ACCURATE LOW-DEGREE POLYNOMIAL APPROXIMATION OF NON-POLYNOMIAL OPERATORS FOR FAST PRIVATE INFERENCE IN HOMOMORPHIC ENCRYPTION

Jianming Tong [*1] **Jingtian Dang** [*2] **Anupam Golder** [3] **Arijit Raychowdhury** [3] **Cong Hao** [3] **Tushar Krishna** [3]

ABSTRACT

As machine learning (ML) permeates fields like healthcare, facial recognition, and blockchain, the need to protect sensitive data intensifies. Fully Homomorphic Encryption (FHE) allows inference on encrypted data, preserving the privacy of both data and the ML model. However, it slows down non-secure inference by up to five magnitudes, with a root cause of replacing non-polynomial operators (ReLU and MaxPooling) with high-degree Polynomial Approximated Function (PAF). We propose **SMART-PAF**, a framework to replace non-polynomial operators with *low-degree PAF* and then recover the accuracy of PAF-approximated model through four techniques: (1) Coefficient Tuning (CT) – adjust PAF coefficients based on the input distributions before training, (2) Progressive Approximation (PA) – progressively replace one non-polynomial operator at a time followed by a fine-tuning, (3) Alternate Training (AT) – alternate the training between PAFs and other linear operators in the decoupled manner, and (4) Dynamic Scale (DS) / Static Scale (SS) – dynamically scale PAF input value within $(-1, 1)$ in training, and fix the scale as the running max value in FHE deployment. The synergistic effect of CT, PA, AT, and DS/SS enables **SMART-PAF** to enhance the accuracy of the various models approximated by PAFs with various low degrees under multiple datasets. For ResNet-18 under ImageNet-1k, the Pareto-frontier spotted by **SMART-PAF** in latency-accuracy tradeoff space achieves $1.42\times \sim 13.64\times$ accuracy improvement and $6.79\times \sim 14.9\times$ speedup than prior works. Further, **SMART-PAF** enables a 14-degree PAF ($f_1^2 \circ g_1^2$) to achieve $7.81\times$ speedup compared to the 27-degree PAF obtained by minimax approximation with the same 69.4% post-replacement accuracy. Our code is available at https://github.com/TorchFHE/SmartPAF

1 INTRODUCTION

As ML becomes more pervasive in fields such as healthcare (Mateen et al., 2020), facial recognition (Raji & Fried, 2021), and blockchain (Zhang et al., 2021), concerns regarding privacy leakage of private and sensitive data have arisen. Fully Homomorphic Encryption (FHE) (Albrecht et al., 2021) 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, i.e. five orders of magnitude longer than the corresponding non-secure version (Samardzic et al., 2021). This slowdown is primarily due to non-polynomial operators (e.g. ReLU, MaxPooling, and Softmax, etc.), which dominate approximately half of the total latency (Park et al., 2022). 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 FHE due to the lack of native support. To overcome this, previous research has explored (1) a hybrid scheme, which offloads non-polynomial operators to other secure schemes with a secure data transfer to communicate data among schemes, and (2) approximation, which replaces non-polynomial operators with polynomial approximation functions (PAF).

The hybrid scheme is challenging in practice because a plethora of prior arts illustrate the prohibitive communication overheads for transferring data securely among different schemes (Gilad-Bachrach et al., 2016; Lou et al., 2021; Ran et al., 2022).

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

[*]Equal contribution [1]School of Computer Science, Georgia Institute of Technology, North Avenue, Atlanta, 30332, Georgia, U.S. [2]Electrical and Electronics Engineering, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, 15213, Pennsylvania, U.S. [3]School of Electrical and Computer Engineering, Georgia Institute of Technology, North Avenue, Atlanta, 30332, Georgia, U.S.. Correspondence to: Jingtian Dang <dangjingtian@cmu.edu>, Jianming Tong <jianming.tong@gatech.edu>.

Proceedings of the 5th MLSys Conference, Santa Clara, CA, USA, 2024.

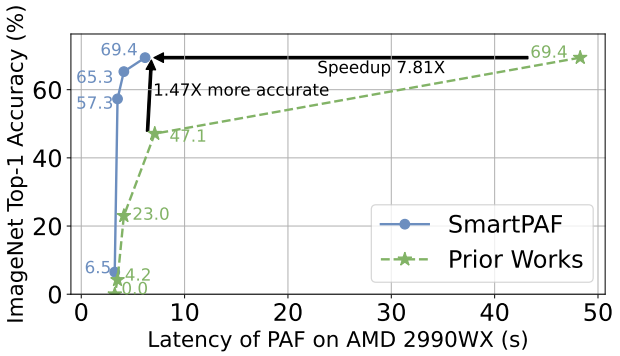

Figure 1: **SMART-PAF** replaces ReLU and MaxPooling by low-degree PAFs, and achieves better accuracy-latency Pareto-frontier than prior works (Lee et al., 2021; 2022; Cheon et al., 2020) on ResNet-18 (ImageNet-1k).

with bootstrapping. An example includes a SotA 27-degree PAF (Lee et al., 2021; Kim et al., 2022). 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 dominating communication latency overheads in practice (Mishra et al., 2020a; Lou et al., 2021; Ran et al., 2022). These approaches are suboptimal and highlight the need for exploring alternative approaches.

To address this challenge, previous research has explored coefficients fine-tuning to reduce accuracy drop when replacing non-polynomial operators with low-latency low-degree PAFs. 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 for a simple 7-layer CNN model under CiFar-10 dataset (Lou et al., 2021), let alone deeper ResNet-18 model under more complex task like ImageNet-1k. To explore the optimal degree with minimal accuracy degradation for various models under different tasks, 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 techniques and a framework to enable the convergence of the PAF-approximated model when using PAFs with arbitrary degrees. This is the first-ever framework, to the best of our knowledge, to (1) replace all non-polynomial operators with $8 \sim 14$-degree PAFs, and (2) to adopt proposed training techniques to minimize accuracy degradation. Out contributions are listed as follows. (The comparison with prior arts is shown in Tab. 1)

Pre-Training Novel Techniques:

- **Coefficient Tuning (CT):** Using a uniform PAF to replace all non-polynomial layers neglects variations in input distributions, causing a marked drop in accuracy of PAF-approximated model (Tab. 3). CT refines PAF coefficients based on local input distributions, enhancing the accuracy of PAF-approximated model without fine-tuning by $1.04\times \sim 2.38\times$ across PAFs with various degrees.
- **Progressive Approximation (PA):** Directly replacing all non-polynomial operators and training the full network, as adopted by previous works, lacks a theoretical convergence guarantee under SGD (Fig. 9). Instead, PA sequentially replaces non-polynomial operators in inference order, with each substitution followed by training layers preceding the replacement point. The theoretical convergence of PA under SGD is analyzed in §3.1, resulting in an accuracy boost of $0.4\% \sim 1.9\%$.

In-Training Novel Techniques:

- **Alternate Training (AT):** When training PAF and linear operator (like convolution) in a PAF-approximated model together, issues arise: slow convergence with a small learning rate or divergence with a large one. This stems from the different hyperparameter needs of PAF coefficients and linear layers training. Therefore, AT trains PAF coefficients and other layers separately, using different hyperparameters, and alternates between PAF coefficients training and other layers training. This leads to accuracy climbing by $0.6\% \sim 2.3\%$ (Tab. 3).
- **Dynamic Scaling (DS) and Static Scaling (SS):** PAF training is sensitive to the input value range. Prior works take a fixed scale value to reduce the input range in training because FHE does not support value-dependent operators. However, input values in training might overflow under a fixed scale, hampering fine-tuning accuracy and potentially leading to divergence. We address this by introducing Dynamic Scaling, normalizing inputs to the $[-1, 1]$ range during training. Subsequently, we apply Static Scaling to the post-training model, anchoring the scale to the peak value in FHE deployment.

SMART-PAF Framework:

- The order of applying proposed techniques affects final accuracy, and thus a systematic scheduling framework is proposed to automatically perform non-polynomial replacement and PAF-approximated model training. Our evaluation results show that for ResNet-18 (ImageNet-1k) and VGG-19 (CiFar-10), **SMART-PAF** consistently enables low-degree PAFs to demonstrate higher accuracy than high-degree SotA PAFs (Tab. 3 and Tab. 4). For example, **SMART-PAF** identifies the Pareto-frontier

Table 1: Comparison of **SMART-PAF** with prior works (Lou et al., 2021; Ran et al., 2022; Gilad-Bachrach et al., 2016; Hesamifard et al., 2017; Lee et al., 2021; Xie et al., 2022; Samardzic et al., 2021; Kim et al., 2022; Samardzic et al., 2022; Reagen et al., 2021; Juvekar et al., 2018b; Mishra et al., 2020a; Riazi et al., 2020; Lou & Jiang, 2019).

	Low Communication Overhead	Low Accuracy Degradation	Low Latency Overhead
SafeNet, CryptoGCN	✗	✗	✓
CryptoNet, CryptoDL, LoLa, CHE	✗	✗	✓
F1, CraterLake, BTS	✓	✓	✗
HEAX, Delphi, Gazelle, Cheetah	✗	✗	✓
SHE	✓	✓	✗
SMART-PAF	✓	✓	✓

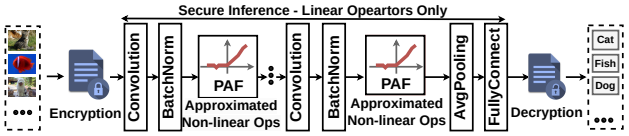

Figure 2: Overview of the FHE-base ML inference where original non-polynomial operators are replaced by Polynomial Approximated Activation (PAF).

of various PAFs shown in Fig. 1 and spots the sweet-point PAF with 14 degrees, which can achieve the same 69.4% validation accuracy with 7.81× latency speedup in ResNet-18 inference on ImageNet-1k, compared to the minimax approximation with 27-degree PAF (Lee et al., 2021).

This work pushes the state-of-the-art in PAF-approximated model training for higher accuracy with lower latency (Fig. 1). First, we formalize PAF-approximated model training into an optimization problem (§3.1). Second, §4 elaborates on the **SMART-PAF** techniques and framework with both intuitive and theoretical analysis. Third, we individually assess the **SMART-PAF** techniques for VGG-19 (CiFar-10) and ResNet-18 (ImageNet-1k) in §5.

2 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) (Cheon et al., 2016) 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 (Riazi et al., 2020). Under the CKKS scheme, only polynomial operators are allowed as shown in Fig. 2 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 (Lou et al., 2021).

2.2 Polynomial Approximated Function (PAF)

High-degree polynomials are theoretically capable of approximating arbitrary functions, but the approximation of non-polynomial operators like ReLU or MaxPooling using PAFs can result in severe approximation errors (Lee et al., 2022). Prior research (Lee et al., 2022) has shown that it is more effective to approximate the $sign(x)$ function [1] and then construct the ReLU and Max operator using it, such as with $\frac{(x+sign(x)\cdot x)}{2}$ and $\frac{(x+y)+(x-y)\cdot sign(x-y)}{2}$.

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 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.

We adopt cascaded polynomial based PAFs throughout the paper because they achieve lower approximation error than a single polynomial with the same degree (Lee et al., 2022; 2021). 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. 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 potentially 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.

Prior works (Lee et al., 2021; Lou et al., 2021) adopted a fixed scale in training because FHE does not support value-dependent operation. Specifically, (Lee et al., 2021) determines the fixed scale by adding some margin to the

[1] $sign(x)$ outputs $1/-1$ if x is positive/negative, and 0 for zero

Table 2: PAF and corresponding multiplication depth. α indicates precision parameter (Lee et al., 2021), and f_i, g_i refer to PAF base in (Cheon et al., 2020)

Form	$\alpha=10$	$f_1^2\circ g_1^2$	$\alpha=7$	$f_2\circ g_3$	$f_2\circ g_2$	$f_1\circ g_2$
Degree	27	14	12	12	10	5
Multiplication Depth	10	8	6	6	6	5

maximum of input values. However, input values in training might overflow the range specified by a fixed scale, hampering fine-tuning accuracy and potentially leading to divergence. Therefore, we propose dynamic-scale training and only pick a fixed scale in model deployment.

3 Problem Statement

3.1 Polynomial Approximation Problem Statement

The essential problem for approximating non-polynomial operators with PAF is the regression problem shown in Eq. 2. Specifically, the goal is to find a vector of coefficients $\mathbf{a}_i = \{a_i^0, \cdots, a_i^N\}$ for a N-degree PAF, such that the cumulative error of replacing non-polynomial function $R(\mathbf{x}_i)$ by a N-degree PAF is minimal given input data as $\mathbf{x}_i$ $(i = 0, \cdots, D-1)$, where D stands for the total number of non-polynomial layers we use non-polynomial layers and non-polynomial operators interchangeably. (Boyd et al., 2004).

$$\min_{\mathbf{a}_0,\cdots,\mathbf{a}_{D-1}} f(\mathbf{a},\mathbf{x}) = \sum_{i=0}^{D-1}\left[\frac{1}{N}\sum_{i=0}^{N}(R(\mathbf{x}_i,\mathbf{a}_0,\cdots,\mathbf{a}_i)-\mathbf{a}_i\cdot\mathbf{x}_i)^2\right] \quad (1)$$

where $\mathbf{x}_i = \{x_i^0, x_i^1, \cdots, x_i^N\}$. The subscript / superscript of x and a indicates layer index / degree index. On the contrary, the subscript / superscript for $\mathbf{a}$ and $\mathbf{x}$ indicates layer index / the training epoch. After replacing $0, \cdots, (i-2)$-th ReLU by PAFs, the input of the i-th ReLU $\mathbf{x_i}$ becomes dependent on $\mathbf{a}_0, \cdots, \mathbf{a}_{i-1}$, i.e. i-th ReLU becomes $R(\mathbf{x}_i, \mathbf{a}_0, \cdots, \mathbf{a}_i)$ instead of $R(\mathbf{x}_i)$.

The Eq. 1 is non-convex as it optimizes all D vectors of coefficients $\mathbf{a_i}$, $i \in [0, D)$ under the changing $\mathbf{x}$.

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

The Eq. 2 only optimizes a single layer, such that $\mathbf{x}_i$ statistically is fixed, which makes it convex. In such cases, the stochastic gradient descent could guarantee finding the optimal solution $\mathbf{a}^*$ given initial value as $\mathbf{a^0}$. The error could converge in the speed order as shown in Eq. 3 after training for T epochs (Boyd et al., 2004). And a good initialization value of $\mathbf{a}$ could reduce the total error and thus improve the convergence speed.

$$f(\mathbf{a^T}) - f(\mathbf{a}^*) = O\left(\frac{||\mathbf{a^0}-\mathbf{a}^*||^2}{\sqrt{T}}\right) \quad (3)$$

where $\mathbf{a^T}$ indicates the value of $\mathbf{a}$ after training for T epochs.

3.2 Post-replacement Model Retraining

The PAF-approximated model 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 the changes in input data distributions.

4 SMART-PAF Techniques

4.1 Overview

Training methods adopted by prior arts lead to 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 analyze critical reasons behind the divergence of existing training algorithms and propose corresponding four techniques to guarantee training convergence while replacing non-polynomial operators with PAFs of arbitrary degrees. Then we propose a framework to schedule techniques automatically.

4.2 Coefficient Tuning (CT)

Eq. 3 indicates that a good initialization can reduce the overall training time, facilitating faster convergence, which has been ignored by prior arts (Lee et al., 2021; Cheon et al., 2020; Lee et al., 2022; Lou et al., 2021). These studies have typically initialized PAFs with the same coefficients using traditional regression algorithms. Such initialization is suboptimal because it overlooks the distribution differences of data at different non-polynomial operators.

Therefore, we propose coefficient tuning (CT) as a technique to use profiled data distribution [2] to obtain a closer-to-original initialization point and then use PAF with different initialization points to replace different non-polynomial operators, as shown in Fig. 3. The CT 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

[2] We assume that distribution probability range is consistent across input samples.

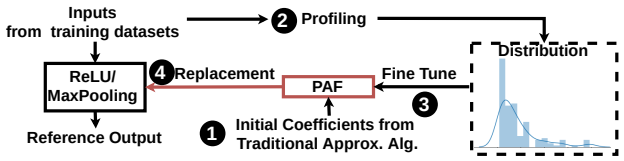

Figure 3: Coefficient Tuning (CT) uses profiled distribution to tune PAF coefficients to generate more accurate results on a reduced smaller input range (high-probability range in profiled data distribution), leading to closer-to-original initialization, higher accuracy, and lower training time

the initial value of coefficients $\mathbf{a}^0$ are closer to the optimal value $\mathbf{a}^*$ as indicated by Eq. 3.

4.3 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 all layers behind the replacement point, making the regression (Eq. 2) non-convex by varying both PAF coefficients $\mathbf{a}_i$ and input data $\mathbf{x}_i$ ($i \in [0, D)$) for all D non-polynomial layers, causing training divergence. Instead, we propose the Progressive Approximation (PA) approach, which replaces the non-polynomial layer with PAF, one layer at a time, followed by fine-tuning of PAF coefficients until accuracy convergence as shown in Fig. 4. In such cases, all $\mathbf{x}_i$ ($i \in [0, D)$) statistically get fixed because we only vary PAF coefficients of a single layer, which is one of all $(\mathbf{a}_0, \mathbf{a}_1, \cdots, \mathbf{a}_{D-1})$, ensuring a simple convex regression problem shown in Eq. 2, which is easily optimizable by SGD.

For example in Fig. 4, PA starts with replacing the first ReLU with a PAF, and then fine-tunes PAF coefficients until accuracy converges. Then PA repeats the same flow for the following non-polynomial operators.

Intuitively, the replacement of all non-polynomial operators by PAFs simultaneously introduces a huge deviation to the original model which is hard for the training algorithm to recover, while PA applies the overall approximation error progressively instead of applying all errors at once, restricting the approximation error to the optimizable range of the training algorithm to enable convergence and effectively mitigate the accuracy degradation.

4.4 Alternate Training (AT)

Replacing the non-polynomial operators with PAFs requires model fine-tuning, as pretrained model parameters are no longer optimal for the new PAF-approximated model. However, multiple previous works fine-tune the PAF-approximated models as a whole to optimize both PAF coefficients and the parameters of other layers (like Convolution, BatchNorm, etc.). Such a training scheme, however, deteriorates accuracy, indicating a failure of convergence. We demonstrate it later in Fig. 9. This is because modifications to PAF coefficients can have a significant impact on final inference results, unlike changes to convolution weights, which may not affect overall results at all. Such an observation 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.

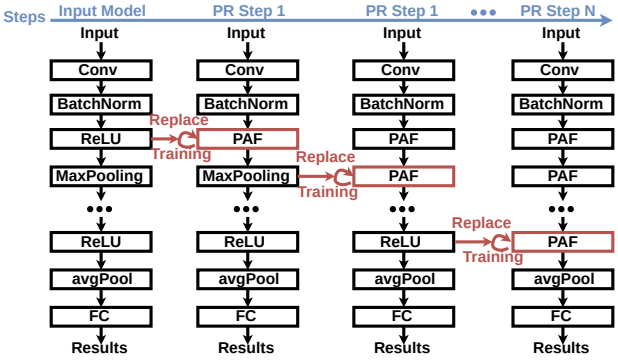

Figure 4: Progressive Approximation (PA) progressively replaces non-polynomial operators, one layer at a time followed by coefficients fine-tuning, to guarantee an SGD-optimizable convex regression problem shown in Eq. 2, enabling training convergence for replacing the targeted model with PAFs of arbitrary degrees.

To tackle training divergence, PAF coefficients fine-tuning should be decoupled from fine-tuning parameters of other layers, and two fine-tuning processes should use different training hyperparameters. We thus propose Alternate Training (AT) to fine-tune PAF coefficients and parameters of other layers separately in an alternate manner. Specifically, AT fine-tunes PAF coefficients first while keeping other parameters fixed, as shown in AT step 1 in Fig. 5. After reaching a specific epoch threshold, PAF coefficients get frozen and AT fine-tunes parameters of other layers in AT step 2. Note that the training in different AT steps may use different hyperparameters because of different parameter sensitivity in different layers. AT repeats until the accuracy finally converges, often resulting in an accuracy climb.

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

4.5 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 (Lee et al., 2021). However, we contend that this approach is suboptimal because it re-

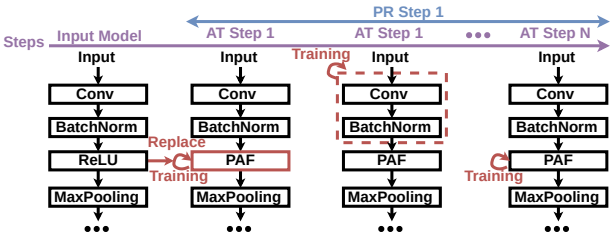

Figure 5: Alternate Training (AT) fine-tunes PAF coefficients and parameters of other layers separately in an alternate manner.

sults in high approximation error across the entire input range, which can impede training convergence and lead to significant accuracy degradation (e.g., accuracy drops from 69% to 10% for ResNet-18 on ImageNet-1k).

To tackle this issue, we introduce Dynamic Scaling (DS) and add an auxiliary layer before each PAF to normalize the value range of its input data to $[-1,1]$ during fine-tuning. DS determines the scale on a batch-by-batch basis. For each batch, the scale is set to the highest absolute input value, and then all data are proportionally scaled to fit within the $[-1,1]$ range. This ensures that the inputs within each batch spread the $[-1,1]$ range, maximizing their relative differences for better distinguishment while staying within bounds.

However, the post-training model is intended for FHE deployment, where Dynamic Scaling (DS) is inapplicable because FHE cannot select the batch's maximal value due to the absence of value-dependent operators. Therefore, we propose Static Scaling (SS) to fix the scale of each auxiliary layer in *PAF-approximated model* as the input running maximum under the training dataset Such a scale determination relies on the similarity between training data and validation data. In our evaluation, either a higher or smaller scale results in lower accuracy for both VGG-19 (CiFar-10 dataset) and ResNet-18 (ImageNet-1k dataset).

4.6 SMART-PAF Framework

The sequence in which CT, PA, AT, and DS/SS are applied can greatly affect the final validation accuracy. Furthermore, employing existing techniques, such as dropout for overfitting mitigation and Stochastic Weights Averaging (SWA) for faster convergence, is crucial.

To navigate these complexities, we introduce a scheduler in **SMART-PAF**. This scheduler systematically applies training configurations and proposed techniques, organizing the training into steps, with each step replacing a single non-polynomial layer with PAF. Coefficient Tuning is applied offline before all steps. The sequence within a single step is outlined below:

- *Training Group:* A training group first trains the PAF-approximated model for E epochs, followed by applying SWA using weights of all E epochs. The model with the highest validation accuracy proceeds to the following procedures.
- *Accuracy Improvement Detection:* If the validation accuracy gets improved during the process, the framework launches a new training group until no accuracy improvement is observed.
- *Overfitting Avoidance:* The framework applies Dropout followed by a new training group if spots overfitting [3].
- *Alternative Training:* When previous techniques do not improve accuracy anymore, AT is applied to swap training targets between PAF and other layers, followed by a new training group.
- *Step Termination Condition:* Current step terminates when observing no accuracy improvement in above processes.

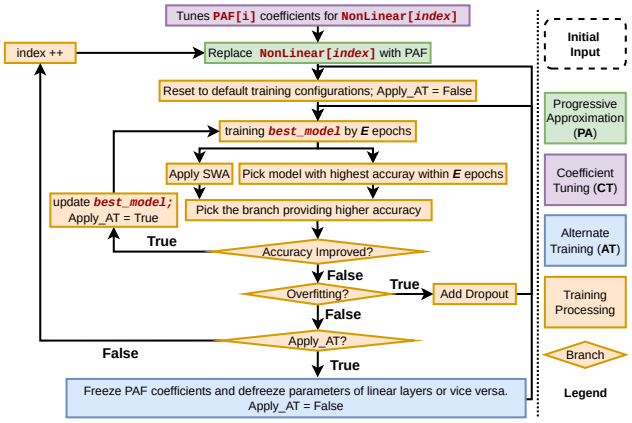

Figure 6: Overview of **SMART-PAF** framework, which automatically schedules **SMART-PAF** techniques including CT, PA, AT, and existing techniques including Dropout and Stochastic Wegiths Averaging (SWA).

SMART-PAF framework dedicates one step for each non-polynomial layer, orchestrating these steps progressively based on the inference order (Progressive Approximation). Dynamic Scaling is adopted in the entire fine-tuning process, while Static Scaling is applied for the post-finetuning model.

5 Evaluation

5.1 Evaluation Setup

Model We evaluated **SMART-PAF** with models being used by prior works for fair comparison (Lee et al., 2021; Lou et al., 2021), including VGG-19 (18 ReLU and 5 MaxPooling), and ResNet-18 (17 ReLU and 1 MaxPooling).

Datasets We adopt CiFar-10 and ImageNet-1k (He et al.,

[3] we adopt the empirical overfitting condition, which is "training accuracy > validation accuracy + 10%"

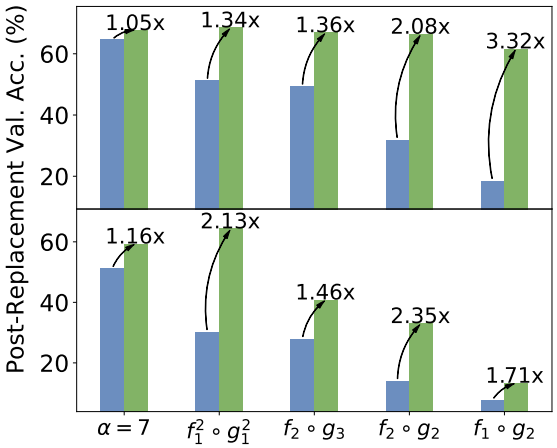

Figure 7: Comparison of ResNet-18 (ImageNet-1k) validation accuracy w/o Fine-Tuning between Coefficient Tuning (CT, green bar) and baseline (blue bar). Top: only replace ReLU by PAF. Bottom: replace all ReLU and MaxPooling.

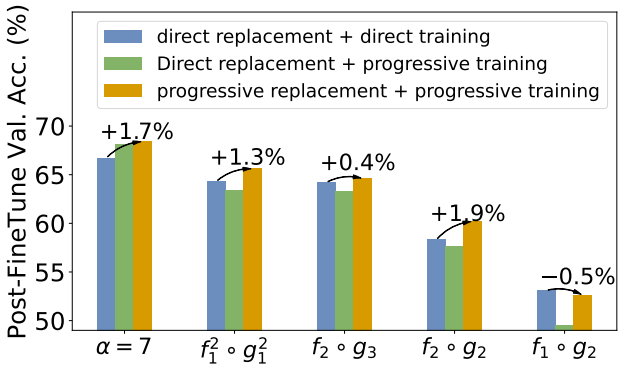

Figure 8: Comparison of ResNet-18 (ImageNet-1k) validation accuracy w/ Fine-Tuning between Progressive Approximation (PA, orange) and baseline (blue). Green bar only adopts progressive training w/o progressive replacement and suffers from severe accuracy degradation.

2015; Krizhevsky et al., 2012) to test the generality of proposed techniques for tasks with different complexity.

PAF Form We adopt 6 PAFs (Tab. 2) with minimal multiplication depth under different degrees as the approximation of $sign(x)$. ReLU and Max operators are replaced by $\frac{(x+sign(x)\cdot x)}{2}$ and $\frac{(x+y)+(x-y)\cdot sign(x-y)}{2}$, separately.

Latency Evaluation We implement PAF in Microsoft SEAL library (Chen et al., 2017) using CKKS (Cheon et al., 2016) (Degree: 32768, modulus bitwidth: 881) and evaluate the wall-clock PAF latency on AMD Threadripper 2990WX.

Accuracy Evaluation The accuracy is obtained through evaluating PAF-approximated models under given datasets.

Training Hyperparameters We adopt different training hyperparameters for training PAF coefficients and parameters of other layers, as shown in Tab. 5. We set the epoch $E = 20$ for each training group. A smaller epoch leads to negligible accuracy changes while a larger epoch leads to long latency of SWA which slows down the training processing.

5.2 Coefficient Tuning (CT) Evaluation

CT adjusts PAF coefficients based on profiled value distributions and improve the $1.05 \sim 3.32\times$ post-replacement validation accuracy w/o fine tuning, as shown in Fig. 7.

When only approximating ReLU, CT shows more benefits 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.

Further, approximating both ReLU and MaxPooling leads to $10.9\% \sim 21\%$ accuracy drop than approximating ReLU only, as shown by comparing the blue bar in the top and bottom figures in Fig. 7. Contrary to the ReLU approximations in earlier works (Park et al., 2022), this suggests that solely assessing ReLU does not provide a full picture of PAF's non-polynomial approximation prowess. Comprehensive evaluations should encompass both ReLU and MaxPooling, wherein CT consistently aids in accuracy restoration, improving up to 34.1% accuracy for $f_1^2 \circ g_1^2$!

5.3 Progressive Approximation Evaluation

During fine-tuning, the baseline (prior works) uses a direct replacement + direct training approach: it replaces all non-polynomial operators with PAFs and trains other layers, excluding the PAFs. In contrast, PA adopts a step-wise strategy. In each step, it replaces one non-polynomial operator with a PAF and trains preceding layers progressively (termed progressive replacement and progressive training). This method yields an accuracy improvement ranging from $0.4 \sim 1.7\%$, as illustrated in Fig. 8. Significantly, the progressive replacement is pivotal for this improvement. This is evident from the marked accuracy degradation of the green bar (direct replacement + progressive training) relative to the orange bar. This is because progressive replacement simplifies the optimization, enabling its theoretical convergence under SGD, as discussed in §3.1.

While PA doesn't always ensure superior accuracy—for instance, the baseline training (blue) outperforms PA in the $f_1 \circ g_2$ setup—it can enhance post-fine-tuning accuracy beyond the limits of higher-degree PAFs shown in Fig. 8. Consequently, the **SMART-PAF** framework divides training

Table 3: Ablation study under of ResNet-18/VGG-19 models under CiFar-10/ImageNet-1k datasets. CT: Coefficient Tuning; PA: Progressive Approximation; AT: Alternate Training; DS: Dynamic Scaling; SS: Static Scaling. DS is only used in fine-tuning to improve accuracy and must converted to SS to be adopted in Homomorphic Encryption (HE) because DS contains value-dependent operators that are not supported by HE. Best HE-compatible accuracies are colored in grey while results with the best validation accuracy are bolded. An accuracy of 0% indicates training divergence.

Model-Dataset	Technique Setup	$f_1^2 \circ g_1^2$	$\alpha = 7$	$f_2 \circ g_3$	$f_2 \circ g_2$	$f_1 \circ g_2$
Replace ReLU ResNet-18 ImageNet-1k Original Accuracy 69.3%	baseline + DS w/o fine tune	51.30%	64.70%	49.40%	32.00%	18.60%
	baseline + CT + DS w/o fine tune	68.60%	67.70%	67.00%	66.50%	61.70%
	baseline + DS	64.30%	66.70%	64.20%	58.30%	53.10%
	baseline + AT + DS	65.20%	68.30%	63.70%	60.50%	52.00%
	baseline + PA + DS	65.60%	**68.40%**	64.60%	60.20%	52.60%
	baseline + PA + AT + DS	64.90%	67.40%	64.60%	56.50%	47.10%
	baseline + CT + PA + DS	68.20%	67.00%	**67.60%**	65.90%	60.80%
	baseline + CT + PA + AT + DS	**69.00%**	68.10%	61.40%	**66.50%**	**63.10%**
	Accuracy Improvement over "baseline + DS"	+4.7%(1.07×)	+1.7%(1.03×)	+3.4%(1.05×)	+8.2%(1.14×)	+10%(1.19×)
Replace all non-polynomial ResNet-18 ImageNet-1k Original Accuracy 69.3%	baseline + DS w/o fine tune	30.3%	51.2%	28%	14.1%	7.8%
	baseline + CT + DS w/o fine tune	64.4%	59.4%	40.9%	33.1%	13.3%
	baseline + DS	59.6%	66.2%	62%	49%	37%
	baseline + SS (**prior work (Lee et al., 2022)**)	25.5%	47.1%	23%	4.2%	0%
	baseline + CT + PA + AT + DS	**69.9%**	**68%**	**65.7%**	**64.1%**	**57.8%**
	SMART-PAF: baseline + CT + PA + AT + SS	69.4%	67%	65.3%	57.3%	6.5%
	Accuracy Improvement over (Lee et al., 2022)	+43.9%(2.72×)	+19.9%(1.42×)	+42.3%(2.84×)	+53.1%(13.64×)	+6.5%(∞)
Replace all non-polynomial VGG-19 CiFar-10 Original Accuracy 93.95%	baseline + DS	93.4%	92.38%	89.87%	89.87%	86.57%
	baseline + SS (**prior work (Lee et al., 2022)**)	91.06%	81.35%	76.58%	58.11%	43.84%
	baseline + CT + DS	93.39%	93.6%	93.3%	**92.4%**	**91.53%**
	baseline + CT + PA + AT + DS	**93.6%**	**93.81%**	**93.59%**	91.49%	91.51%
	SMART-PAF: baseline + CT + PA + AT + SS	92.16%	92.62%	91.51%	88.45%	76.93%
	Accuracy Improvement over (Lee et al., 2022)	+1.1%(1.01×)	+11.27%(1.14×)	+14.93%(1.2×)	+30.34%(1.52×)	+33.09%(1.75×)

into smaller steps, proceeding with the best results between PA and baseline training, as depicted in Fig. 6.

5.4 Ablation Study

5.4.1 Configuration Setup

To investigate the effectiveness of different combinations of proposed techniques, we conduct an ablation study with results of VGG-19 (CiFar-10) ResNet-18 (ImageNet-1k) being presented in Tab. 3. We separate ReLU from MaxPooling to show the approximation effects of different types of non-polynomial operators.

- **Baseline + SS:** We take (Lee et al., 2021; 2022; Cheon et al., 2020) as the baseline. We obtain PAF coefficients through deterministic methods and uses one PAF to replace $sign(x)$ in all non-polynomial operators. Further, a fixed scale is chosen which falls into the category of Static Scaling. For a fair comparison, we still adopted Dynamic Scaling for the training baseline and converted it to a fixed scale after training. The final training accuracy before SS conversion is noted as "baseline + DS".
- **Baseline + "technique" + DS:** The final training accuracy using "technique" before SS conversion.
- **Baseline + "technique" + SS:** The validation accuracy of FHE-deployable PAF-approximated model.

5.4.2 Impact of Techniques Combinations

When replacing all non-polynomial operators, combining CT, PA, and AT yields the highest validation accuracy in PAF coefficients fine-tuning for ResNet-18 (Imagenet-1k). However, for VGG-19 (CiFar-10), there are outliers ($f_2 \circ g_2$, and $f_1 \circ g_2$) obtaining the highest validation accuracy using *CT* only, as indicated by bolded values lying at different rows in Tab. 3.

Further, when replacing ReLU only, the optimal accuracy point stems from a subset of proposed techniques, e.g. $\alpha = 7$ obtains best training accuracy 68.4% when using *PA* while $f_2 \circ g_3$ achieves 67.6% when using $CT + PA$. This variability inspired the development of step-wise **SMART-PAF** framework, allowing us to test different combinations of proposed methods and select the most effective results for every step.

In short, the full combination of proposed techniques is prone to perform the best under complex tasks. While subsets of techniques might be sufficient for producing good accuracy for simpler tasks.

5.4.3 Impact of Approximating MaxPooling

The MaxPooling operator exhibits greater sensitivity to PAF replacement, evidenced by reduced accuracy in replacing both ReLU and MaxPooling than replacing ReLU only. This sensitivity arises because MaxPooling's single sliding window requires nested PAF calls, leading to the propagation and accumulation of approximation errors.

5.4.4 Impact of Dataset

For VGG-19 under the CiFar-10 dataset, **SMART-PAF** enhances the accuracy of prevailing Pareto-frontier by factors ranging from 1.01× to 1.75×. This improvement jumps to 1.42× up to 13.64× for ResNet-18 under the ImageNet-1k dataset. This distinction is attributed to the inherent com-

Table 4: **SMART-PAF** V.S. (Lee et al., 2021). Results of two PAFs with better accuracy and latency are bolded.

VGG19 under CiFar-10, Original Accuracy 93.95%	**SMART-PAF**					Lee (Lee et al., 2021)
PAF format	$f_1 \circ g_2$	$f_2 \circ g_2$	$f_2 \circ g_3$	$\alpha = 7$	$f_1^2 \circ g_1^2$	$\alpha = 14$
Validation Accuracy (Replace all non-polynomial)	72.24%	86.36%	**91.05%**	91.82%	**92.39%**	90.21%
Accuracy Improvement over (Lee et al., 2021)	-19.97%	-3.85%	**+0.84%**	+1.61%	**+2.08%**	90.21%
ReLU Latency on CPU (ms)	3240.16	3510.82	**4122.58**	7113.35	**6179.18**	48278.84 (baseline)
Speedup over (Lee et al., 2021)	14.90×	13.75×	**11.71**×	6.79×	**7.81**×	1.0

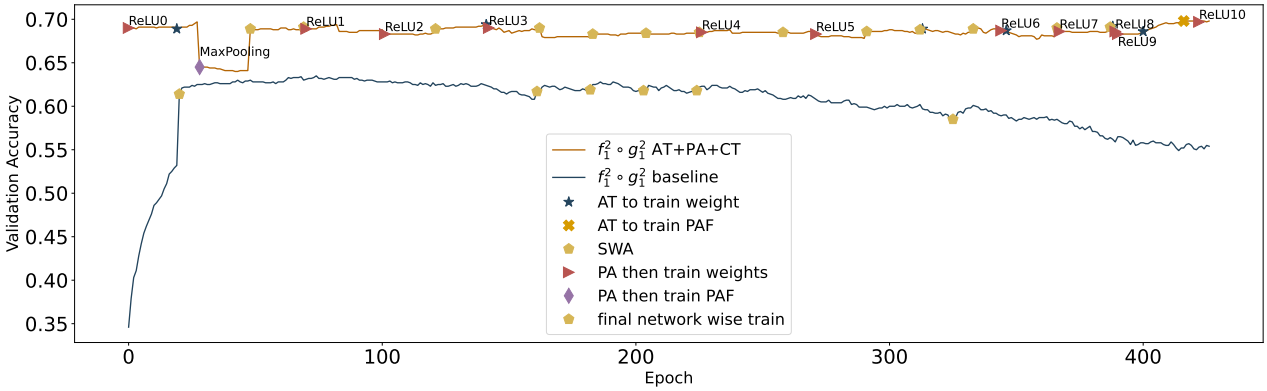

Figure 9: Comparison of training curve (baseline v.s. **SMART-PAF**) ResNet-18 (ImageNet-1k) with ReLU approximated by PAF (14-degree $f_1^2 \circ g_1^2$)

plexity of ImageNet-1k, having 224×224 image size and 1k categories, compared to CiFar-10's 32×32 image size and 10 categories. For instance, replacing with $f_1^2 \circ g_1^2$ results in a 0.55% accuracy dip for VGG-19 on CiFar-10. In contrast, the same PAF leads to a significant 30.3% decline for ResNet-18 on ImageNet-1k, highlighting the crucial role of task-tailored PAF selection in preserving accuracy.

5.5 Comparison with SotA Implementations

5.5.1 *Latency/Accuracy v.s. (Lee et al., 2021)*

We also compare **SMART-PAF** with the 27-degree PAF (Lee et al., 2021) widely used in prior FHE accelerators including F1(Samardzic et al., 2021) and BTS (Kim et al., 2022).

For VGG-19 under CiFar-10 as shown in Tab. 4, **SMART-PAF** identified two higher-accuracy PAFs with 7.81× and 11.71× speedup and 0.84% and 2.08% accuracy improvement.

Considering ResNet-18 under ImageNet-1k, the PAF-approximated model using the 27-degree PAF demonstrates 69.3% top-1 accuracy. In contrast, **SMART-PAF** spotted the 14-degree PAF ($f_1^2 \circ g_1^2$) that achieves 7.81× speedup with 69.4% PAF-approximated validation accuracy. Remarkably, this marginally surpasses original pretrained ResNet-18's accuracy 69.3%, underscoring **SMART-PAF**'s efficacy.

5.5.2 *Accuracy v.s. (Lee et al., 2022; Cheon et al., 2020)*

We contrast **SMART-PAF** with prevailing Pareto-frontier PAFs (Lee et al., 2022) in Tab. 3. When all non-polynomial operators (ReLU and MaxPooling) were substituted, **SMART-PAF** consistently achieves $1.42\times \sim 13.64\times$ accuracy enhancement over (Cheon et al., 2020) on ResNet-18 (ImageNet-1k) and $1.01\times \sim 1.75\times$ under VGG-19 (CiFar-10).

6 Training Processing Deep Dive

In this section, we perform a detailed comparison and analysis of the training curve using both the baseline training strategy and **SMART-PAF** with the 14-degree PAF ($f_1^2 \circ g_1^2$) (Lee et al., 2021) in Tab. 2. Both baseline and **SMART-PAF** train the PAF-approximated model step-by-step and leverage the same dropout and SWA strategy for a fair comparison.

Before the training, the baseline (blue in Fig. 9) replaces all non-polynomial operators with the same PAF, the coefficients of which are determined by regression algorithm (Lee et al., 2021; 2022). By contrast, **SMART-PAF** only replaces the first non-polynomial layer with PAF followed by a Coefficient Tuning to reduce the accuracy drop, leaving other non-polynomial layers to be replaced progressively in the future steps, hence resulting in a 34.1% initial accuracy improvement compared to the baseline as shown in Fig. 9.

During training, the accuracy of baseline drops after an initial boost, Specifically, when the first training step ends ($E = 20$ epochs), Stochastic Weights Averaging (SWA) is applied to smooth the weights update, recovering around 10% accuracy demonstrated by the first yellow pentagon on the blue curve. However, every training step thereafter leads to worse accuracy as shown by the dropping trend of the blue curve, indicating a training failure because of no convergence guarantee under the SGD algorithm.

On the contrary, **SMART-PAF** replaces the ReLU progressively with post-CT PAFs, and validation accuracy increases

for **SMART-PAF** when training the PAF-approximated model under SGD. During the middle of the training, each replacement of non-polynomial layer with PAF causes some accuracy degradation, which is further optimized back through both Stochastic Weight Averaging (SWA) and Alternate Training (AT), as shown in the climbing orange curve after each ReLU replacement (purple diamond) in Fig. 9.

Moreover, AT enhances validation accuracy after replacing ReLU 0, 8, 9, and so on, as indicated by the ascending orange curve after applying AT (marked by an orange cross and dark star). However, AT shows no improvement for ReLU 3 and 6, consistent with findings in Tab. 3. This suggests that AT alone could help accuracy but doesn't guarantee accuracy enhancement for PAF-approximated models.

The detailed comparison illustrates both the invalidity of baseline training methods for PAF-approximated ML models and the effectiveness of **SMART-PAF**.

7 Related Work

ML inference is prone to a severe threat of private information leakage in the cloud. Fully Homomorphic Encryption (FHE) (Albrecht et al., 2021) guarantees privacy by directly enabling computation on the encrypted data.

FHE schemes do not support non-polynomial operators like ReLU or MaxPooling. Therefore, HEAX (Riazi et al., 2020), Delphi (Mishra et al., 2020a), Gazelle (Juvekar et al., 2018b), and Cheetah (Reagen et al., 2021) presented hybrid schemes consisting of FHE, Multi-Party Computation (Rouhani et al., 2017), or Garbled Circuits (GC) (Juvekar et al., 2018a), where they relied on non-FHE scheme to process non-polynomial kernels. However, both MPC and GC are non-practical because of large-size packets in need of transferring among data sources and compute nodes.

Alternatively, (Lee et al., 2021; Gilad-Bachrach et al., 2016; Hesamifard et al., 2017; Brutzkus et al., 2019; Xie et al., 2022; Mishra et al., 2020b) propose replacing all non-polynomial kernels with the same low-degree polynomial approximation and processing it in the FHE domain. However, a 27-degree PAF is used for low accuracy degradation, which consumes a large-portion of overall latency because of the long multiplication chain in FHE accelerators (Samardzic et al., 2021; 2022; Kim et al., 2022).

To further reduce the overhead of non-polynomial kernels, SAFENet (Lou et al., 2021), and CryptoGCN (Ran et al., 2022) 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 training divergence in high-degree polynomials. Till now, there are no systematic training techniques for fine-tuning the model consisting of both linear operators and polynomial approximation functions.

Recently, AESPA (Park et al., 2022) claims a quadratic approximation to replace ReLU with negligible accuracy degradation of VGG and ResNet models under CiFar-10/100 and TinyImageNet. However, the scheme does not show the method to approximate MaxPooling which is more sensitive and hard to approximate than ReLU as we quantified in Tab. 3. Further, the high accuracy preserving capability under TinyImageNet of quadratic does not guarantee low accuracy degradation under complex datasets like ImageNet, as we quantified in §5.4.

8 Conclusion

This paper demonstrates that the training of ML models with non-polynomial 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 typical training methods hinders the exploration of using PAF (higher than 5 degrees) to replace non-polynomial operators for better post-replacement accuracy. This paper proposes four techniques and a training framework to address such a challenge: (1) Coefficient Tuning provides good initialization of PAF coefficients using profiled data distribution for higher post-replacement initial accuracy and faster convergence; (2) Progressive Approximation enables the PAF-approximated model training convergence under SGD algorithm through progressive layer-wise non-polynomial operators replacement and training; (3) Alternate Training separately trains PAF coefficients and layers except PAFs with different hyperparameters to avoid training interference; and (4) Dynamic Scale in training and Static Scale in post-training post-replacement inference under FHE to avoid value overflow. The order of applying techniques affects final accuracy, and **SMART-PAF** framework is thus proposed to automatically apply the above techniques as well as dropout and SWA for accuracy improvement. **SMART-PAF** identifies the optimal Pareto-frontier in the latency-accuracy tradeoff space with $1.42\times \sim 13.64\times$ accuracy improvement and $6.79\times \sim 14.9\times$ speedup for ResNet-18 (ImageNet-1k). Further, within such Pareto-frontier, **SMART-PAF** spots the 14-degree PAF ($f_1^2 \circ g_1^2$) that achieves the same 69.4% post-replacement accuracy with $7.81\times$ latency speedup compared to 27-degree PAF obtained by minimax approximation. The validity and efficacy of **SMART-PAF** are tested by multiple models under different datasets. We believe that **SMART-PAF** potentially opens a new paradigm of systematically augmenting PAF-approximated ML models for accurate and fast private inference.

9 ACKNOWLEDGE

We thank Guanghui Wang for mathematical formulation in §3.1, Hanrui Wang for his valuable feedbacks to improve this paper. This work was supported in part by ACE, one of seven centers in JUMP 2.0, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

REFERENCES

Albrecht, M., Chase, M., Chen, H., Ding, J., Goldwasser, S., Gorbunov, S., Halevi, S., Hoffstein, J., Laine, K., Lauter, K., et al. Homomorphic encryption standard. *Protecting privacy through homomorphic encryption*, pp. 31–62, 2021.

Boyd, S., Boyd, S. P., and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

Brutzkus, A., Gilad-Bachrach, R., and Elisha, O. Low latency privacy preserving inference. In *International Conference on Machine Learning*, pp. 812–821. PMLR, 2019.

Chen, H., Laine, K., and Player, R. Simple encrypted arithmetic library-seal v2. 1. In *International conference on financial cryptography and data security*, pp. 3–18. Springer, 2017.

Cheon, J. H., Kim, A., Kim, M., and Song, Y. Homomorphic encryption for arithmetic of approximate numbers. Cryptology ePrint Archive, Paper 2016/421, 2016. URL https://eprint.iacr.org/2016/421. https://eprint.iacr.org/2016/421.

Cheon, J. H., Kim, D., and Kim, D. Efficient homomorphic comparison methods with optimal complexity. In *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 221–256. Springer, 2020.

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 Balcan, M. F. and Weinberger, K. Q. (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 201–210, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/gilad-bachrach16.html.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Hesamifard, E., Takabi, H., and Ghasemi, M. Cryptodl: Deep neural networks over encrypted data. *ArXiv*, abs/1711.05189, 2017.

Juvekar, C., Vaikuntanathan, V., and Chandrakasan, A. Gazelle: A low latency framework for secure neural network inference. In *Proceedings of the 27th USENIX Conference on Security Symposium*, SEC'18, pp. 1651–1668, USA, 2018a. USENIX Association. ISBN 9781931971461.

Juvekar, C., Vaikuntanathan, V., and Chandrakasan, A. Gazelle: A low latency framework for secure neural network inference, 2018b. URL https://arxiv.org/abs/1801.05507.

Kim, S., Kim, J., Kim, M. J., Jung, W., Kim, J., Rhu, M., and Ahn, J. H. Bts: An accelerator for bootstrappable fully homomorphic encryption. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ISCA '22, pp. 711–725, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450386104. doi: 10.1145/3470496.3527415. URL https://doi.org/10.1145/3470496.3527415.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Proceedings of the Neural Information Processing Systems (NIPS)*, 2012.

Lee, E., Lee, J.-W., No, J.-S., and Kim, Y.-S. Minimax approximation of sign function by composite polynomial for homomorphic comparison. *IEEE Transactions on Dependable and Secure Computing*, 19(6):3711–3727, 2022. doi: 10.1109/TDSC.2021.3105111.

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*, abs/2105.10879, 2021.

Lou, Q. and Jiang, L. *SHE: A Fast and Accurate Deep Neural Network for Encrypted Data*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Lou, Q., Shen, Y., Jin, H., and Jiang, L. {SAFEN}et: A secure, accurate and fast neural network inference. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Cz3dbFm5u-.

Mateen, B. A., Liley, J., Denniston, A. K., Holmes, C. C., and Vollmer, S. J. Improving the quality of machine learning in health applications and clinical research. *Nature Machine Intelligence*, 2(10):554–556, 2020.

Mishra, P., Lehmkuhl, R., Srinivasan, A., Zheng, W., and Popa, R. A. Delphi: A cryptographic inference system for neural networks. In *Proceedings of the 2020 Workshop on Privacy-Preserving Machine Learning in Practice*, PPMLP'20, pp. 27–30, New York, NY, USA, 2020a. Association for Computing Machinery. ISBN 9781450380881. doi: 10.1145/3411501.3419418. URL https://doi.org/10.1145/3411501.3419418.

Mishra, P., Lehmkuhl, R. T., Srinivasan, A., Zheng, W., and Popa, R. A. Delphi: A cryptographic inference service for neural networks. In *IACR Cryptology ePrint Archive*, 2020b.

Park, J., Kim, M. J., Jung, W., and Ahn, J. H. Aespa: Accuracy preserving low-degree polynomial activation for fast private inference, 2022.

Raji, I. D. and Fried, G. About face: A survey of facial recognition evaluation. *CoRR*, abs/2102.00813, 2021. URL https://arxiv.org/abs/2102.00813.

Ran, R., Wang, W., Gang, Q., Yin, J., Xu, N., and Wen, W. CryptoGCN: Fast and scalable homomorphically encrypted graph convolutional network inference. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=VeQBBm1MmTZ.

Reagen, B., Choi, W.-S., Ko, Y., Lee, V. T., Lee, H.-H. S., Wei, G.-Y., and Brooks, D. Cheetah: Optimizing and accelerating homomorphic encryption for private inference. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 26–39, 2021. doi: 10.1109/HPCA51647.2021.00013.

Riazi, M. S., Laine, K., Pelton, B., and Dai, W. Heax: An architecture for computing on encrypted data. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '20, pp. 1295–1309, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371025. doi: 10.1145/3373376.3378523. URL https://doi.org/10.1145/3373376.3378523.

Rouhani, B. D., Riazi, M. S., and Koushanfar, F. Deepsecure: Scalable provably-secure deep learning, 2017. URL https://arxiv.org/abs/1705.08963.

Samardzic, N., Feldmann, A., Krastev, A., Devadas, S., Dreslinski, R., Peikert, C., and Sanchez, D. F1: A fast and programmable accelerator for fully homomorphic encryption. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '21, pp. 238–252, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450385572. doi: 10.1145/3466752.3480070. URL https://doi.org/10.1145/3466752.3480070.

Samardzic, N., Feldmann, A., Krastev, A., Manohar, N., Genise, N., Devadas, S., Eldefrawy, K., Peikert, C., and Sanchez, D. Craterlake: A hardware accelerator for efficient unbounded computation on encrypted data. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ISCA '22, pp. 173–187, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450386104. doi: 10.1145/3470496.3527393. URL https://doi.org/10.1145/3470496.3527393.

Xie, T., Yamana, H., and Mori, T. Che: Channel-wise homomorphic encryption for ciphertext inference in convolutional neural network. *IEEE Access*, 10:107446–107458, 2022. doi: 10.1109/ACCESS.2022.3210134.

Zhang, Y., Wang, S., Zhang, X., Dong, J., Mao, X., Long, F., Wang, C., Zhou, D., Gao, M., and Sun, G. Pipezk: Accelerating zero-knowledge proof with a pipelined architecture. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pp. 416–428, 2021. doi: 10.1109/ISCA52012.2021.00040.

A Training Hyperparameters

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

Table 5: Baseline training parameters, where other layers include Convolution, Linear, and BatchNorm.

Configuration	Value
Replaced layer	ReLU& MaxPooling
Optimizer	Adam
learning rate for PAF	1e-4
learning rate for other layers	1e-5
Weight decay for PAF	0.01
Weight decay for other layers	0.1
BatchNorm Tracking	False
Dropout	False

B Post-Training PAF 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.

B.1 $\alpha = 7$

Mathematical Format: The original format for the Mini-Max approximated polynomial ($\alpha = 7$) (Lee et al., 2021) is shown in Eq. 4. Such a proposed polynomial $p_7(x)$ was used to approximate the $sign(x)$ function, which outputs 1 when x is positive and -1 when x is negative. Then non-polynomial ReLU could be constructed using $\frac{x+x\cdot sign(x)}{2}$.

$$\begin{aligned} p_7(x) &= p_{7,2}(x) \circ p_{7,1}(x) \\ p_{7,1}(x) = \sum_{i=0}^{7} a_i \times x^i \qquad & p_{7,2}(x) = \sum_{i=0}^{7} b_i \times x^i \end{aligned} \tag{4}$$

The odd function nature of $sign(x)$ results in a negligible coefficient 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 Eq. 5.

$$\begin{aligned} p_7(x) &= p_{7,2}^{odd\ only}(x) \circ p_{7,1}^{odd\ only}(x) \\ p_{7,1}^{odd\ only}(x) &= a_1 x + a_3 x^3 + a_5 x^5 + a_7 x^7 \\ p_{7,2}^{odd\ only}(x) &= b_1 x + b_3 x^3 + b_5 x^5 + b_7 x^7 \end{aligned} \tag{5}$$

Table 6: $f_1 \circ g_2$ best coefficients list

layer id	f_2 coefficients		g_3 coefficients		
	c_1	c_3	d_1	d_3	d_5
0	3.064987659	-4.359854698	3.644091129	-7.056697369	4.412326813
1	2.939064741	-3.989520550	3.756805420	-7.105865479	4.209794998
2	2.962512255	-4.095692158	3.725888252	-7.275540352	4.892793179
3	2.996977568	-4.153297901	3.783520699	-7.263069630	4.682956696
4	2.898474693	-4.044208527	3.641639471	-7.243083000	4.771345139
5	2.895201445	-3.905539751	3.689141512	-7.129144192	4.736110687
6	3.018208981	-4.113882542	3.705801964	-7.180747986	4.518863201
7	2.848899364	-3.874762058	3.611979723	-6.771905422	4.524455547
8	3.008141994	-4.087264061	3.836204052	-7.746193886	4.919332504
9	2.968442440	-3.986024141	3.703149557	-7.153123856	4.776097775
10	2.900203228	-3.924145937	3.688660622	-7.306476593	4.663645267
11	2.782385111	-3.684296608	3.651248932	-6.951449394	4.715543270
12	2.958166838	-3.980643034	3.829906940	-7.610838890	4.719619274
13	2.811106443	-3.719117880	3.632898569	-6.837011814	4.688860893
14	2.911352396	-3.886567831	3.674616098	-6.988801003	4.670355797
15	2.796648502	-3.706235886	3.595447540	-6.843948841	4.560091972
16	3.042621136	-3.979726553	3.910200596	-7.521365166	4.733543873

Under such polynomial format, coefficients used in PAF for replacing ReLU at different locations are shown in Tab. 7.

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

The format of different building-block polynomials are shown in Eq. 8.

$$\begin{aligned} f_1(x) &= c_1 \cdot x + c_3 \cdot x^3 \\ g_1(x) &= d_1 \cdot x + d_3 \cdot x^3 \\ f_2(x) &= c_1 \cdot x + c_3 \cdot x^3 + c_5 \cdot x^5 \\ g_2(x) &= d_1 \cdot x + d_3 \cdot x^3 + d_5 \cdot x^5 \\ g_3(x) &= d_1 \cdot x + d_3 \cdot x^3 + d_5 \cdot x^5 + d_7 \cdot x^7 \end{aligned} \tag{6}$$

$f_1^2 \circ g_1^2$ refers to combining both f_1 and g_1 into a composite polynomial in the sequence shown in Eq. 7, with PAF coefficients for at different ReLU layers shown in Tab. 9.

$$f_1^2 \circ g_1^2(x) = g_1^1(g_1^0(f_1^1(f_1^0(x)))) \tag{7}$$

Similarly, coefficients value of $f_2 \circ g_3$, $f_2 \circ g_2$ and $f_1 \circ g_2$ are shown in Tab. 10, Tab. 11 and Tab. 8, separately.

C Multiplication Depth Analysis

CKKS (Cheon-Kim-Kim-Song) is a leveled homomorphic encryption scheme capable of evaluating L-level arithmetic circuits without the need for bootstrapping. Here, L signifies the depth of the arithmetic circuit that can be computed, which is contingent upon the parameters defining the homomorphic context. Every Rescaling or modulus reduction decreases one level to reduce the noise of multiplication operation.

In this context, the "multiplication depth" of a PAF refers to the number of levels reduced during its evaluation. The multiplication depth is also decided by the highest degree term in PAF. To minimize level consumption in PAF multiplication, contemporary methodologies leverage a divide-and-conquer strategy. For instance, for a polynomial featuring a highest degree term of $a \cdot x^n$, the required depth is computed as $\lceil \log_2(n+1) \rceil$. When dealing with composite polynomials, the overall depth necessitated is the aggregate of the depths required for each constituent sub-polynomial.

Taking $f_1 \circ g_2 = g_2(f_1(x))$ as an example. The multiplication depth of each intermediate results are shown in Tab. 8.

$$\begin{aligned} y = f_1(x) &= c_1 \cdot x + c_3 \cdot x^3 \\ g_2(y) &= d_1 \cdot y + d_3 \cdot y^3 + d_5 \cdot y^5 \end{aligned} \tag{8}$$

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

a_1	a_3	a_5	a_7	b_1	b_3	b_5	b_7
7.304451	-34.68258667	59.85965347	-31.87552261	2.400856	-2.631254435	1.549126744	-0.331172943

Table 8: $f_1 \circ g_2$ Multiplication Depth Example

Multiplication Depth	0	1	2	3	4	5
Variables	c_3, x	$c_3 \cdot x, x^2$	$c_3 \cdot x^3, y = f_1(x)$	$d_5 \cdot y, y^2$	y^4	$d_5 \cdot y^5$

Table 9: Coefficients of $f_1^2 \circ g_1^2$ at different layers

	f_1^2 coefficients				g_1^2 coefficients			
layer id	c_1^0	c_3^0	c_1^1	c_3^1	d_1^0	d_3^0	d_1^1	d_3^1
0	2.736806631	-3.864239931	2.115309238	-2.268822908	2.239115477	-2.424801588	2.189934731	-1.481475353
1	2.609737396	-2.629375458	2.115823507	-1.854049206	2.300836086	-2.241225243	2.231765747	-1.455139399
2	2.572752714	-2.620458364	2.008517504	-1.67325747	2.017426491	-1.779745221	2.066540718	-1.300397515
3	2.874353647	-3.49595499	2.073785543	-1.72846055	2.091589212	-1.851963162	2.141039133	-1.372249603
4	2.588399172	-3.086382866	2.01845789	-1.867060781	1.999999881	-1.845559597	2.052644968	-1.279196978
5	2.604569435	-2.614924431	1.93332684	-1.466841698	1.942190886	-1.626866937	2.10518527	-1.243854761
6	2.510973692	-2.517734289	2.132683754	-2.017316103	2.235149622	-2.204242945	2.183528662	-1.424280167
7	2.751836777	-2.765525579	2.021913052	-1.521527886	2.008341789	-1.650658488	2.125827074	-1.320276856
8	2.517604351	-2.519313574	2.131887913	-1.986418962	2.247759819	-2.206320763	2.191907883	-1.425198913
9	2.562408924	-2.520729303	2.110760212	-1.814227581	2.062101603	-1.789000034	2.126989841	-1.338556409
10	2.437770844	-2.398545027	2.016869307	-1.811605096	2.103379965	-1.996958494	2.111694336	-1.30810833
11	2.781474829	-2.742717981	2.02037096	-1.498650432	2.043134928	-1.701895356	2.140466452	-1.345968127
12	2.483508587	-2.447231293	2.057531595	-1.836180925	2.189022541	-2.110060215	2.162631512	-1.370931029
13	2.787295341	-2.709958792	2.00928688	-1.456294537	2.007162809	-1.627877712	2.114115715	-1.327487946
14	2.674963474	-2.604590893	2.028381109	-1.637359142	2.129605532	-1.939982772	2.159248829	-1.392939448
15	2.731667519	-2.661221027	2.026224852	-1.519181132	2.036108494	-1.692675114	2.118255377	-1.338307023
16	2.670770168	-2.607930183	2.119180441	-1.756756186	2.236502171	-2.061469316	2.230870724	-1.45818007

Table 10: $f_2 \circ g_3$ best coefficients list

	f_2 coefficients			g_3 coefficients			
layer id	c_1	c_3	c_5	d_1	d_3	d_5	d_7
0	3.487593412	-6.971315384	2.381806374	4.736026287	-16.16058159	25.20542908	-13.1174
1	3.484929323	-7.034649372	3.685389519	4.983552456	-17.01627541	25.34817886	-12.4504
2	3.312547922	-6.849102974	3.659186125	4.616300583	-15.70791912	25.24704933	-13.7765
3	3.42953968	-7.291306973	3.949234486	4.785545349	-16.25030518	25.22435379	-13.1702
4	3.550015688	-7.992001534	3.389156818	4.644083023	-15.87583256	25.47412872	-13.8047
5	3.484149933	-7.679964066	3.130941153	4.65158844	-15.79552174	25.19073868	-13.6172
6	1.875	-1.25	0.375	4.481445313	-16.18847656	25.01367188	-12.5586
7	3.137469292	-6.013744831	2.900674343	4.600552082	-15.5252409	24.95741463	-13.7303
8	3.355214119	-5.68600893	1.215050697	4.856618881	-16.73614693	25.50185585	-12.7147
9	3.605870724	-9.147006989	6.160003185	4.596205711	-15.64334202	25.45436478	-14.1617
10	3.669521809	-8.906849861	5.65577507	4.712775707	-16.15146828	25.63137817	-13.6679
11	3.432019472	-8.035040855	4.964941978	4.565317631	-15.44346809	25.10269928	-13.9918
12	3.677670956	-8.38080883	4.933722496	4.846800804	-16.69511223	25.66197395	-13.0236
13	3.383493662	-8.223423958	5.385590076	4.52063942	-15.19449425	24.9539814	-14.2344
14	3.32148385	-7.110795498	4.014864445	4.572896957	-15.55243587	25.26078415	-14.0067
15	3.381628513	-7.793000221	4.806651115	4.586762428	-15.50544167	25.14218521	-14.0126
16	3.627621889	-8.305987358	5.061814785	4.829498291	-16.53964996	25.57732391	-13.1699

Table 11: $f_2 \circ g_2$ best coefficients list

	f_2 coefficients			g_3 coefficients		
layer id	c_1	c_3	c_5	d_1	d_3	d_5
0	3.632708073	-8.879578590	4.333632946	3.700465441	-7.351731300	5.071476460
1	3.412810802	-7.752333164	4.516210556	3.855783939	-7.789761543	5.177268505
2	3.355527401	-8.588312149	5.618574142	3.640014887	-7.615984440	5.668038368
3	3.533123493	-9.278223038	6.205972672	3.779361486	-7.770857811	5.565216064
4	1.875000000	-1.250000000	0.375000000	3.255859375	-5.964843750	3.707031250
5	3.421332598	-9.231142044	6.353975773	3.687772274	-7.753697395	5.787805080
6	3.494106293	-8.028047562	3.792766333	3.851673841	-8.117405891	5.920250893
7	3.236023188	-7.844894886	4.858978271	3.662446976	-7.398378849	5.480692863
8	3.308430910	-7.289185524	3.084533691	3.766145468	-8.078896523	5.651748657
9	3.438756227	-9.819555283	7.128154278	3.620871305	-7.664072514	5.793798447
10	3.470819712	-9.487674713	6.564511299	3.746651173	-8.130080223	6.042979240
11	3.344857931	-8.513930321	5.686520100	3.717740774	-7.314604759	5.406781673
12	3.561307669	-9.413117409	6.282663822	3.941442251	-8.642221451	6.365680695
13	3.235330582	-8.009678841	5.256969452	3.645334482	-7.250671864	5.429522514
14	3.269543648	-7.355520248	4.257196426	3.702267408	-7.359237194	5.368722439
15	3.318752050	-8.203745842	5.435956478	3.630973339	-7.331366062	5.393109322
16	3.595479012	-9.167343140	6.192716122	3.955091715	-8.303151131	6.023469925