial operators, the model with the highest validation accuracy (“best_model”), and branching variables.

The scheduler trains the given model for $E$ epochs at every step, applying SWA using training updates from all $E$ epochs, and selects the model with the highest accuracy from all $E$ epochs and the post-SWA model. If such a selected model achieves higher accuracy than the current “best_model”, then the scheduler updates the “best_model” and repeats the model process until no further accuracy improvement is observed. Then the scheduler applies dropout and AT based on the overfitting detection and condition variable (“Apply_AT”) and then puts the model back to one more round of training. Note that both AT and dropout could be applied multiple times, enabling further accuracy climb.

*we adopt the empirical overfitting condition, which is “training accuracy > validation accuracy + 10%”*
5 Evaluation

5.1 Model, Benchmark and Dataset

We evaluate PAF-FHE was using ResNet-18, a model comprising 17 ReLU layers. ImageNet 1k, a widely used validation dataset in practical applications, was utilized for validation [6, 11]. The dataset consists of 1000 training and 1000 validation images. We use ResNet-18 to study the effect of different techniques because it is the most popular ML model that consists of both ReLU and MaxPooling layers. To further evaluate the generality across different models and datasets, we evaluate PAF-FHE further using VGG-19 on CI-Far-10, a test candidate used by various prior works [13, 15] and compare PAF-FHE with previous SotA implementations.

5.2 Coefficients Tuning Evaluation

Coefficients Tuning (CT), as shown in Fig. 6 \(^4\), improves overall validation accuracy by \(1.04 \sim 2.38\times\) compared to the baseline shown in Tab. 2. The benefits of CT are more pronounced for polynomials with lower degrees, as they have less capability to fit the entire input range and are more prone to significant accuracy reduction. CT mitigates this loss by focusing on fitting the high-probability region in the distribution, resulting in less initial post-replacement approximated error. On the other hand, polynomials with higher degrees have less overall approximated error across the entire input range and, therefore, show less improvement from CT.

\(^4\)We only applied CT and DS for ReLU layers to study the pure effect of CT.
5.3 Progressive Approximation Evaluation

Progressive Approximation (PA) combines two techniques: (a) progressive replacement of ReLU with PAFs as opposed to direct replacement of all ReLU at once, and (b) progressive training of the post-replacement model parameters compared to direct training of all parameters. This results in four different configurations, as shown in Fig. 7.

Among most PAFs with degrees ranging from 8 to 27, Progressive Approximation (PA) achieves the best overall accuracy because it allows gradual optimization of small deviations introduced by the approximation error. Furthermore, progressive training is more critical than

---

[^1]: We only applied PA and DS for ReLU layers to study the pure effect of PA.
progressive replacement, as the accuracy improvement between direct replacement with pro-
gressive training and progressive replacement with progressive training is similar. In some
cases, direct replacement with progressive training even yields better validation accuracy, e.g.
in the \( f_1 \circ g_2 \) configuration.

### 5.4 Ablation Study of All Proposed Techniques

To demonstrate the effectiveness of the proposed techniques, we conduct an ablation
study with results presented in Tab. 3. Tab. 3 separate ReLU from MaxPooling to show
approximation effects of different types of non-linear operators.

When only replacing ReLU with PAFs with different degrees, PAF-FHE consistently
improve accuracy over the baseline training strategy, by \( 1.03 \times \sim 1.19 \times \) for different degrees,
rendering a \( 1.06 \sim 3.39 \times \) accuracy compensation from direct ReLU replacement. Further,
when replacing all non-linear operators, PAF-FHE could improve up-to \( 1.08 \times \) accuracy
improvement, which already achieves the same accuracy as the original ResNet-18 without
any replacement, demonstrating the validity of PAF-FHE. Beyond this, we could also see
that the MaxPooling operator is more sensitive to PAF replacement because a single slid-
ing window in MaxPooling involves a nested call of PAFs to obtain the final output, where
approximation errors propagate and accumulates.

In summary, all PAFs with varying degrees create a natural tradeoff space between accu-
racy and latency. Our proposed approach allows us to identify the optimal point in this tradeoff
space, a 12-degree PAF (\( f_2^2 \circ g_1^2 \)) that achieves the same 69.04% post-replacement accuracy
with 55% lower latency compared to state-of-the-art 27-degree polynomials.

### 5.5 Comparison with SotA Implementations

We also compare optimal PAFs trained using our proposed techniques with SotA PAFs with
different degrees ranging from 5 to 27 on various ML models under different datasets.

---

**Table 3: Ablation study of all different proposed techniques**

<table>
<thead>
<tr>
<th>Replace ReLU Only</th>
<th>( f_2^2 \circ g_1^2 )</th>
<th>( f_2 \circ g_3 )</th>
<th>( f_2 \circ g_2 )</th>
<th>( f_1 \circ g_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>direct replacement</td>
<td>51.30%</td>
<td>64.70%</td>
<td>49.40%</td>
<td>32.00%</td>
</tr>
<tr>
<td>baseline</td>
<td>64.30%</td>
<td>66.70%</td>
<td>64.20%</td>
<td>58.30%</td>
</tr>
<tr>
<td>CT</td>
<td>68.60%</td>
<td>67.70%</td>
<td>67.00%</td>
<td>66.50%</td>
</tr>
<tr>
<td>AT</td>
<td>65.20%</td>
<td>68.30%</td>
<td>63.70%</td>
<td>60.50%</td>
</tr>
<tr>
<td>PA</td>
<td>65.60%</td>
<td>68.40%</td>
<td>64.00%</td>
<td>60.20%</td>
</tr>
<tr>
<td>PA + AT</td>
<td>64.90%</td>
<td>67.40%</td>
<td>64.60%</td>
<td>56.50%</td>
</tr>
<tr>
<td>CT + PA</td>
<td>68.20%</td>
<td>67.00%</td>
<td>67.60%</td>
<td>65.90%</td>
</tr>
<tr>
<td>CT + PA + AT + SS</td>
<td>69.00%</td>
<td>68.10%</td>
<td>66.50%</td>
<td>63.10%</td>
</tr>
</tbody>
</table>

\[ \begin{array}{cccc} 
\text{Accuracy Improvement} & \text{Accuracy Improvement} & \text{Accuracy Improvement} & \text{Accuracy Improvement} \\
\text{over Direct Replacement} & \text{over Baseline} & \text{over Baseline} & \text{over Baseline} \\
\end{array} \]

<table>
<thead>
<tr>
<th>Replace all non-polynomial</th>
<th>( f_2^2 \circ g_1^2 )</th>
<th>( f_2 \circ g_3 )</th>
<th>( f_2 \circ g_2 )</th>
<th>( f_1 \circ g_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT + PA + AT + SS</td>
<td>69.4%</td>
<td>67%</td>
<td>65.3%</td>
<td>37.3%</td>
</tr>
</tbody>
</table>

\[ \begin{array}{cccc} 
\text{Accuracy Improvement} & \text{Accuracy Improvement} & \text{Accuracy Improvement} & \text{Accuracy Improvement} \\
\text{over Direct Replacement} & \text{over Baseline} & \text{over Baseline} & \text{over Baseline} \\
\end{array} \]

\[ \begin{array}{cccc} 
1.07 \times & 1.03 \times & 1.05 \times & 1.14 \times & 1.19 \times \\
1.07 \times & 1.22 \times & 1.27 \times & 1.79 \times & 0.22 \times \\
1.08 \times & 1.01 \times & 1.02 \times & 0.98 \times & 0.12 \times \\
\end{array} \]
Table 4: PAF-FHE V.S. SotA (VGG19 - CiFar-10 - original accuracy 93.95%)

<table>
<thead>
<tr>
<th>PAF Format</th>
<th>PAF-FHE</th>
<th>Lee [13]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiplication Depth</td>
<td>f₁ ◦ g₂</td>
<td>f₂ ◦ g₂</td>
</tr>
<tr>
<td>Validation Accuracy</td>
<td>72.24%</td>
<td>86.36%</td>
</tr>
</tbody>
</table>

Table 5: PAF-FHE V.S. SotA (VGG19 - ImageNet 1k - original accuracy 75%)

<table>
<thead>
<tr>
<th>PAF Format</th>
<th>PAF-FHE</th>
<th>Lee [13]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiplication Depth</td>
<td>f₁ ◦ g₂</td>
<td>f₂ ◦ g₂</td>
</tr>
<tr>
<td>Validation Accuracy</td>
<td>1.1%</td>
<td>50.1%</td>
</tr>
</tbody>
</table>

Note: N/A means the paper doesn’t provide results on PAFs with the same multiplication depth.

5.5.1 Comparison of VGG-19 on CiFar-10
Tab. 4 demonstrates that PAF-FHE enables lower degree polynomials to achieve even higher validation accuracy than the SotA implementation [13], e.g. 9-multiplication-depth PAF f₂ ◦ g₃ could achieve 91.05% accuracy compared to 27-multiplication-depth SotA PAF with 90.21% accuracy [13]. By leveraging higher multiplication depth, even higher accuracy (92.39%) can be achieved. Such benefits mainly come from Dynamic Scaling training, Static Scaling inference, and Coefficient Tuning, which together boost around 56% to 81% validation accuracy compared to the baseline [13].

5.5.2 Comparison of VGG-19 on ImageNet-1k
To demonstrate the validity of PAF-FHE on datasets and tasks with higher complexity, we also evaluate the same VGG-19 model on ImageNet1k with results shown in Tab. 5. PAF-FHE trades 72% latency for 3% accuracy degradation as illustrated by the f₂ ◦ g₃.

6 Related Work
ML inference is prone to a severe threat of private information leakage in the cloud. The demand for privacy-preserving ML inference invalidates traditional cryptographic schemes like Advanced Encryption Standard (AES) [4] where encrypted ciphertext has to be decrypted for processing. Moreover, the continuous streaming data in cloud applications also invalidate the Multi-Party Computation [23] and Garbled Circuits (GC) [9] because both of them rely on prohibitively high communication overhead. To eliminate the overhead of communication as well as the secure data sharing, non-polynomial operators can be approximated by low-degree polynomials for inference [2, 5, 7, 13, 18, 26].

In the current FHE schemes which most prior arts focus on, non-polynomial operators like ReLU or MaxPooling are not supported. Therefore, HEAX [22], Delphi [17], Gazelle [8], and Cheetah [21] presented hybrid schemes consisting of FHE, MPC, or GC, where they relied on non-FHE scheme to process non-polynomial kernels. However, both MPC and GC are non-realistic because of the large-size packets in need of transferring among data sources and compute nodes. Alternatively, F1 [24], CraterLake [25], and BTS [10] propose replacing all non-polynomial kernels with a single low-degree polynomial approximation and processing it in the FHE domain. However, the latency gets bottlenecked by the replaced polynomial approximation for a long multiplication chain. To further reduce the overhead
of non-polynomial kernels, SAFENet [15], and CryptoGCN [20] proposed a finer granular replacement of non-polynomial operators with a lower degree, then they leverage ML training to figure out the parameters of each individual approximated polynomial. However, they suffer from significant accuracy degradation because of the non-convergence of training in high-degree polynomials.

7 Conclusion

This paper demonstrates that the training of ML models with non-linear operators replaced with Polynomial Approximated Functions (PAF) is a fundamentally different problem than the typical model training, such that typical training algorithms hardly converge and even lead to worse accuracy. The limitation of training algorithms hinders the exploration of using PAF (higher than 5 degrees) to replace non-linear operators for better post-replacement accuracy. This paper proposes four techniques to address such a challenge: (1) Dynamic Scale (DS) in training and Static Scale (SS) in post-training post-replacement inference under FHE to deliver minimal scaling without infinite approximation error; (2) Coefficients Tuning to provide good initialization of PAF coefficients using profiled data distribution for higher post-replacement initial accuracy and faster convergence; (3) Progressive Approximation ensures convex regression problem through layer-wise non-linear operators replacement and training; and (4) Alternate Training to decouple training of PAF coefficients and other parameters to avoid inference. PAF-FHE is thus proposed to automatically apply the above techniques as well as dropout and SWA to improve post-replacement model training. Leveraging PAF-FHE, we explore the tradeoff space of PAF with variant degrees ranging from 8 to 27 and found that a 12-degree PAF yielded optimal results. Our evaluation of it on ResNet-18 (ImageNet 1k dataset) demonstrates a 69.4% overall post-approximated accuracy (same accuracy as pre-replacement ResNet-18) with 55% latency reduced compared to SotA 27-degree PAF. We believe that PAF-FHE potentially opens a new paradigm of obtaining PAF through ML training.

Appendix A Deep Dive into Training Processing

In this section, we perform a detailed comparison and analysis of the training curve using both the baseline training strategy and our proposed training techniques with the 14-degree PAF (α = 7) [13] in Tab. 2.

In the baseline, all ReLU operators get replaced by the same PAF initiated by traditional algorithm [12, 13] such that baseline training strategy diverges and leads to worse validation accuracy, and it has to resort to Stochastic Weights Averaging (SWA) to smooth the data update to maintain the original accuracy. Furthermore, every single training process after SWA leads to worse accuracy as shown by the dropping trend of the blue curve after each yellow star in Fig. A1, indicating an invalidation of the prior training strategy on post-replacement models.

On the contrary, our proposed techniques replace the ReLU progressively with post-CT PAFs, such that there won’t be a severe accuracy drop in the very beginning shown in Fig. A1, resulting in a 3% accuracy improvement compared to the baseline. During the middle of the training, each replacement of ReLU with PAF causes some accuracy degradation, which is
Fig. A1: The training curve of ResNet-18 with ReLU approximated by PAF (14-degree) w/ and w/o proposed training techniques.

optimized back through both Stochastic Weight Averaging (SWA) and Alternate Training (AT), as shown in the climbing orange curve after each ReLU replacement in Fig. A1.

The detailed comparison illustrates both the invalidity of general training algorithms for post-replacement ML models and the effectiveness of our proposed training techniques to enable accuracy convergence.

Appendix B  Training Hyperparameters

The following hyperparameters were used as the baseline training methodology to train the machine learning model.

<table>
<thead>
<tr>
<th>Model</th>
<th>ResNet-18 (pretrained)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset</td>
<td>ImageNet-1k</td>
</tr>
<tr>
<td>Replaced layer</td>
<td>ReLU</td>
</tr>
<tr>
<td>Original validation accuracy</td>
<td>69.3%</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Adam</td>
</tr>
<tr>
<td>The learning rate for PAF</td>
<td>1e-4</td>
</tr>
<tr>
<td>Learning rate for other layers except PAF.</td>
<td>1e-5</td>
</tr>
<tr>
<td>Weight decay for PAF</td>
<td>0.01</td>
</tr>
<tr>
<td>Weight decay for other layers</td>
<td>0.1</td>
</tr>
<tr>
<td>BatchNorm Tracking No update to running average and variance</td>
<td>False</td>
</tr>
<tr>
<td>Dropout</td>
<td>False</td>
</tr>
</tbody>
</table>

Appendix C  Value of the Best Polynomial Coefficients

In this section, we present the specific coefficient values of polynomials with the highest validation accuracy shown in bold in Tab. 3. All proposed techniques including Coefficients Tuning (CT), Progressive Approximation (PA), and Alternate Training (AT) happen at the granularity of the layer and thus PAFs at different ReLU layers have different coefficients.
C.1 $\alpha = 7$

**Mathematical Format:** The original format for the MiniMax approximated polynomial ($\alpha = 7$) [13] is shown in Equ. C1. Such a proposed polynomial $p_7(x)$ was used to approximate the $\text{sign}(x)$ function, which outputs 1 when $x$ is positive and $-1$ when $x$ is negative. Then non-polynomial ReLU could be constructed using $\frac{x + \text{sign}(x)}{2}$.

\[
p_7(x) = p_7,2(x) \circ p_7,1(x)
\]
\[
p_7,1(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 + a_6 x^6 + a_7 x^7
\]
\[
p_7,2(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + b_4 x^4 + b_5 x^5 + b_6 x^6 + b_7 x^7
\]

The odd function nature of $\text{sign}(x)$ results in a negligible coefficients value of composite polynomial $p_7(x)$ in entries with an even degree of $x$. Such even-degree entries could be safely removed without the impact on the overall accuracy. Therefore, we remove all entries with even degrees to obtain $p_7(x)$ in the format shown in Equ. C2.

\[
p_7(x) = p_{odd\ only}^7,2(x) \circ p_{odd\ only}^7,1(x)
\]
\[
p_{odd\ only}^7,1(x) = a_1 x + a_3 x^3 + a_5 x^5 + a_7 x^7
\]
\[
p_{odd\ only}^7,2(x) = b_1 x + b_3 x^3 + b_5 x^5 + b_7 x^7
\]

Under such polynomial format, coefficients used in PAF for replacing ReLU at different locations are shown in Tab. C2.

**Table C2:** Coefficients value for minimax composite polynomial ($\alpha = 7$) used in PAF to replace all ReLU functions

<table>
<thead>
<tr>
<th>$a_1$</th>
<th>$a_3$</th>
<th>$a_5$</th>
<th>$a_7$</th>
<th>$b_1$</th>
<th>$b_3$</th>
<th>$b_5$</th>
<th>$b_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.304451</td>
<td>-34.68258667</td>
<td>59.85965347</td>
<td>-31.87552261</td>
<td>2.400856</td>
<td>-2.631254435</td>
<td>1.549126744</td>
<td>-0.331172943</td>
</tr>
</tbody>
</table>

C.2 $f_1^2 \circ g_1^2$

The format of $f_1$ and $g_1$ are shown in Equ. C3.

\[
f_1(x) = c_1 \cdot x + c_3 \cdot x^3
\]
\[
g_1(x) = d_1 \cdot x + d_3 \cdot x^3
\]

$f_1^2 \circ g_1^2$ refers to combining both $f_1$ and $g_1$ into a composite polynomial in the sequence shown in Equ. C4, with the coefficients for different PAF at different ReLU layers shown in

\[
f_1^2 \circ g_1^2(x) = g_1^2(f_1^0(f_1^0(x)))]
\]

Coefficients of PAF of $f_1^2 \circ g_1^2$ for replacing different ReLU layers are shown in Tab. C3.
C.3 \( f_2 \circ g_3 \)

Similar to \( f_1 \circ g_1 \), the polynomial format of \( f_2 \) and \( g_3 \) are shown in Equ. C5 and Equ. C6, separately.\(^6\)

\[
f_2(x) = c_1 \cdot x + c_3 \cdot x^3 + c_5 \cdot x^5
\]

(C5)

\[
g_3(x) = d_1 \cdot x + d_3 \cdot x^3 + d_5 \cdot x^5 + d_7 \cdot x^7
\]

(C6)

The best post-trained coefficients value are shown in Tab. C5.

### Table C4: \( f_2 \circ g_3 \) best coefficients list

<table>
<thead>
<tr>
<th>layer id</th>
<th>( c_1 )</th>
<th>( c_3 )</th>
<th>( c_5 )</th>
<th>( d_1 )</th>
<th>( d_3 )</th>
<th>( d_5 )</th>
<th>( g )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.484929323</td>
<td>-7.0364687872</td>
<td>0.365895195</td>
<td>4.988522456</td>
<td>-17.01627541</td>
<td>25.34877856</td>
<td>-12.4504</td>
</tr>
<tr>
<td>6</td>
<td>1.875</td>
<td>-1.25</td>
<td>0.375</td>
<td>4.481445313</td>
<td>-16.18847656</td>
<td>25.01367188</td>
<td>-12.5586</td>
</tr>
</tbody>
</table>

### C.4 \( f_2 \circ g_2 \)

The \( f_2 \) is the same function type as Equ. C5 while the function format of \( g_2 \) is shown Equ. C7.

\[
g_2(x) = d_1 \cdot x + d_3 \cdot x^3 + d_5 \cdot x^5
\]

(C7)  

\(^6\)The \( d_i \) in Equ. C6 is a placeholder. All \( d_i \) in different \( g_n \) (\( n = 1, 2, 3, \ldots \)) refer to different coefficients.
Table C5: $f_2 \circ g_2$ best coefficients list

<table>
<thead>
<tr>
<th>g3 layer id</th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$d_1$</th>
<th>$d_2$</th>
<th>$d_3$</th>
<th>$d_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3.632708073</td>
<td>-8.879578590</td>
<td>4.333632946</td>
<td>3.700465441</td>
<td>-7.351731300</td>
<td>5.071476460</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3.412810802</td>
<td>-7.75233164</td>
<td>4.516210556</td>
<td>3.855789393</td>
<td>-7.789761543</td>
<td>5.177268505</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3.585527401</td>
<td>-8.58312149</td>
<td>5.618574142</td>
<td>3.640014887</td>
<td>-7.15984440</td>
<td>5.668038568</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3.533124393</td>
<td>-5.282323038</td>
<td>6.205972672</td>
<td>3.779301486</td>
<td>-7.770857111</td>
<td>5.675621064</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1.875000000</td>
<td>-1.250000000</td>
<td>0.375000000</td>
<td>3.255859375</td>
<td>-5.964843750</td>
<td>5.707310235</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>3.236021818</td>
<td>-7.84849886</td>
<td>4.585978271</td>
<td>3.662446976</td>
<td>-7.393878849</td>
<td>5.480692863</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>3.324039100</td>
<td>-7.291855249</td>
<td>3.084533691</td>
<td>3.766145468</td>
<td>-8.078896523</td>
<td>5.617486657</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>3.344879531</td>
<td>-5.13930321</td>
<td>5.686520100</td>
<td>3.717407744</td>
<td>-7.314604759</td>
<td>5.406718673</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>2.35330588</td>
<td>-8.009678841</td>
<td>5.265969452</td>
<td>3.645534482</td>
<td>-7.250711864</td>
<td>5.292572514</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>2.295436418</td>
<td>-7.35520248</td>
<td>4.257196426</td>
<td>3.702267428</td>
<td>-7.39237194</td>
<td>5.368722439</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>3.318752050</td>
<td>-8.203745842</td>
<td>5.435956478</td>
<td>3.63973330</td>
<td>-7.331366062</td>
<td>5.393109322</td>
<td></td>
</tr>
</tbody>
</table>

Acknowledgments. Jingtian Dang and Jianming Tong contributed to the experiments and prepared all figures. Jianming Tong, Jingtian Dang, and Anupam Golder proposed ideas and wrote the main manuscript text. Arijit Raychowdhury contributed to FHE theory supervising. Callie Hao contributed to real-device deployment supervising. Tushar Krishna contributed to...
idea discussion, and paper reviewing, provided compute resources, and supervised FHE training techniques. We want to thank Guanghui Wang for his help in the theory construction of the proposed optimization methods. This work is funded in part by Semiconductor Research Corporation as part of Jump 2.0.

Authors Contribution. Jingtian Dang and Jianming Tong contributed to the experiments and prepared all figures. Jianming Tong, Jingtian Dang, and Anupam Golder proposed ideas and wrote the main manuscript text. Arijit Raychowdhury contributed to FHE theory supervising. Callie Hao contributed to real-device deployment supervising. Tushar Krishna contributed to idea discussion, paper reviewing, and FHE training techniques supervising.

Conflict of Interest. All (Georgia Tech), Sergi Abadal (UPC), Abhishek Bhattacharjee (Yale University), Brad Beckmann (AMD Research), Joel Emer (NVIDIA/MIT), Paul Gratz (Texas AM), Natalie Enright Jerger (University of Toronto), Nachiket Kapre (University of Waterloo), Mikko Lipasti (Univ of Wisconsin, Madison), Joshua San Miguel (Univ of Wisconsin, Madison), Angshuman Parashar (NVIDIA), Michael Pellauer (NVIDIA), Li-Shiuan Peh (NUS), Sree Subramoney (Intel), Paul Whatmough (Qualcomm Research), Jieming Yin (AMD Research), Yuhao Zhu (Univ of Rochester), Hyoukjun Kwon (UC Irvine)

References


