

Article

CipherAPR: Accelerating RNS-CKKS Encrypted Inference via Importance-Guided and Level-Aware Mixed-Degree Polynomial Design

Junping Wan ¹, Yucen Liao ¹, Yinglong Liao ², Zejiu Tan ¹, Jinming Xu ³, Zoe L. Jiang ^{1,*}  and Binxing Fang ^{4,*}

¹ School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen 518055, China; wanjunping97@stu.hit.edu.cn (J.W.); 24s051006@stu.hit.edu.cn (Y.L.); tanzejiu@stu.hit.edu.cn (Z.T.)

² Department of Optoelectronic Engineering, Jinan University, Guangzhou 510632, China; yinglongliao@stu2023.jnu.edu.cn

³ College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China; jimmyxu@zju.edu.cn

⁴ School of Cyberspace Security, Guangzhou University, Guangzhou 510006, China

* Correspondence: zoeljiang@hit.edu.cn (Z.L.J.); fangbx@cae.cn (B.F.)

Abstract

Fully Homomorphic Encryption (FHE) enables Machine Learning as a Service (MLaaS) providers to perform inference over encrypted data, preserving user privacy. In the RNS-CKKS FHE scheme, however, ReLU activations must be replaced with polynomials. High-degree polynomial approximations preserve accuracy but consume more ciphertext levels, triggering costly bootstrapping operations. Existing mixed-degree methods reduce the bootstrapping count by assigning different polynomial degrees across layers. However, recent FHE compiler research shows that reducing the bootstrapping count alone is insufficient to fully accelerate inference, because bootstrapping placement and the level budget restored after each operation also significantly affect performance. Incorporating such execution-side factors into mixed-degree design substantially enlarges the search space, making straightforward extensions of existing methods computationally infeasible. We propose CipherAPR, an importance-guided framework for level-aware mixed-degree polynomial design. CipherAPR introduces the Low-Magnitude Activation Ratio (LMAR) to prioritize degree updates on accuracy-sensitive layers, combines Domain-Adaptive Approximation (DAAP) with Multi-Objective Coefficient Tuning (MOCT) to produce reusable polynomial approximations that consume fewer ciphertext levels, and applies ciphertext-level utilization (CLU) to filter configurations with poor restored-level utilization. Lightweight latency and accuracy estimators further accelerate offline candidate screening. Experiments on ResNet and VGG show that CipherAPR achieves a $1.09\times$ – $1.39\times$ speedup over AutoFHE with comparable accuracy.

Keywords: encrypted inference; bootstrapping; ciphertext-level utilization; fully homomorphic encryption; RNS-CKKS



Received: 11 May 2026

Revised: 19 June 2026

Accepted: 19 June 2026

Published: 2 July 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Convolutional Neural Networks (CNNs) have demonstrated significant potential in critical applications such as healthcare [1] and finance [2]. For users who wish to utilize advanced model services but lack the necessary computational resources, Machine Learning as a Service (MLaaS) has become a mainstream solution [3]. In the MLaaS paradigm, users upload their local data to cloud servers to request services like model inference. However,

this model carries severe privacy risks, as users are widely concerned about the potential misuse of their sensitive data on the server side. Furthermore, legal frameworks like the General Data Protection Regulation (GDPR) [4] and the California Consumer Privacy Act (CCPA) [5] impose strict limitations on data collection and usage by MLaaS providers. Consequently, there is an urgent need to integrate efficient and compliant data privacy technologies into MLaaS.

In response, researchers have proposed secure inference (SI) techniques that use cryptographic methods to protect user data [6–8]. Prominent among these are Secure Multi-Party Computation (MPC) [9,10] and Fully Homomorphic Encryption (FHE) [11–13]. The core objective of SI is to ensure that the server cannot access the user's raw data or inference results throughout the entire process. Compared to MPC, which requires frequent online interaction, FHE allows for the server to perform computations directly on encrypted data. The user only needs to decrypt the final result locally, offering greater flexibility and practicality for deployment.

Figure 1 illustrates a typical FHE application scenario: A patient encrypts a sensitive medical image and sends it to a remote diagnostic service. The service provider runs a diagnostic model on the ciphertext and returns an encrypted diagnostic result to the patient. Throughout this process, the server has no access to any plaintext medical information, thereby preserving patient privacy. Given this, FHE is widely regarded as one of the most promising privacy-preserving computation technologies [14].

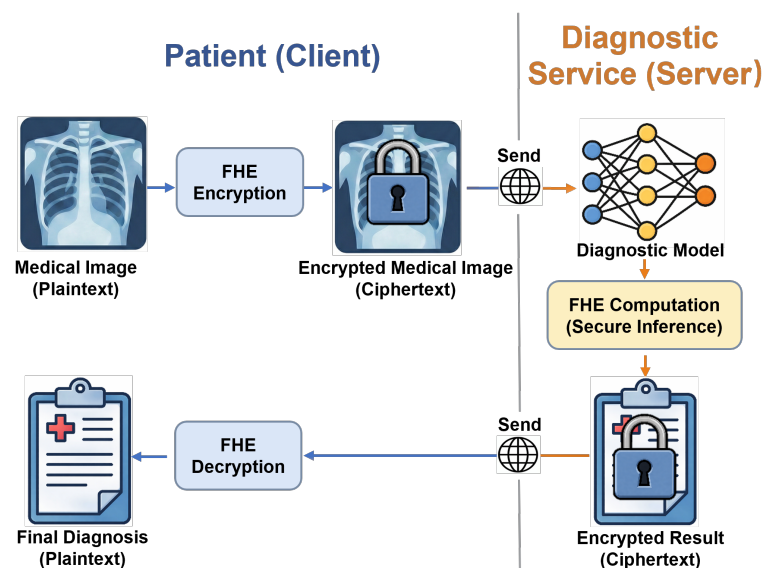


Figure 1. FHE application scenario in medical diagnostic service.

Among the various FHE schemes, RNS-CKKS [15], which supports approximate floating-point arithmetic, has been widely used in secure inference tasks [16,17]. In practice, RNS-CKKS natively supports only arithmetic operations such as addition and multiplication and therefore cannot directly evaluate the nonlinear activation function commonly used in CNNs, namely, the Rectified Linear Unit (ReLU). As a result, the ReLUs in networks such as ResNet [18] and VGG [19] must be replaced with polynomial approximations [16,20,21]. However, evaluating a polynomial under RNS-CKKS requires ciphertext multiplications, and the number of such multiplications that a ciphertext can support is limited before bootstrapping is needed to refresh it. This creates a fundamental trade-off: low-degree polynomials reduce multiplicative cost but often degrade inference accuracy, whereas high-degree polynomials approximate ReLU more precisely but consume more

levels, trigger bootstrapping more frequently, and thus substantially increase end-to-end inference latency.

Prior Work. The existing work can be viewed from the stage at which optimization is performed. Mixed-degree design methods [22,23] determine the polynomial network itself by assigning different polynomial degrees across activation layers, thereby changing the multiplicative-depth profile and the number of required bootstrapping operations. In contrast, FHE compiler methods such as DaCapo, Orion, and ReSBM [24–26] optimize the execution of an already specified polynomial computation graph, for example, through scale management, bootstrapping placement, or restored-level selection. These studies show that encrypted inference latency is affected not only by the bootstrapping count but also by how ciphertext levels are consumed between bootstrapping operations. However, existing mixed-degree design methods still largely use bootstrapping count as the main execution-cost proxy and therefore may overlook degree configurations with better level-use behavior.

Table 1 summarizes the positioning of CipherAPR relative to related design- and compiler-side methods.

Table 1. Positioning of CipherAPR relative to mixed-degree design and compiler-side bootstrapping optimization methods.

Method	Stage	Poly. Degree	Level-Aware	Relation
MPCNN [16]	Design	High	No	Baseline
AutoFHE [22]	Design	Mixed	No	Baseline
DaCapo [24]	Compiler	High	Yes	Complementary
Orion [25]	Compiler	High	Yes	Complementary
ReSBM [26]	Compiler	High	Yes	Complementary
CipherAPR	Design	Mixed	Yes	Proposed

Problem Statement and Challenge. Motivated by the above gap, our goal is to find a mixed-degree polynomial network that achieves a better latency–accuracy trade-off under RNS-CKKS. A layer-wise degree assignment should not only reduce bootstrapping count but also make more effective use of the ciphertext levels restored by bootstrapping. Directly coupling degree search with full execution scheduling would require re-evaluating placement and level decisions for many candidate networks, making the offline design loop prohibitively expensive. Therefore, an effective design method should bring execution-side level-use information into the degree-search stage through a lightweight proxy. It should favor degree configurations whose level consumption better fits bootstrapping segments while keeping candidate evaluation inexpensive enough for offline search.

This Work. This paper introduces CipherAPR, an importance-guided framework for level-aware mixed-degree polynomial design that reduces bootstrapping count while improving the utilization of the levels restored by bootstrapping. The name CipherAPR reflects our focus on ciphertext-level utilization and mixed-degree polynomial approximation in encrypted inference.

First, CipherAPR introduces the Low-Magnitude Activation Ratio (LMAR), an importance metric tailored to polynomial neural networks, to prioritize degree updates on accuracy-sensitive activation layers. Second, it combines Domain-Adaptive Approximation (DAAP) with Multi-Objective Coefficient Tuning (MOCT) to generate reusable, numerically stable, and less computationally intensive polynomial approximations, reducing candidate construction cost during offline design. Third, it performs fast candidate screening with lightweight latency and accuracy estimators, and it uses ciphertext-level utilization (CLU) as a structural filtering signal to deprioritize configurations with imbalanced restored-level utilization.

Contributions. In summary, this paper makes the following contributions:

1. We present CipherAPR, an importance-guided framework for level-aware mixed-degree polynomial design in RNS-CKKS inference. Unlike existing mixed-degree methods that mainly optimize degree assignments for bootstrapping count, CipherAPR incorporates restored-level utilization to improve the latency–accuracy trade-off under bootstrapping-sensitive execution.
2. We develop an efficient offline design pipeline that addresses the enlarged candidate space through three complementary mechanisms. The LMAR concentrates degree adjustments on accuracy-sensitive activation layers. DAAP and MOCT produce reusable, numerically stable polynomial approximations under each candidate degree setting, reducing the per-candidate construction cost. Lightweight screening via latency and accuracy estimators, combined with CLU-based structural filtering, improves the early search trajectory toward configurations with more balanced restored-level utilization.
3. We demonstrate that CipherAPR delivers consistent encrypted inference speedups at comparable accuracy. On the evaluated ResNet and VGG models under different bootstrapping settings, CipherAPR achieves a $1.09\times\text{--}1.39\times$ speedup over AutoFHE.

The rest of this paper is organized as follows. Section 2 introduces the RNS-CKKS scheme and composite polynomial approximation. Section 3 formalizes the motivation and problem definition. Section 4 presents the CipherAPR framework, covering degree allocation, configuration instantiation, evaluation and filtering, and a security and efficiency discussion. Section 5 describes the cryptographic and experimental setup, and Section 6 reports the evaluation results. Section 7 concludes the paper.

2. Preliminaries

2.1. RNS-CKKS Fully Homomorphic Encryption

We adopt the RNS variant of the CKKS scheme [15] (RNS-CKKS), which supports efficient approximate arithmetic over real or complex numbers. In this scheme, a message vector is encoded into a plaintext polynomial $m(x)$ and encrypted into a ciphertext $ct \in \mathcal{R}_{Q_\ell}^2$. The computation is performed over the cyclotomic ring $\mathcal{R}_{Q_\ell} = \mathbb{Z}_{Q_\ell}[X]/(X^N + 1)$, where the modulus $Q_\ell = \prod_{i=0}^{\ell} q_i$ is the product of distinct primes. The parameter ℓ denotes the ciphertext level, representing the remaining multiplicative capacity (often referred to as the remaining level budget). The number of levels consumed by an operation or sequence of operations is termed its multiplicative depth. Homomorphic operations in RNS-CKKS rely on the modulus chain. Specifically, a homomorphic multiplication increases the scale factor of the message, necessitating a rescaling operation to maintain precision. Rescaling divides the ciphertext by the last prime q_ℓ and switches the modulus from Q_ℓ to $Q_{\ell-1}$, thereby consuming one level. Once the level drops to zero ($\ell = 0$), the ciphertext must undergo bootstrapping [27], a computationally intensive procedure that refreshes the ciphertext to a higher level. Since bootstrapping dominates the inference latency, minimizing its frequency is critical for performance. Furthermore, RNS-CKKS utilizes SIMD packing to encode $N/2$ values into a single ciphertext slot. Supported operations include ciphertext addition (Add), ciphertext–ciphertext multiplication (Cmult), and plaintext–ciphertext multiplication (Pmult), which form the building blocks of encrypted neural networks.

2.2. Minimax Composite Polynomial Approximation

Approximating the nonlinear ReLU function in FHE typically involves approximating the sign function, $\text{sgn}(x)$, since $\text{ReLU}(x) = (x + x \cdot \text{sgn}(x))/2$. Lee et al. [28] proposed an efficient method using composite polynomials to approximate $\text{sgn}(x)$ over the union of intervals $[-1, -\epsilon] \cup [\epsilon, 1]$, excluding the singularity at zero. The core idea is to iteratively compose low-degree polynomials, $p(x) = p_k \circ p_{k-1} \circ \dots \circ p_1(x)$ to achieve a high-precision approximation. Specifically, the first polynomial p_1 maps the initial input domain $[\epsilon, 1]$ to

an intermediate range $[\epsilon_1, v_1]$. Subsequent polynomials p_i (for $i = 2, \dots, k$) are designed to map the previous output range $[\epsilon_{i-1}, v_{i-1}]$ to a progressively narrower interval closer to 1, i.e., $[\epsilon_i, v_i] \subseteq [1 - 2^{-\alpha_i}, 1 + 2^{-\alpha_i}]$. This composition effectively amplifies the “steepness” near zero, resulting in a step-like function that closely resembles $\text{sgn}(x)$.

The approximation error is highly sensitive to the width of the discontinuity gap $(-\epsilon, \epsilon)$. The first polynomial p_1 must transition sharply from -1 to 1 across this gap. As illustrated in Figure 2, narrowing the input domain significantly eases this requirement. For instance, comparing the polynomials derived on a standard domain (panels a–c) versus a narrowed domain (panels d–f), it is evident that a narrower domain allows for the polynomial to converge to the target value more swiftly, thereby reducing the approximation error for a fixed polynomial degree. This observation motivates the need for domain-adaptive strategies.

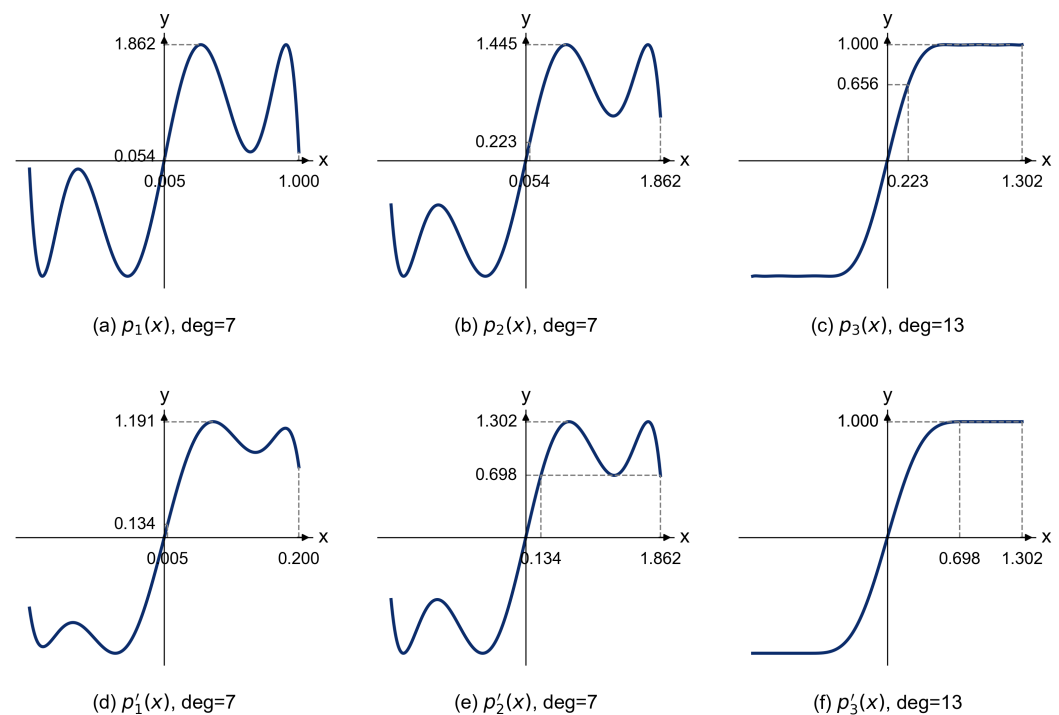


Figure 2. Composite polynomials $p = p_3 \circ p_2 \circ p_1$ with degree set $\{7, 7, 13\}$. Panels (a–c) show p_1 , p_2 , and p_3 on the wide domain $D_1 = [-1, -0.005] \cup [0.005, 1]$; panels (d–f) show the corresponding $p'_1(x)$, $p'_2(x)$, and $p'_3(x)$ on the narrowed domain $D_2 = [-0.2, -0.005] \cup [0.005, 0.2]$.

2.3. RNS-CKKS Inference

Under RNS-CKKS encrypted inference [16], multiplicative-depth consumption depends strongly on the layer type, because each homomorphic multiplication typically requires a rescaling step and thus consumes one level. In convolution and batch normalization layers, the dominant operations are plaintext–ciphertext multiplications (PMults), where encrypted activations are multiplied by plaintext model parameters. In practice, these linear operations are commonly fused and scheduled as a ConvBN operator; in our cost model, we treat a ConvBN as having an effective depth consumption of two. In contrast, ciphertext–ciphertext multiplications (CMults) mainly arise in nonlinear layers, most notably in polynomial approximations of ReLU.

As a concrete example, consider a single polynomial stage $p(x) = wx^2$. Its evaluation first computes x^2 via one CMult, and then applies the coefficient via one PMult, leading to a total depth consumption of two. For higher-degree ReLU approximations constructed through sign-function approximation, the nonlinear cost is largely determined by the com-

posite sign polynomial itself. This is particularly pronounced for composite constructions of the form

$$p(x) = (p_k \circ p_{k-1} \circ \cdots \circ p_1)(x), \quad \deg(p_i) = d_i,$$

where each stage approximates $\text{sgn}(x)$ under the ciphertext-level budget. To reduce the cost of evaluating this composite sign polynomial, we adopt the Baby Step Giant Step (BSGS) strategy [29]. With BSGS, the depth consumption of evaluating the i -th stage can be upper bounded by $\lceil \log_2(d_i + 1) \rceil$, yielding the overall bound

$$\sum_{i=1}^k \lceil \log_2(d_i + 1) \rceil.$$

Table 2 summarizes representative composite degree sets for sign-function approximation and their associated depth consumption and multiplication counts.

Table 2. Composite polynomial degree sets for sign-function approximation and the corresponding multiplicative depths and multiplication counts [28].

Degree Set	Depth Consumption	Multiplications
{3, 7}	5	7
{7, 7}	6	10
{7, 15}	7	13
{15, 15}	8	16
{7, 7, 13}	10	17
{7, 7, 27}	11	20
{7, 15, 27}	12	23
{15, 15, 27}	13	26

2.4. Cryptographic Principle

A key cryptographic principle behind CipherAPR is the level dependence of RNS-CKKS execution cost. Let $\mathcal{B}_\ell = \{q_0, q_1, \dots, q_\ell\}$ denote the active RNS basis at level ℓ , with $Q_\ell = \prod_{q_i \in \mathcal{B}_\ell} q_i$ and $|\mathcal{B}_\ell| = \ell + 1$. We write the aggregate cost of primitive homomorphic operations executed at level ℓ as

$$T_{\text{op}}(\ell) = \sum_{k \in \mathcal{P}} n_k t_k(\ell),$$

where $\mathcal{P} = \{\text{AddCP}, \text{AddCC}, \text{MulCP}, \text{MulCC}, \text{Rot}, \text{Relin}, \text{RS}\}$ is the set of primitive operation types, n_k is the number of operations of type k , and $t_k(\ell)$ is the corresponding single-operation cost. Here, MulCC refers to the dyadic product of two ciphertexts prior to relinearization, and RS denotes rescaling (dividing by one RNS prime). Under a fixed parameterization, we use the following level-dependent approximation:

$$t_k(\ell) \approx \begin{cases} \alpha_k N (\ell + 1), & k \in \{\text{AddCP}, \text{AddCC}, \text{MulCP}, \text{MulCC}, \text{RS}\}, \\ \beta_k N \log N (\ell + 1)^2, & k \in \{\text{Rot}, \text{Relin}\}, \end{cases}$$

where N is the ring degree and α_k, β_k are operation-dependent constants. The first group consists of element-wise operations over each RNS limb, whose cost scales linearly with the number of active primes $\ell + 1$. The second group involves key switching, which requires NTT-based polynomial multiplications across all limbs and thus scales quadratically. This formulation reflects that lower levels correspond to shorter active modulus chains and

fewer RNS primes participating in ring arithmetic. Consequently, all primitive operations become cheaper as the level decreases, although their exact scaling is operation-dependent. Therefore, a smaller post-bootstrapping effective level L_{eff} can reduce the cost of subsequent encrypted computation, but it also leaves a tighter multiplicative-depth budget and may increase the bootstrapping count. Under a fixed bootstrapping implementation, the end-to-end latency can be abstracted as

$$T_{\text{inf}} \approx \sum_{j=1}^{N_{\text{boot}}+1} T_{\text{seg}}^{(j)}(L_{\text{eff}}) + N_{\text{boot}} T_{\text{boot}},$$

where N_{boot} denotes the number of bootstrapping operations. CipherAPR acts on this trade-off by reshaping ciphertext-level consumption through mixed-degree polynomial approximation under a fixed bootstrapping primitive.

3. Motivation and Problem Definition

Mixed-degree polynomial design in RNS-CKKS affects ciphertext-level consumption and thus the feasible bootstrapping schedule and encrypted inference latency. This section formalizes the interaction between degree design and bootstrapping placement as a constrained optimization problem and identifies three key challenges that CipherAPR addresses.

3.1. Motivation

The three challenges correspond to three stages of the design problem: what degrees to assign (Challenge 1), how to instantiate the assignment (Challenge 2), and how to evaluate it efficiently (Challenge 3).

3.1.1. Challenge 1: Degree Allocation Under Ciphertext-Level Effects

In RNS-CKKS, layer-wise polynomial degrees determine ciphertext-level consumption, while the post-bootstrapping effective level L_{eff} determines the remaining multiplicative-depth budget. As a result, the latency implication of a mixed-degree configuration depends on both approximation quality and its interaction with bootstrapping placement. Existing mixed-degree design methods usually fix L_{eff} and leave this coupling unmodeled. Challenge 1 is thus to allocate degrees in a way that remains sensitive to ciphertext-level effects without resorting to exhaustive refinement.

3.1.2. Challenge 2: Efficient and Stable Polynomial Instantiation

Once a degree configuration is fixed, it must be instantiated as layer-wise polynomial approximations of ReLU that are accurate on the empirical activation domain and stable under ciphertext arithmetic. In RNS-CKKS, large coefficients and poorly controlled output ranges can amplify noise growth and degrade numerical robustness, whereas refitting coefficients from scratch for every configuration would incur substantial offline overhead. Challenge 2 is therefore to build a reusable instantiation mechanism that produces high-quality polynomial approximations with stable numerical behavior under ciphertext computation across multiple degree settings.

3.1.3. Challenge 3: Efficient Candidate Evaluation and Filtering

During offline search, each candidate mixed-degree configuration must be assessed for both encrypted inference latency and accuracy. However, exact latency measurement requires running full encrypted inference, and obtaining a reliable accuracy estimate requires training the polynomial network to convergence. Both are prohibitively expensive when many candidates must be compared. Beyond raw cost, a further concern is structural

quality. Even among candidates with competitive latency and accuracy estimates, some may distribute ciphertext-level consumption unevenly across bootstrapping segments, making them less favorable starting points for subsequent search iterations. Challenge 3 is therefore to build an efficient evaluation and filtering mechanism that can rank candidates by latency and accuracy under a limited offline budget while favoring configurations with smoother segment-level utilization.

3.2. Problem Definition

We formulate CipherAPR under a fixed cryptographic setting Λ , including the RNS-CKKS ring degree, modulus chain, and target security level. Under this fixed setting, the design variable is the layer-wise polynomial degree assignment of the encrypted network.

A mixed-degree configuration is represented by a degree vector

$$\mathbf{d} = (d_1, d_2, \dots, d_L), \quad \text{where } d_i \in \mathbb{D}$$

where d_i denotes the polynomial degree assigned to the i -th activation layer and $\mathbb{D}$ is the set of available polynomial degrees. For a given configuration $\mathbf{d}$, let $\Pi(\mathbf{d}; \Lambda)$ denote the set of feasible bootstrapping placements under the fixed cryptographic setting Λ . We define the encrypted inference latency of $\mathbf{d}$ as

$$t(\mathbf{d}) = \min_{\pi \in \Pi(\mathbf{d}; \Lambda)} T(\mathbf{d}, \pi; \Lambda),$$

where $T(\mathbf{d}, \pi; \Lambda)$ is the end-to-end latency under degree configuration $\mathbf{d}$, bootstrapping placement π , and cryptographic setting Λ . The optimization problem is then

$$\mathbf{d}^* = \arg \min_{\mathbf{d} \in \mathcal{D}} t(\mathbf{d}) \quad \text{s.t.} \quad \text{Acc}(\mathbf{d}) \geq \text{Acc}_{\text{target}}$$

where $\mathcal{D} \subseteq \mathbb{D}^L$ denotes the set of feasible layer-wise degree configurations.

4. CipherAPR Design

4.1. Overview

CipherAPR addresses this problem through a three-stage offline pipeline under fixed cryptographic parameters. The three stages are degree allocation, configuration instantiation, and configuration evaluation and filtering. Before detailing these stages, we clarify three related terms used throughout this section. Polynomial approximation refers to the mathematical function approximating ReLU. Configuration instantiation is the overall process of generating deployable coefficients for a given degree assignment. Activation replacement denotes substituting the original ReLUs with these polynomials in the network. Figure 3 summarizes the full pipeline.

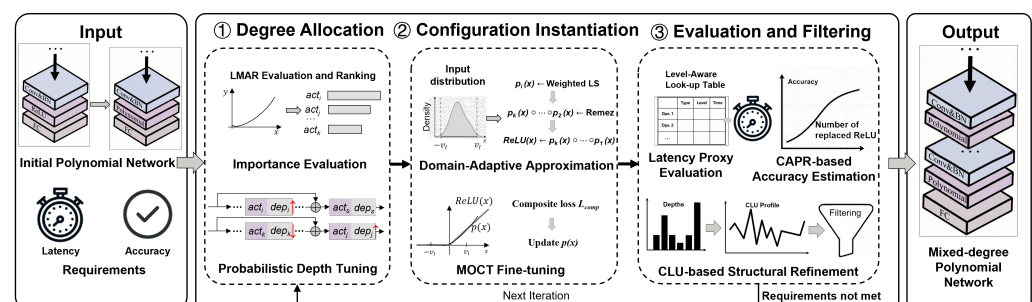


Figure 3. Overview of CipherAPR framework for RNS-CKKS inference. Arrows indicate the offline design flow and the feedback loop when constraints are unmet.

In the degree allocation stage, the LMAR identifies activation layers whose near-zero input concentration makes them most sensitive to approximation quality. A probabilistic degree-update rule then concentrates adjustments on these layers under the available level budget. In the configuration instantiation stage, each degree configuration is realized as a deployable set of layer-wise polynomial approximations through Domain-Adaptive Approximation (DAAP) and MOCT-based coefficient tuning, ensuring ciphertext-safe numerical behavior. Because the resulting polynomials achieve higher fidelity than classical constructions at each degree level, the instantiation stage effectively allows for lower-degree polynomials to be selected under the same error tolerance, reducing per-layer multiplicative depth. In the configuration evaluation and filtering stage, CipherAPR screens candidates using a relative latency proxy and a low-budget staged accuracy estimator, Cosine-scheduled Activation Progressive Replacement (CAPR). It then forms a provisional shortlist and applies CLU-based structural filtering to favor candidates with smoother level consumption before the next search iteration.

Table 3 summarizes the main symbols used in the subsequent design details.

Table 3. Summary of key notation used in CipherAPR.

Symbol	Description
$\mathbf{d} = (d_1, \dots, d_L)$	Layer-wise polynomial degree assignment.
$\mathbb{D}$	Candidate set of polynomial degrees.
L_{eff}	Effective ciphertext level restored after bootstrapping.
$\text{LMAR}_{\ell}, \epsilon$	Low-Magnitude Activation Ratio of layer ℓ and its threshold.
$v_{\ell}, B, \rho_{\text{ext}}$	DAAP core-domain radius, domain scaling factor, and extension factor.
$\hat{t}(\mathbf{d}), \widehat{\text{Acc}}(\mathbf{d})$	Search-time latency proxy and CAPR-based accuracy estimate.
$\text{CLU}_s, \text{RMS}_{\text{adj}}$	Segment-level ciphertext-level utilization and adjacent-segment variation.

4.2. Degree Allocation

4.2.1. Dynamic Importance Evaluation of Activation Layers

Existing ReLU-importance criteria typically measure the accuracy drop caused by removing individual activations or the sensitivity reflected by changes in gradient magnitudes under a fixed network structure [30,31]. In our setting, however, multiple activations are replaced by polynomials simultaneously, and their relative importance changes as degree assignments evolve during search. Gradient-based saliency [32] is usually evaluated on a fixed network and would require backward passes for many candidate polynomial configurations; for high-degree polynomial activations, such propagation can also be costly or unstable, as noted in AutoFHE [22]. We therefore need a dynamic, forward-only metric that reflects approximation sensitivity without repeated backward computation.

A layer's sensitivity to polynomial degree is highest when its activations concentrate near the ReLU switching point at zero. In this region, a low-degree approximation is more likely to distort whether small activations are suppressed or passed forward, and the resulting error can propagate through subsequent layers. To capture this effect, we define the Low-Magnitude Activation Ratio (LMAR).

$$\text{LMAR}_{\ell} = \frac{1}{k_{\ell}} \sum_{i=1}^{k_{\ell}} 1(|x_{\ell,i}| < \epsilon),$$

where $x_{\ell,i}$ denotes the input of the i -th neuron in activation layer ℓ and k_{ℓ} is the total neuron count. A higher LMAR_{ℓ} indicates stronger near-zero concentration, suggesting that the layer would benefit more from a higher-degree polynomial under the same level budget.

We determine the threshold ϵ from the empirical activation distribution using Otsu's method [33]. Activation magnitudes across all layers are partitioned into two classes at threshold ϵ . The within-class variance is defined as

$$\sigma_w^2(\epsilon) = w_1(\epsilon) \sigma_1^2(\epsilon) + w_2(\epsilon) \sigma_2^2(\epsilon),$$

and the between-class variance is

$$\sigma_b^2(\epsilon) = w_1(\epsilon) w_2(\epsilon) (\mu_1(\epsilon) - \mu_2(\epsilon))^2,$$

where $w_1(\epsilon)$ and $w_2(\epsilon)$ denote the class probabilities and $\mu_1(\epsilon)$ and $\mu_2(\epsilon)$ the class means. The optimal ϵ maximizes $\sigma_b^2(\epsilon) / \sigma_w^2(\epsilon)$, yielding a single global threshold that best separates near-zero from non-near-zero activations. In practice, the LMAR primarily uses the rank ordering of activation layers; the absolute threshold value has limited influence on the probabilistic update rule. A theoretical analysis of the LMAR is provided in Appendix A.

Because the LMAR is forward-only, inexpensive, and stable as a rank-ordering signal, it is well suited for the dynamic mixed-degree search where candidate configurations change iteratively. It directly targets the near-zero ReLU region where polynomial approximation is most likely to cause gating inconsistency, without incurring the cost of backward propagation across many candidate networks.

4.2.2. Probabilistic Activation-Layer Depth Tuning

The goal of this stage is to reshape the per-layer multiplicative depth so that level consumption is distributed efficiently across bootstrapping segments. Since the multiplicative depth of each activation layer is determined by its polynomial degree, CipherAPR achieves this by adjusting the degree assigned to each layer under the guidance of the LMAR, concentrating degree updates on layers where the resulting depth change most benefits the encrypted latency–accuracy trade-off.

Starting from an initial low-degree configuration in which all ReLU activations are uniformly replaced with preset low-degree polynomials, CipherAPR iteratively refines the layer-wise degree assignment. For each activation layer ℓ , three actions are considered: IncreaseDegree (ID), DecreaseDegree (DD), and KeepDegree (KD), which respectively increase, decrease, or preserve the current polynomial degree.

Let the set of activation layers be $\mathcal{A}$ and the number of layers be $M = |\mathcal{A}| \geq 2$. We calculate $L_\ell = \text{LMAR}_\ell$ for each layer and perform a descending sort to obtain the rank $r_\ell \in \{1, \dots, M\}$ ($r_\ell = 1$ indicates the highest LMAR). The normalized rank score is $L_\ell^{\text{norm}} = \frac{M-r_\ell}{M-1} \in [0, 1]$. We compute the base probability of degree adjustment actions using a centered hinge function with center $\tau_c = 0.5$ and a gradually shrinking margin $m \in [0, 0.5)$

$$\begin{aligned} h_\ell^{\text{ID}} &= \max\{0, 2(L_\ell^{\text{norm}} - \tau_c) - 2m\}, \\ h_\ell^{\text{DD}} &= \max\{0, 2(\tau_c - L_\ell^{\text{norm}}) - 2m\}. \end{aligned}$$

The action probabilities are then defined as

$$\begin{aligned} P_{\text{ID}}(\ell) &= \beta_\ell^{\text{ID}} h_\ell^{\text{ID}}, \\ P_{\text{DD}}(\ell) &= \beta_\ell^{\text{DD}} h_\ell^{\text{DD}}, \\ P_{\text{KD}}(\ell) &= \max\{0, 1 - P_{\text{ID}}(\ell) - P_{\text{DD}}(\ell)\}, \end{aligned}$$

where β_ℓ^{ID} and β_ℓ^{DD} are dynamic coefficients. Initially, we set $\beta_\ell^{\text{ID}} = \beta_0^{\text{ID}}$ and $\beta_\ell^{\text{DD}} = \beta_0^{\text{DD}}$, where $\beta_0^{\text{ID}}, \beta_0^{\text{DD}} \in (0, 1]$ are global upper bounds. If layer ℓ was modified in the previ-

ous round, we apply cooling: $\beta_\ell^{ID} \leftarrow \rho \beta_\ell^{ID}$ and $\beta_\ell^{DD} \leftarrow \rho \beta_\ell^{DD}$, where $\rho \in (0, 1)$. Let $d_{\max} = \max \mathbb{D}$ be the global maximum available degree. For layer ℓ , ID draws uniformly from the feasible set $\{d \in \mathbb{D} \mid d_\ell < d \leq d_{\max}\}$, and DD draws from $\{d \in \mathbb{D} \mid 0 \leq d < d_\ell\}$.

The margin parameter m controls the selectivity of the update rule. A larger margin keeps more layers in the *KeepDegree* region and leads to a more conservative search, whereas a smaller margin allows for more layers to be updated and promotes exploration. The cooling factor ρ reduces the probability of modifying a layer that was changed in the previous round, thereby preventing oscillatory updates on the same layer and encouraging exploration of other promising activation sites.

4.3. Configuration Instantiation

After fixing the degree of each activation layer, the resulting configuration must be instantiated as a deployable encrypted-inference network. The layer-wise polynomials produced in this stage are designed to achieve high approximation fidelity and can therefore be inserted directly into the polynomial network for inference. At the same time, their coefficient magnitudes and output ranges must remain stable under ciphertext arithmetic. To this end, our approach integrates Domain-Adaptive Approximation (DAAP) with a subsequent Multi-Objective Coefficient Tuning (MOCT) stage to produce polynomial approximations that remain numerically robust under encrypted arithmetic. Because the resulting polynomials achieve higher approximation fidelity at each degree level than classical constructions, a DAAP + MOCT polynomial of degree d can attain approximation error comparable to that of a classical polynomial of higher degree. This effectively allows for the degree-allocation stage to select lower-degree polynomials under the same error tolerance, reducing the per-layer multiplicative depth and freeing ciphertext levels for subsequent computation.

4.3.1. Domain-Adaptive Approximation

For each activation layer to be replaced, we first estimate its input bounds $[l_\ell, u_\ell]$ from the empirical input distribution obtained during importance evaluation and define a symmetric approximation domain $[-v_\ell, v_\ell]$ for the ReLU as

$$v_\ell = \frac{\max\{|l_\ell|, |u_\ell|\}}{B}, \quad v_\ell \in [a, 1],$$

where $B > 1$ is a global numerical scaling factor used for activation-domain normalization, and $a > 0$ is a small lower bound. The factor B is used solely for activation-domain normalization and is unrelated to the CKKS encryption scale. For notational simplicity, we write $v = v_\ell$ in the following.

We express the ReLU function as $x \cdot (0.5 + 0.5 \cdot \text{sgn}(x))$ and approximate the sign function using a composition of polynomials

$$p(x) = (p_k^{d_k} \circ \dots \circ p_1^{d_1})(x),$$

where $p_j^{d_j}(x)$ denotes the j -th sub-polynomial of degree d_j .

We consider a core approximation domain $I_{\text{core}} = [-v, v]$ for the ReLU polynomial, and an extended domain $I_{\text{ext}} = [-\rho_{\text{ext}}v, \rho_{\text{ext}}v]$ with $\rho_{\text{ext}} > 1$, which captures rare but potentially harmful out-of-core inputs. Since $\text{sgn}(x)$ is odd, we only describe the approximation on the positive domain $[\delta, v]$, where $\delta > 0$ is a small numerical cutoff. First, we apply a segmented weighted least-squares approximation on the extended domain $[\delta, \rho_{\text{ext}}v]$ to obtain a polynomial $p_1^{d_1}$. This polynomial is constructed so that its output falls into a narrow band $[\delta_1, v_1] \subseteq [1 - \tau_1, 1 + \tau_1]$ away from zero, while boundary fluctuations on the input domain $[\delta, v]$ are suppressed. Then, the remaining sub-polynomials $p_2^{d_2}, \dots, p_k^{d_k}$ are

computed using the Remez algorithm on a sequence of domains $[\delta_j, v_j]$, $j = 2, \dots, k$, each minimizing the maximum approximation error on its corresponding domain. Composing these sub-polynomials yields an approximation of $\text{sgn}(x)$ on $[\delta, v]$, which we then plug into $x \cdot (0.5 + 0.5 p(x))$ to obtain the final polynomial approximation of ReLU.

The parameters B and ρ_{ext} jointly balance approximation fidelity and numerical stability on the ciphertext side. Since $v_\ell = \max |l_\ell|, |u_\ell|/B$, a small B results in a wider fitting domain, increasing the approximation error for a fixed polynomial degree, whereas an excessively large B overly compresses the core domain, potentially producing large coefficients that can cause errors in ciphertext computation [21]. The extension factor ρ_{ext} provides a safety margin around the core domain, thereby covering moderate activation tails and reducing approximation errors near the core-domain boundary.

4.3.2. Multi-Objective Polynomial Coefficient Fine-Tuning

The approximation precision and numerical stability of the polynomial are further improved by the MOCT procedure, which fine-tunes its coefficients. Let

$$g(x) = x \cdot (0.5 + 0.5 p(x))$$

denote the resulting ReLU approximation. Based on the domain-adaptive construction in the previous subsection, MOCT improves approximation precision on I_{core} , constrains the behavior of $g(x)$ on I_{ext} to avoid abnormal fluctuations outside the core domain, and regularizes both curvature and coefficient norms. To this end, we design a custom loss function, *CompositeLoss*, consisting of four components.

- Core Error (CE). Quantifies the approximation error between $g(x)$ and the target $\text{ReLU}(x)$ within the core domain I_{core} , defined as $\text{CE} = \max_x |g(x) - \text{ReLU}(x)|$.
- Extended Domain Constraint (EDC). Imposes penalties when $g(x)$ exceeds the output range implied by the extended domain I_{ext} , defined as $\text{EDC} = \mathbb{E}_x [\text{ReLU}(|g(x)| - \rho_{\text{ext}} v)]$.
- Curvature Penalty (CP). Suppresses numerical oscillations by regularizing the local curvature of $g(x)$ (e.g., Runge's phenomenon), defined as $\text{CP} = \mathbb{E}_x \left[\left| \frac{d^2 g(x)}{dx^2} \right| \right]$.
- Coefficient Regularization (CR). Applies norm-based regularization to the coefficients of $p(x)$ to constrain their magnitude, thereby improving numerical stability for ciphertext computation, defined as $\text{CR} = \sum_i a_i^2$.

These terms are aggregated using a scalar weight vector $\mathbf{w} = [w_1, w_2, w_3, w_4]$ to form the composite loss function $\mathcal{L}_{\text{comp}} = w_1 \cdot \text{CE} + w_2 \cdot \text{EDC} + w_3 \cdot \text{CP} + w_4 \cdot \text{CR}$.

The MOCT weights are chosen to balance the numerical scales of the four loss components while reflecting their optimization priorities. The CE term emphasizes core-domain approximation fidelity, while the EDC term penalizes extended-domain violations when they occur. CP and CR serve as auxiliary regularizers that discourage oscillatory behavior and overly large coefficients. The same weights are fixed across all models and datasets.

We minimize $\mathcal{L}_{\text{comp}}$ by adjusting the odd-degree coefficients in $p_1(x)$ for a fixed number of epochs (e_0) while keeping the remaining coefficients fixed. The output is a set of optimized coefficients that define the final polynomial approximation of ReLU. Algorithm 1 summarizes the procedure.

Algorithm 1 Multi-Objective Coefficient Tuning (MOCT)

Require: Core radius v , extension factor ρ_{ext} , coefficient vectors $\{\vec{a}_i\}_{i=1}^k$ of sub-polynomials $p_i(x)$, training epochs e_0 , loss weights $[w_1, w_2, w_3, w_4]$

Ensure: Optimized coefficients $\vec{a}_1^*$ of polynomial $p_1(x)$

- 1: Define ReLU approximation $g(x) = x \cdot (0.5 + 0.5 p(x))$
- 2: Set odd-degree coefficients in $p_1(x)$ as trainable; fix all others
- 3: **for** $epoch = 1$ to e_0 **do**
- 4: Compute $\mathcal{L} \leftarrow \text{CompLoss}(g(x))$
- 5: Backpropagate and update the trainable odd-degree coefficients of $p_1(x)$
- 6: **end for**
- 7: **return** Optimized coefficients $\vec{a}_1^*$

4.4. Configuration Evaluation and Filtering

4.4.1. Latency Proxy and Staged Accuracy Estimation

We use a latency proxy and a staged accuracy estimate to screen candidates before structural filtering and pool update.

Bootstrapping Placement and Latency Proxy. For a mixed-degree configuration, the latency is defined as $t(\mathbf{d}) = \min_{\pi \in \Pi(\mathbf{d}; \Lambda)} T(\mathbf{d}, \pi; \Lambda)$. CipherAPR approximates this value under the fixed cryptographic setting by testing a discrete set of feasible bootstrapping positions around each activation. At location ℓ , bootstrapping may be inserted at the residual block boundary before the layer, immediately before the layer, or immediately after the layer. We use dynamic programming to find the minimum proxy latency over these choices. The DP state tracks the current layer index and remaining ciphertext level. Transitions correspond to inserting or not inserting bootstrapping at each candidate location. Each inserted bootstrapping resets the level to L_{eff} and adds one bootstrapping cost.

We evaluate latency with a unified proxy. For linear operators, the cost comes from a lookup table indexed by the input ciphertext level. The table is built from primitive operation counts such as ciphertext–plaintext multiplications. For polynomial layers, the cost is modeled as a monotone function of degree d_ℓ times the baseline proxy time at the same input level, while pruned activations with $d_\ell = 0$ add no activation cost. After the bootstrapping positions are fixed, each segment has starting level L_s and consumed depth D_s . We discard the unused levels $u_s = L_s - D_s$ and index the lookup table by the consumed depth $\tilde{L}_s = D_s$. This makes the proxy more sensitive to differences among bootstrapping partitions while preserving a useful ordering of execution cost.

CAPR Accuracy Estimation. To estimate $\text{Acc}(\mathbf{d}')$, we use CAPR, which performs progressive replacement with a small training budget. In CipherAPR, CAPR serves only as a screening estimate and does not replace the final accuracy obtained after full training.

Let the network contain k ReLUs, the total training epochs be T , the minimum epochs per stage be $M_{\min}$, and the number of stages be Q . CAPR uses a cosine schedule to allocate the training epochs m_q and replacement count s_q for each stage

$$m_q = \left\lfloor \left(\frac{2T}{Q} - \Delta \right) \cdot \frac{1 + \cos\left(\pi \cdot \frac{q-1}{Q-1}\right)}{2} \right\rfloor + M_{\min}, \quad q = 1, \dots, Q,$$

where Δ controls the decay and we use $\Delta = 2M_{\min}$ by default. We then set $s_q = \lceil k \cdot m_q / T \rceil$ and $n_q = \sum_{j=1}^q s_j$. At stage q , the first n_q activations in the LMAR ranking are replaced with polynomials, and the network is trained for m_q epochs. For the last stage, we set $m_Q \leftarrow \max(M_{\min}, T - \sum_{q=1}^{Q-1} m_q)$.

The stage number Q is a search-time training-schedule parameter. A very small Q makes activation replacement abrupt and can lead to unstable accuracy estimates because many ReLUs are replaced before the network adapts. A very large Q provides a smoother

replacement schedule but increases search-time overhead and may split the fixed training budget into too many small stages. We therefore use a moderate Q to balance estimation fidelity and search efficiency. This parameter is used only for offline candidate screening and does not affect the final encrypted inference procedure.

4.4.2. Score Preselection and CLU Structural Filtering

Score Preselection. CipherAPR maintains two pools of width W , namely, an accuracy pool R_{acc} and a latency pool R_{lat} . For the candidate set $\mathcal{C}$, let $\widehat{\text{Acc}}_{\max} = \max_{\mathbf{d}'' \in \mathcal{C}} \widehat{\text{Acc}}(\mathbf{d}'')$ and $\hat{t}_{\max} = \max_{\mathbf{d}'' \in \mathcal{C}} \hat{t}(\mathbf{d}'')$. Each candidate $\mathbf{d}' \in \mathcal{C}$ receives the score

$$\text{Score}(\mathbf{d}') = \lambda \frac{\widehat{\text{Acc}}(\mathbf{d}')}{\widehat{\text{Acc}}_{\max}} - (1 - \lambda) \frac{\hat{t}(\mathbf{d}')}{\hat{t}_{\max}}$$

where a larger λ favors accuracy and a smaller λ favors latency. To preserve room for CLU filtering, we first keep the best W_{pre} candidates under each preference, where $W_{\text{pre}} = \lceil W/\eta_{\text{CLU}} \rceil$, and take their union as the provisional set $\widehat{\mathcal{C}}$. Here, W denotes the final maintained-pool width. The provisional set can be larger because it is formed by merging the two preference shortlists before CLU filtering.

CLU Structural Filtering. After score-based preselection identifies candidates with competitive latency and accuracy, a structural concern remains. Even among these candidates, some may exhibit uneven level consumption across bootstrapping segments. This pattern increases the risk of suboptimal bootstrapping placement during actual execution, especially in residual networks where skip connections introduce level merges. We therefore prefer candidates whose level usage is more even.

Let the network be partitioned into S segments induced by the bootstrapping positions, and let $\mathcal{L}_s$ denote the layer set of segment s . If c_ℓ is the level consumption (multiplicative depth) of layer ℓ , define

$$D_s = \sum_{\ell \in \mathcal{L}_s} c_\ell, \quad \text{CLU}_s = \frac{D_s}{L_{\text{eff}}}, \quad s = 1, \dots, S.$$

To measure variation across adjacent segments, set $\Delta_s = \text{CLU}_{s+1} - \text{CLU}_s$ for $s = 1, \dots, S-1$ and compute

$$\text{RMS}_{\text{adj}} = \sqrt{\frac{1}{S-1} \sum_{s=1}^{S-1} \Delta_s^2}.$$

A smaller RMS_{adj} indicates smoother level usage. We therefore sort $\widehat{\mathcal{C}}$ by increasing RMS_{adj} and retain the top $K_{\text{CLU}} = \lceil \eta_{\text{CLU}} |\widehat{\mathcal{C}}| \rceil$ candidates as $\mathcal{C}_{\text{CLU}}$. Because CLU is applied after score preselection, it filters only candidates that are already competitive in latency or accuracy. Under our default $W = 15$ and $\eta_{\text{CLU}} = 0.95$, $W_{\text{pre}} = 16$ and $|\widehat{\mathcal{C}}|$ is at most 32. CLU is therefore an intentionally conservative structural-safety pruning step: when the two preference shortlists are small or highly overlapping, it may retain all candidates; when the provisional set is sufficiently diverse, it typically removes only the most imbalanced candidate before the pool update.

4.4.3. Pool Update and Termination

After CLU filtering, the pools are updated from $\mathcal{C}_{\text{CLU}}$. The accuracy pool R_{acc} retains the W highest scoring configurations under λ_{acc} , and the latency pool R_{lat} retains the W highest scoring configurations under λ_{lat} ; their union contains at most $2W$ retained configurations for the next iteration. In the implementation used for the reported results, each

search iteration generates one LMAR-updated candidate from each parent configuration after duplicate removal. Since the accuracy-oriented and latency-oriented pools each contain at most W configurations, at most $2W$ candidate networks are CAPR-evaluated in one iteration; with $W = 15$, this cap is 30. If any retained candidate satisfies $\widehat{\text{Acc}}(\mathbf{d}) \geq \text{Acc}_{\text{target}}$, the search terminates early and returns $R_{\text{acc}} \cup R_{\text{lat}}$ as a shortlist for subsequent higher-fidelity training and encrypted evaluation. Otherwise, the search continues until T_{max} iterations and returns the same shortlist. Algorithm 2 summarizes the offline search loop.

Algorithm 2 CipherAPR Offline Search Loop

Require: Initial configuration $\mathbf{d}_{\text{init}}$, available degree set $\mathbb{D}$, pool width W , preference weights $\lambda_{\text{acc}}, \lambda_{\text{lat}}$, CLU filtering ratio η_{CLU} , max iterations T_{max} , accuracy target $\text{Acc}_{\text{target}}$

Ensure: Shortlisted configuration set $\mathcal{P}$

```

1:  $R_{\text{acc}} \leftarrow R_{\text{lat}} \leftarrow \{\mathbf{d}_{\text{init}}\}; W_{\text{pre}} \leftarrow \lceil W/\eta_{\text{CLU}} \rceil$ 
2: for  $t = 1$  to  $T_{\text{max}}$  do
3:    $\mathcal{P} \leftarrow R_{\text{acc}} \cup R_{\text{lat}}$ 
4:   Expand  $\mathcal{P}$  by LMAR-guided probabilistic degree tuning and instantiate the resulting
     candidates with DAAP and MOCT to obtain  $\mathcal{C}$ 
5:   For each  $\mathbf{d}' \in \mathcal{C}$ , estimate  $\hat{t}(\mathbf{d}')$  by the latency proxy and  $\widehat{\text{Acc}}(\mathbf{d}')$  by CAPR
6:    $\hat{\mathcal{C}} \leftarrow \text{TopK}_{\lambda_{\text{acc}}}(\mathcal{C}, W_{\text{pre}}) \cup \text{TopK}_{\lambda_{\text{lat}}}(\mathcal{C}, W_{\text{pre}})$ 
7:   Filter  $\hat{\mathcal{C}}$  by CLU to obtain  $\mathcal{C}_{\text{CLU}}$ 
8:    $R_{\text{acc}} \leftarrow \text{TopK}_{\lambda_{\text{acc}}}(\mathcal{C}_{\text{CLU}}, W); R_{\text{lat}} \leftarrow \text{TopK}_{\lambda_{\text{lat}}}(\mathcal{C}_{\text{CLU}}, W)$ 
9:   if  $\exists \mathbf{d} \in R_{\text{acc}} \cup R_{\text{lat}}$  such that  $\widehat{\text{Acc}}(\mathbf{d}) \geq \text{Acc}_{\text{target}}$  then
10:    return  $R_{\text{acc}} \cup R_{\text{lat}}$ 
11:  end if
12: end for
13: return  $R_{\text{acc}} \cup R_{\text{lat}}$ 

```

4.5. Security and Efficiency Discussion

Before turning to the experiments, we briefly discuss the security properties and system-level efficiency of the proposed pipeline.

4.5.1. Data Security

Our security discussion is limited to confidentiality in the standard FHE inference setting. CipherAPR operates entirely as an offline design-time framework. Its role is to determine the layer-wise polynomial degree assignment $\mathbf{d}^*$ and to instantiate the corresponding layer-wise polynomial approximations before any encrypted inference takes place. The resulting polynomial network $\tilde{f}$ is a fixed public function; it plays the same structural role as the model weights in a standard FHE inference pipeline.

During online inference, CipherAPR changes only the fixed public function evaluated over the ciphertext:

$$\text{ct}_{\text{out}} = \text{Eval}_{\text{pk}}(\tilde{f}, \text{ct}_{\text{in}}),$$

where $\tilde{f}$ is the mixed-degree polynomial network produced offline and pk is the public evaluation key. The underlying encryption, decryption, key switching, and bootstrapping mechanisms remain unchanged. In particular, all components of CipherAPR, namely, LMAR, DAAP, MOCT, CAPR, and CLU, operate on plaintext activations or public model parameters during the offline design phase and have no access to user ciphertexts. Consequently, the semantic security of the scheme is inherited directly from the standard RNS-CKKS parameterization and its underlying hardness assumptions (Ring Learning With Errors). The server at inference time observes only ciphertexts and the public evaluation key, exactly as in any standard FHE inference setting.

The offline activation statistics used by CipherAPR are computed during the model provider's preprocessing stage using training or calibration data and do not require access to users' private inference inputs. During online inference, user inputs remain encrypted, and the server only evaluates the fixed polynomial network under the standard RNS-CKKS procedure. Although the resulting polynomial coefficients may reflect aggregate properties of the calibration distribution, they do not reveal individual encrypted queries, which is consistent with the standard MLaaS/FHE inference setting where model preparation is completed before deployment.

4.5.2. System Efficiency

The offline design space scales exponentially as $|\mathbb{D}|^L$. CipherAPR narrows this space through lightweight candidate screening: the LMAR restricts degree updates to a focused subset of layers; DAAP and MOCT generate reusable polynomial coefficients for shared degree/domain settings; the latency proxy avoids encrypted execution during screening; and CAPR provides a low-budget accuracy signal before full training. The dominant offline cost is therefore CAPR-based fine-tuning of candidate networks, whereas the LMAR, the latency proxy, and CLU remain lightweight computations over calibration statistics or bootstrapping segments.

Figure 4 links each component to the specific efficiency bottleneck it addresses in the offline search loop.

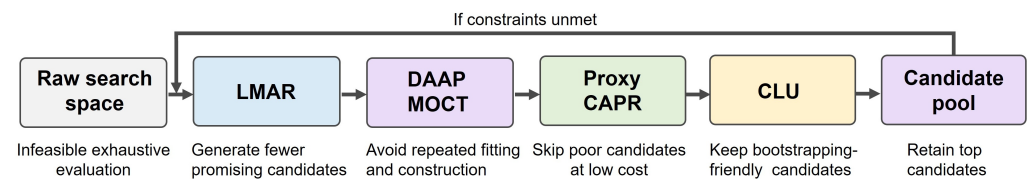


Figure 4. System-efficiency path of CipherAPR's offline search. Each block shows the action of one component and the corresponding efficiency effect discussed in Section 4.5.2.

5. Cryptographic and Experimental Setup

We separate offline design from online encrypted execution. Training, degree allocation, and polynomial instantiation are performed offline in Python 3.8 with PyTorch (1.13.1) on an NVIDIA RTX A6000 GPU under Ubuntu 20.04. Ciphertext inference is implemented in C++ on top of AutoFHE using the Microsoft SEAL library [34] and executed on an Intel Xeon Platinum 8462Y+ processor with 128 vCPUs, 1 TiB of memory, and Ubuntu 20.04. This separation reflects the intended deployment model of CipherAPR. Only encrypted execution is timed online, whereas all design procedures are completed offline.

5.1. Datasets

We use CIFAR-10 and CIFAR-100 [35] as benchmarks for encrypted inference. Each dataset contains 50,000 training images and 10,000 test images of size 32×32 pixels. To accelerate the offline design loop, we use a fast-training subset of 5000 images sampled from the training set, while ciphertext inference is evaluated on the full 10,000-image test set. We report top-1 accuracy only to ensure that the encrypted inference comparisons are made at comparable accuracy targets.

5.2. Parameters

We fix a common encrypted-execution setting for all methods, summarized in Table 4. In the approximation pipeline, the global normalization factor is set to $B = 80$. For RNS-CKKS execution, we use ring degree $N = 2^{16}$, private-key Hamming weight 192,

and a total modulus-chain length of 1552 bits, corresponding to the MPCNN-style chain $51 + 46 \times 16 + 51 \times 14 + 51$, which satisfies 128-bit security [36,37].

Table 4. RNS-CKKS parameter setting used in encrypted inference.

Parameter	Value
Ring degree N	2^{16}
Numerical scaling factor B	80
Total modulus-chain length	1552 bits
Private-key Hamming weight	192
Security level	128-bit

For inference, we compare CipherAPR with MPCNN [16] and AutoFHE [22] under the same RNS-CKKS setting and polynomial evaluation method. These baselines address the same structural-design problem as CipherAPR. They decide the polynomial network to be evaluated under FHE. DaCapo, Orion, and ReSBM [24–26] are compiler-side execution optimizers for an already-fixed polynomial computation graph, rather than methods for searching the layer-wise polynomial-degree structure. We therefore cite them as execution-side optimization methods but do not include them as structural-design baselines.

The original networks are trained with SGD for 200 epochs using batch size 128, initial learning rate 0.1 with cosine decay, momentum 0.9, weight decay 0.0005, and gradient clipping 1.0. The polynomial networks are trained for 300 epochs with initial learning rate 0.08, cosine decay, and at least 8 epochs per stage.

For the final CipherAPR training, we use batch size 128, learning rate 0.01, momentum 0.9, weight decay 5×10^{-4} , gradient clipping 0.1, and a Kullback–Leibler (KL) divergence weight $\tau = 0.9$. For CAPR-based accuracy estimation, each evaluated configuration is fine-tuned for 20 total epochs with at least 2 epochs per stage. This budget is used only to provide a low-cost staged accuracy signal during search.

For polynomial instantiation, we set the approximation-domain lower bound to $a = 0.2$, use sigmoid decay with extension factor $\rho_{\text{ext}} = 1.5$, and use 30 and 100 images for CIFAR-10 and CIFAR-100 during domain estimation, with uniform perturbation $\varepsilon = 0.1$. The MOCT model is trained for 10,000 epochs by updating the odd-degree coefficients in $p_1(x)$ while keeping the remaining coefficients fixed, using regularization weights $w_1 = 3 \times 10^7$, $w_2 = 1 \times 10^4$, $w_3 = 1 \times 10^5$, and $w_4 = 1$. These weights balance the different numerical scales of the loss terms while prioritizing the core approximation objectives.

The search hyperparameters are kept consistent across benchmark settings unless otherwise stated. We set the maintained-pool width to $W = 15$, the maximum number of search iterations to $T_{\text{max}} = 30$, and the CLU retention ratio to $\eta_{\text{CLU}} = 0.95$. The preference weights are initialized as $\lambda_{\text{acc}}^{(0)} = 1.0$ and $\lambda_{\text{lat}}^{(0)} = 0.1$, with the latency-oriented accuracy weight annealed during search. The LMAR threshold is determined by Otsu’s method; these search-time controls do not change the final encrypted inference procedure.

6. Evaluation

This section presents the evaluation goals, research questions, and results.

6.1. Evaluation Goals and Research Questions

This section focuses on the cryptographic execution behavior of CipherAPR under a fixed RNS-CKKS setting while retaining two supporting analyses that directly justify its degree-allocation and polynomial-instantiation mechanisms. We first examine whether CipherAPR reduces bootstrapping count and end-to-end encrypted inference latency while preserving comparable accuracy. We then evaluate the individual components: (i) whether

LMAR provides an effective importance signal for degree allocation, (ii) whether DAAP and MOCT improve polynomial approximation quality, and (iii) whether the evaluation-and-filtering stage, including a relative latency proxy, score-based preselection, CLU-based structural filtering, and CAPR-based accuracy estimation, provides useful offline screening guidance. Based on these goals, we pose the following research questions.

- RQ1: Under fixed RNS-CKKS parameters and comparable accuracy targets, does CipherAPR reduce bootstrapping count and end-to-end encrypted inference latency relative to prior baselines?
- RQ2: During progressive activation replacement, does LMAR-based prioritization recover inference accuracy faster under the same replacement budget than baseline orderings?
- RQ3: Do DAAP and MOCT reduce ReLU approximation errors relative to classical Minimax and untuned variants, thereby improving polynomial instantiation quality under a fixed degree budget?
- RQ4: Do the relative latency proxy, score-based preselection, and CLU-based structural filtering provide useful guidance for bootstrapping-sensitive offline configuration evaluation, and can CAPR provide a practical low-budget staged accuracy estimate in this design loop?

The research questions are designed to isolate the contribution of each component. RQ2 evaluates the LMAR as an importance signal for degree allocation by comparing replacement orderings under a fixed budget. RQ3 evaluates DAAP and MOCT for polynomial-instantiation quality through approximation-error comparisons and a weight-zero ablation. RQ4 evaluates the offline screening stage, including the latency proxy, CAPR, and CLU, through search-trajectory and candidate-reduction statistics. Together, these component-level analyses complement the end-to-end encrypted inference evaluation in RQ1.

6.2. RQ1: End-to-End Encrypted Inference Latency and Bootstrapping Count

Table 5 reports end-to-end encrypted inference performance under paired settings with lower and higher bootstrapping demand. On the evaluated ResNet models, CipherAPR generally preserves encrypted accuracy comparable to AutoFHE while reducing bootstrapping count and latency. The most pronounced gain appears on CIFAR-10/ResNet20 under the higher-bootstrapping setting, where latency decreases from 4659 s to 3341 s (a $1.39\times$ speedup), while ciphertext accuracy remains competitive (92.61% versus AutoFHE's 92.89%). Similar latency reductions occur on CIFAR-10/ResNet32 (7294 s to 6064 s, $1.20\times$) and CIFAR-100/ResNet32 (7187 s to 5953 s), where ciphertext accuracy even slightly improves. This trend is equally reflected in amortized latency: for example, under the highlighted higher-bootstrapping settings, it drops from 36.40 s to 26.10 s on CIFAR-10/ResNet20 and from 22.12 s to 18.34 s on CIFAR-100/VGG11. These results indicate that CipherAPR consistently delivers meaningful latency gains without sacrificing encrypted accuracy.

Table 6 decomposes runtime into bootstrapping, linear, and polynomial components. The main speedups are consistent with reduced bootstrapping demand: for example, on CIFAR-10/ResNet20 the bootstrapping count drops from 11 to 7, and on CIFAR-10/ResNet32 it drops from 19 to 13. In the latter case, the linear cost increases from 1838 s to 2208 s because fewer bootstraps can lengthen uninterrupted computation segments, causing some rotations, rescaling steps, and key-switching-related operations to run at higher ciphertext levels with more active RNS primes. Nevertheless, the bootstrapping-time reduction remains larger than this linear-cost increase, so the end-to-end latency still improves. This confirms that CipherAPR optimizes bootstrapping-sensitive execution; individual runtime components may increase when the overall bootstrapping plan improves.

Table 5. Performance of RNS-CKKS encrypted inference on ResNet/VGG under a fixed parameter setting. Amort. Lat. (s) denotes the amortized latency per image, computed by dividing the total packed-batch inference latency by the number of images processed in one packed batch.

Dataset	Network	Plain Acc (%)	MPCNN [16]				AutoFHE [22]				CipherAPR			
			#Boot	Cipher Acc (%)	Latency (s)	Amort. Lat. (s)	#Boot	Cipher Acc (%)	Latency (s)	Amort. Lat. (s)	#Boot	Cipher Acc (%)	Latency (s)	Amort. Lat. (s)
CIFAR-10	ResNet20	92.66	18	92.59	6299	49.21	5	91.92	2751	21.49	4	92.15	2291	17.90
							11	92.89	4659	36.40	7	92.61	3341	26.10
	ResNet32	93.46	30	93.39	10,231	79.93	8	93.46	4664	36.44	7	93.14	4129	32.26
							19	93.56	7294	56.98	13	93.76	6064	47.38
CIFAR-100	ResNet44	93.72	42	93.57	15,163	118.46	8	92.55	5137	40.13	8	93.04	4728	36.94
							22	93.65	8639	67.49	16	93.71	7250	56.64
	VGG11	90.53	9	90.41	3157	24.66	4	90.68	1931	15.09	4	90.67	1835	14.34
	ResNet32	70.81	30	70.59	10,357	80.91	16	71.34	7187	56.15	12	71.48	5953	46.51
	VGG11	64.64	9	63.95	3200	25.00	7	64.31	2832	22.12	6	64.26	2348	18.34

Table 6. Execution breakdown of RNS-CKKS encrypted inference on ResNet/VGG.

Dataset	Network	MPCNN [16]					AutoFHE [22]					CipherAPR				
		#Boot	Total Eff. #L	Boot (s)	Linear (s)	Poly (s)	#Boot	Total Eff. #L	Boot (s)	Linear (s)	Poly (s)	#Boot	Total Eff. #L	Boot (s)	Linear (s)	Poly (s)
CIFAR-10	ResNet20	18	306	4813	786	700	5	85	1370	1304	77	4	73	1095	1162	34
							11	150	2995	1411	253	7	104	1907	1335	99
	ResNet32	30	498	7813	1336	1082	8	96	2234	2404	26	7	118	1955	2129	45
							19	235	5040	1838	416	13	195	3449	2208	407
CIFAR-100	ResNet44	42	690	11,638	1893	1632	8	120	2171	2940	26	8	131	2149	2550	29
							22	277	5715	2492	432	16	234	4157	2662	431
	VGG11	9	160	2326	483	348	4	72	1125	692	114	4	68	1131	629	75
	ResNet32	30	498	7809	1466	1082	16	230	4377	2408	402	12	176	3183	2357	413
	VGG11	9	160	2332	511	357	7	104	1885	719	228	6	86	1616	611	121

We also evaluate CipherAPR on VGG11 under CIFAR-10/CIFAR-100. The gains on VGG11 remain favorable but are smaller and less consistently tied to large bootstrapping-time reductions, which is consistent with its shorter encrypted execution path. Appendix B provides additional depth profiles for representative mixed-degree networks.

The CIFAR-10 ResNet32 higher-bootstrapping setting further illustrates how structural design affects execution demand. AutoFHE ResNet32(C10, BT19) and CipherAPR ResNet32(C10, BT13) are both evaluated under the same encrypted backend, but their degree profiles lead to different effective-level requirements. In this experimental setting, each bootstrapping operation restores the ciphertext to at most 16 levels, and the input ciphertext starts from level 18. For a run with N_{boot} bootstrapping operations, the total available level budget is $L_{\text{avail}} = 16N_{\text{boot}} + 18$, and the unused budget can be estimated as $L_{\text{avail}} - L_{\text{eff}}^{\text{total}}$, where $L_{\text{eff}}^{\text{total}}$ is the Total Eff. #L reported in Table 6. Under this calculation, AutoFHE ResNet32(C10,BT19) has $19 \times 16 + 18 = 322$ available levels but uses 235 effective levels, leaving 87 levels unused. CipherAPR ResNet32(C10,BT13) has $13 \times 16 + 18 = 226$ available levels and uses 195 effective levels, leaving only 31 levels unused. Across the paired AutoFHE/CipherAPR settings in Table 6, CipherAPR has a smaller unused level budget in seven of nine cases, with the clearest pattern appearing on the ResNet models. This pattern is consistent with CipherAPR's level-aware approximation design. Polynomial degrees are not selected only by approximation accuracy or bootstrapping count but are also screened by CLU to favor level-consumption profiles that better fit bootstrapping segments. The approximation stage therefore incorporates restored-level utilization as an execution-side factor without invoking a full compiler scheduler. This helps explain why the bootstrapping-time reduction dominates the moderate increase in linear-operation cost.

6.3. RQ2: LMAR-Based Degree-Prioritization Analysis

We next evaluate the LMAR as an offline prioritization signal for budgeted degree allocation. The goal is to test whether the LMAR can identify the activation layers whose early polynomial replacement restores network accuracy most effectively, because such layers are the most worthwhile targets when the number of degree adjustments must be kept limited under ciphertext constraints.

We conduct this analysis on ResNet32 with CIFAR-10. Let K denote the total number of activation sites in the network. At each step, we construct a partially replaced network by enabling exactly $k \in \{1, \dots, K\}$ activation sites as polynomials while keeping the remaining $K - k$ sites as identity. We compare three replacement orders, namely, Forward, LMAR-based, and Reverse-LMAR. After each step, we apply the same fine-tuning procedure and report inference accuracy.

In Figure 5a, LMAR-based prioritization yields the fastest accuracy recovery. The advantage is most evident at small k , where the LMAR curve rises earlier than the Forward and Reverse-LMAR orderings. This trend suggests that, under a limited replacement budget, the LMAR tends to prioritize the activation sites whose early polynomial replacement contributes most to accuracy recovery. Some partially replaced models even reach 94.05% accuracy after the same fine-tuning procedure, slightly exceeding the original ReLU network baseline.

Figure 5b shows that a similar advantage remains when higher-degree polynomials are used, suggesting that the LMAR ordering is not tied to only one replacement setting. Figure 5c further shows that, starting from the best second-degree configuration at $k = 13$, selectively raising the degree of additional activations yields further gains before the improvement gradually saturates.

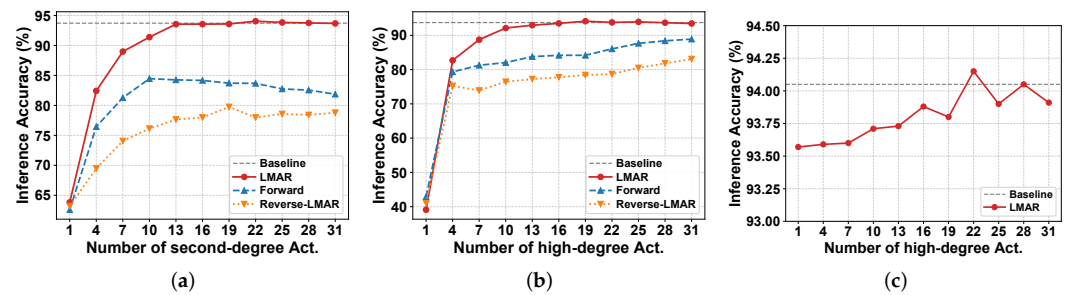


Figure 5. Inference accuracy during incremental activation replacement under different prioritization orders: (a,b) show replacement with second-degree and higher-degree polynomials; (c) shows selective high-degree replacement starting from the best configuration obtained in (a).

These findings support the LMAR as a practical, budget-aware prioritization heuristic for offline degree allocation. Under a constrained replacement budget, LMAR-guided ordering achieves competitive accuracy with fewer early replacements than the Forward and Reverse-LMAR baselines, confirming that near-zero activation concentration is an informative signal for identifying the most impactful layers.

6.4. RQ3: Approximation Quality of Ciphertext-Safe Polynomial Instantiation

All approximation errors in this subsection are measured on the final ReLU approximation $g(x) = x \cdot (0.5 + 0.5p(x))$. The intermediate sign polynomial alone is not used for these error metrics. Accordingly, the reported depth refers to the multiplicative depth needed to evaluate the ReLU approximation under the adopted composite polynomial construction.

Table 7 compares Minimax, DAAP alone (w/o MOCT), and DAAP + MOCT under the same ReLU approximation depth budgets. DAAP + MOCT lowers MaxE by 39.44–85.39% relative to Minimax and is also competitive in SSE, supporting its use for ciphertext-safe activation replacement. Table 8 then isolates the DAAP domain parameters before coefficient tuning, where core denotes $[-v, v]$ and outer denotes $[-\rho_{\text{ext}}v, -v] \cup [v, \rho_{\text{ext}}v]$.

Table 7. Approximation error of final ReLU polynomials under fixed ReLU approximation multiplicative depths. SSE and MaxE denote sum of squared-error and maximum-error metrics.

Depth	Minimax [28]		w/o MOCT		MOCT		MaxE Improvement (%)	
	SSE	MaxE	SSE	MaxE	SSE	MaxE	vs. Minimax	vs. w/o MOCT
4	7.55×10^{-1}	6.65×10^{-2}	3.04×10^{-1}	1.42×10^{-2}	1.85×10^{-1}	9.72×10^{-3}	85.39	31.54
5	3.10×10^{-1}	3.44×10^{-2}	7.96×10^{-2}	7.18×10^{-3}	7.70×10^{-2}	6.76×10^{-3}	80.36	5.90
6	3.73×10^{-2}	1.51×10^{-2}	7.39×10^{-2}	9.37×10^{-3}	2.62×10^{-2}	3.62×10^{-3}	76.03	61.37
7	4.92×10^{-3}	7.58×10^{-3}	3.41×10^{-3}	2.62×10^{-3}	2.26×10^{-3}	2.41×10^{-3}	68.22	8.08
8	6.59×10^{-4}	3.82×10^{-3}	7.50×10^{-4}	1.85×10^{-3}	6.77×10^{-4}	1.71×10^{-3}	55.33	7.74
9	1.05×10^{-4}	1.87×10^{-3}	2.42×10^{-4}	1.25×10^{-3}	1.99×10^{-4}	1.13×10^{-3}	39.44	9.73
10	9.03×10^{-5}	1.73×10^{-3}	4.29×10^{-5}	5.74×10^{-4}	4.06×10^{-5}	5.72×10^{-4}	66.91	0.33
11	1.02×10^{-5}	9.03×10^{-4}	8.11×10^{-6}	3.88×10^{-4}	1.58×10^{-6}	3.82×10^{-4}	57.66	1.40
12	1.35×10^{-6}	4.72×10^{-4}	1.30×10^{-6}	1.88×10^{-4}	1.19×10^{-6}	1.82×10^{-4}	61.51	3.27

The DAAP sweep shows a clear error–coefficient trade-off. For example, at depth 6, increasing v from 0.12 to 0.30 reduces the maximum coefficient from 3.89×10^2 to 2.54×10^1 but increases Core MaxE from 2.85×10^{-3} to 2.91×10^{-2} . Increasing ρ_{ext} similarly reduces the coefficient scale while making the outer interval harder to approximate. We therefore use $(v, \rho_{\text{ext}}) = (0.20, 1.50)$ as a balanced default. Table 9 reports MOCT weight-zero ablations under this default domain, with MOCT tuning only the odd-degree coefficients of $p_1(x)$.

Table 8. DAAP-only sensitivity of domain parameters. Rows with different v fix $\rho_{\text{ext}} = 1.50$, and rows with different ρ_{ext} fix $v = 0.20$.

Depth	Setting	Core MaxE	Outer MaxE	Max $ a_i $
6	$v = 0.12, \rho_{\text{ext}} = 1.50$	2.85×10^{-3}	2.19×10^{-3}	3.89×10^2
6	$v = 0.20, \rho_{\text{ext}} = 1.50$	9.57×10^{-3}	1.36×10^{-2}	8.47×10^1
6	$v = 0.30, \rho_{\text{ext}} = 1.50$	2.91×10^{-2}	4.18×10^{-2}	2.54×10^1
12	$v = 0.12, \rho_{\text{ext}} = 1.50$	1.49×10^{-4}	2.55×10^{-7}	2.77×10^6
12	$v = 0.20, \rho_{\text{ext}} = 1.50$	1.88×10^{-4}	4.00×10^{-5}	7.84×10^4
12	$v = 0.30, \rho_{\text{ext}} = 1.50$	4.28×10^{-4}	6.72×10^{-4}	4.66×10^3
6	$v = 0.20, \rho_{\text{ext}} = 1.05$	3.87×10^{-3}	4.21×10^{-3}	2.46×10^2
6	$v = 0.20, \rho_{\text{ext}} = 1.50$	9.57×10^{-3}	1.36×10^{-2}	8.47×10^1
6	$v = 0.20, \rho_{\text{ext}} = 2.50$	1.81×10^{-2}	5.37×10^{-2}	1.86×10^1
12	$v = 0.20, \rho_{\text{ext}} = 1.05$	1.61×10^{-4}	1.63×10^{-6}	8.83×10^5
12	$v = 0.20, \rho_{\text{ext}} = 1.50$	1.88×10^{-4}	4.00×10^{-5}	7.84×10^4
12	$v = 0.20, \rho_{\text{ext}} = 2.50$	4.27×10^{-4}	1.08×10^{-3}	4.53×10^3

Table 9. Default MOCT result and weight-zero ablation under the default DAAP domain. Depth denotes ReLU approximation multiplicative depth; depth 6 uses sign-polynomial degree set [3, 7], and the depth-12 summary uses sign-polynomial degree set [7, 7, 27].

Depth	Case	Core MaxE	Outer MaxE	Max $ a_i $
6	DAAP baseline	9.57×10^{-3}	1.36×10^{-2}	8.47×10^1
6	MOCT default	6.60×10^{-3}	1.84×10^{-2}	8.45×10^1
6	$w_1 = 0$	9.57×10^{-3}	1.36×10^{-2}	8.47×10^1
6	$w_2 = 0$	6.60×10^{-3}	1.84×10^{-2}	8.45×10^1
6	$w_3 = 0$	3.40×10^{-3}	7.12×10^{-2}	1.04×10^2
6	$w_4 = 0$	6.60×10^{-3}	1.84×10^{-2}	8.45×10^1
12	DAAP baseline and all $w_i = 0$ cases	1.88×10^{-4}	3.99×10^{-5}	7.84×10^4

At depth 6, MOCT default reduces Core MaxE from 9.57×10^{-3} to 6.60×10^{-3} with a nearly unchanged coefficient scale. The $w_1 = 0$ row reverts to the DAAP baseline, showing that CE drives the core-domain update. By contrast, $w_2 = 0$ and $w_4 = 0$ are nearly identical to the MOCT default in this balanced setting. The EDC and CR terms still serve as safeguards: w_2 mainly guards against extreme numerical distributions that cause extended-domain output violations, while w_4 mainly addresses extreme coefficient magnitudes. Under the default DAAP domain, such extreme cases are not prominent, so their ablation effect is limited. The $w_3 = 0$ row lowers Core MaxE but increases Outer MaxE to 7.12×10^{-2} and raises the maximum coefficient to 1.04×10^2 , showing that CP helps avoid a core-only fit with poorer outer-domain behavior. At depth 12, the DAAP baseline is already stable and the weight-zero cases remain unchanged.

6.5. RQ4: Evaluation of the Offline Screening and Filtering Stage

Figure 6 and Table 10 evaluate the execution-oriented screening stage. Figure 6a shows that the latency proxy preserves the within-setting ordering of measured encrypted latency across bootstrapping settings. The proxy serves as a within-setting ranking signal; global latency calibration is outside its role. Figure 6b shows that CAPR's progressive replacement quickly separates trainable candidates from the failure regime under a small training budget.

Table 10 summarizes representative search trajectories with and without CLU-based structural filtering. With CLU, the search reaches 99.9% and 100.0% normalized best-so-far scores in the accuracy- and latency-oriented settings after 210 CAPR evaluations, compared with 97.6% and 95.5% without CLU. Under $W = 15$ and $\eta_{\text{CLU}} = 0.95$, CLU is conservative: the retained search state contains at most $2W$ configurations, and CLU only acts as a structural-safety pruning signal within the score-preselected provisional set. The final best scores remain nearly identical, indicating that CLU mainly improves the path-dependent early search trajectory toward a comparable best-so-far score.

The dominant offline cost is CAPR-based candidate evaluation. Each candidate is fine-tuned for $E_{\text{CAPR}} = 20$ CAPR epochs on the 5000-image fast-training subset with batch size 128, corresponding to 800 mini-batch updates per candidate. We estimate the serial offline search cost as

$$T_{\text{search}} \approx N_{\text{eval}} T_{\text{CAPR}}^{20},$$

where N_{eval} is the number of CAPR-evaluated configurations and T_{CAPR}^{20} is the per-candidate 20-epoch CAPR time. In pilot runs, T_{CAPR}^{20} is on the order of 10–25 min for ResNet settings and 20–30 min for VGG settings. Therefore, a 180–210-evaluation CLU-guided run corresponds roughly to 1–3 serial days for ResNet-type models and 2–4 serial days for VGG-type models. The without-CLU setting scales proportionally with the larger number of evaluated candidates. These numbers are serial cost estimates. Full-search wall-clock measurements require end-to-end search logs. Since candidate evaluations are independent and each iteration evaluates at most $2W = 30$ candidates, practical wall-clock time can be reduced by candidate-level parallelism when sufficient workers are available.

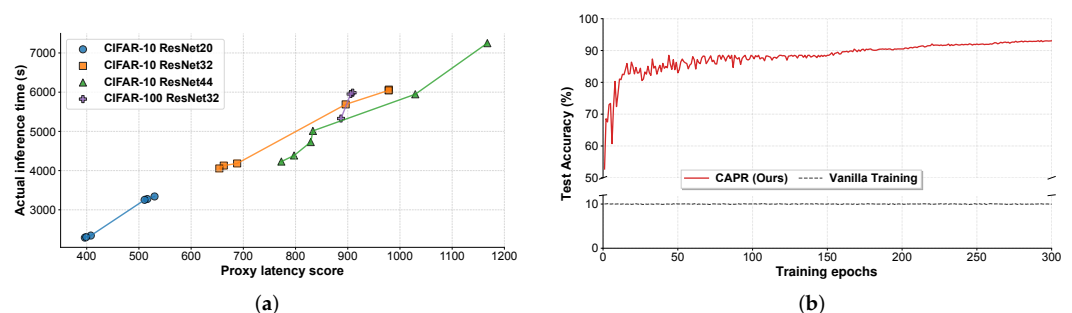


Figure 6. Offline screening components. (a) Relative latency proxy under different bootstrapping settings. (b) CAPR-based low-budget staged accuracy estimation.

Table 10. Representative search trajectories with and without conservative CLU structural filtering. Scores are normalized within each pool setting, and score cells report CLU/without CLU.

Pool Setting	Best-So-Far Norm. Score After 210 CAPR Evals	Normalized Final Best Score	CAPR Evals to Reach CLU-Run Best	Path-Dependent CAPR Eval. Saving
Accuracy-oriented	99.9%/97.6%	99.9%/100.0%	210/330	36.4% fewer
Latency-oriented	100.0%/95.5%	100.0%/100.0%	180/330	45.5% fewer

7. Conclusions

We presented CipherAPR, an importance-guided framework for level-aware mixed-degree polynomial design in RNS-CKKS encrypted inference. Building on recent results from FHE compiler research, we argued that optimizing degree assignments for bootstrapping count alone is insufficient. A degree configuration should also align with subsequent bootstrapping placement and restored-level budgets. CipherAPR addresses this design-stage problem through LMAR-guided degree allocation on accuracy-sensitive layers, DAAP- and MOCT-based polynomial instantiation for reusable and ciphertext-safe

approximations, and lightweight evaluation via a latency proxy, CAPR, and conservative CLU-based structural filtering.

The present study focuses on offline structural design under a fixed encrypted backend. In this scope, CipherAPR does not replace compiler-side scheduling. Instead, it produces mixed-degree polynomial networks whose level-consumption profiles are more suitable for downstream execution planning. Coupling such generated degree profiles with compiler-side scheduling is therefore a natural next step for studying end-to-end co-optimization across structural design and encrypted execution.

Experiments on ResNet and VGG models under different bootstrapping settings show that, at comparable encrypted accuracy, CipherAPR achieves a $1.09\times$ – $1.39\times$ speedup over AutoFHE. The gains are larger when bootstrapping demand is heavier, and the execution breakdown shows that CipherAPR often leaves a smaller unused restored-level budget. These results support using restored-level utilization as a design-stage signal rather than treating bootstrapping count as the only execution proxy. Supporting analyses further show that the proposed components play complementary roles. The LMAR provides a practical dynamic importance signal for degree tuning, DAAP and MOCT improve approximation quality and ciphertext-side numerical stability under fixed degree budgets, and the evaluation-and-filtering stage guides the offline search toward configurations that better fit subsequent bootstrapping optimization.

Overall, the results indicate that level-aware mixed-degree polynomial design is a practical direction for accelerating FHE inference under fixed cryptographic settings. Scaling to substantially deeper CNNs such as ResNet-110 may require more efficient accuracy proxies or hybrid importance metrics, because both the raw degree-assignment space and the CAPR evaluation cost grow with the number of activation layers. Extending CipherAPR to transformer architectures remains another direction, since softmax, layer normalization, and attention blocks introduce different polynomial-approximation and level-consumption patterns.

Author Contributions: Conceptualization, J.W., J.X., Z.L.J. and B.F.; methodology, J.W., Y.L. (Yucen Liao), Y.L. (Yinglong Liao) and Z.T.; software, J.W.; validation, J.W., Y.L. (Yucen Liao) and Z.T.; formal analysis, J.W., Y.L. (Yinglong Liao) and Z.T.; investigation, J.W.; resources, Z.L.J. and B.F.; data curation, J.W.; writing—original draft preparation, J.W.; writing—review and editing, J.W., Y.L. (Yucen Liao), Y.L. (Yinglong Liao), Z.T., J.X., Z.L.J. and B.F.; visualization, Y.L. (Yucen Liao), Y.L. (Yinglong Liao) and Z.T.; supervision, Z.L.J. and B.F.; project administration, Z.L.J. and B.F.; funding acquisition, Z.L.J. and B.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key R&D Program of China (2023YFB3106500), the National Natural Science Foundation of China (62272131), the Shenzhen Social Science Fund (SZ2025B011), and the Shenzhen Science and Technology Major Project (KJZD20230923114908017).

Data Availability Statement: The data presented in this study are available from the corresponding author upon reasonable request. Public datasets used in this study are CIFAR-10 and CIFAR-100.

Conflicts of Interest: The authors declare no conflict of interest. The funding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

Appendix A. Theoretical Justification for LMAR-Based Tuning

The strategy of prioritizing higher-degree polynomial approximations for layers with a high $\text{LMAR}_\ell(\epsilon)$ to more efficiently improve network classification accuracy is based on the following theoretical analysis.

Theoretical Bounds and Allocation Strategy

Proposition A1 (Upper Bound on Probability of Sign Inconsistency). *Let the pre-activation of the ℓ -th layer be the random variable X_ℓ . If the polynomial $p_d(x)$ satisfies the out-of-band shape constraints, i.e., it maintains the correct sign when $|x| \geq \epsilon$, then the probability of the gating sign inconsistency event $\mathcal{E}_{\text{flip}} = \{\text{sgn}(p_d(X_\ell)) \neq \text{sgn}(X_\ell)\}$ satisfies*

$$\Pr(\mathcal{E}_{\text{flip}}) \leq \Pr(|X_\ell| < \epsilon) = \text{LMAR}_\ell(\epsilon).$$

This proposition shows that the gating risk introduced by the polynomial approximation is strictly bounded by $\text{LMAR}_\ell(\epsilon)$. In other words, the LMAR directly quantifies the probability that a layer's activation state is misjudged due to an approximation error near zero.

Proposition A2 (Upper Bound on Loss induced by Approximation Error). *Let $c_{\epsilon,d} = \sup_{|x| < \epsilon} |p_d(x)|$ be the maximum deviation within the approximation domain. Assuming the subsequent sub-network and the loss function are L_ℓ -Lipschitz and L_{loss} -Lipschitz continuous, respectively, the change in expected loss is bounded by*

$$\begin{aligned} \mathbb{E} \Delta \mathcal{L} \leq & L_{\text{loss}} L_\ell \left(\mathbb{E}[|p_d(X_\ell) - r(X_\ell)| \mathbf{1}\{X_\ell \geq 0\}] \right. \\ & \left. + c_{\epsilon,d} \text{LMAR}_\ell(\epsilon) \right). \end{aligned}$$

The derived upper bound implies that the global loss degradation depends on both the layer sensitivity (L_ℓ) and the local approximation error near zero. Although explicitly computing L_ℓ in deep networks is computationally intractable, the bound clarifies why $\text{LMAR}_\ell(\epsilon)$ is informative for degree allocation: for a fixed approximation family and comparable downstream sensitivity, a larger concentration of activations near zero enlarges the contribution of approximation error to the expected loss. Therefore, the LMAR serves as a principled importance signal for prioritizing which layers should receive degree increases first. This interpretation supports the LMAR-based ranking used in the degree-allocation stage of CipherAPR, without introducing a separate optimization objective beyond the probabilistic update rule described in the main text.

Appendix B. Multiplicative Depth of Polynomial Networks

Figure A1 reports the per-layer multiplicative depths of representative CipherAPR instantiations, where 0 denotes pruning, 2 denotes a shallow polynomial, and values greater than 2 denote deeper approximations. Larger bootstrapping budgets usually allow for more layers to take a depth greater than 2, but the resulting profile remains selective and non-uniform. For example, on ResNet32 with CIFAR-10, the number of depth-greater-than-2 activations increases from 2 under the 7-bootstrapping setting to 12 under the 13-bootstrapping setting, while several layers still remain pruned or stay at depth 2. This pattern indicates that CipherAPR does not apply one fixed polynomial degree across the network. Instead, it allocates pruning, shallow approximation, and deeper approximation to different activation sites according to the model and the available ciphertext budget.

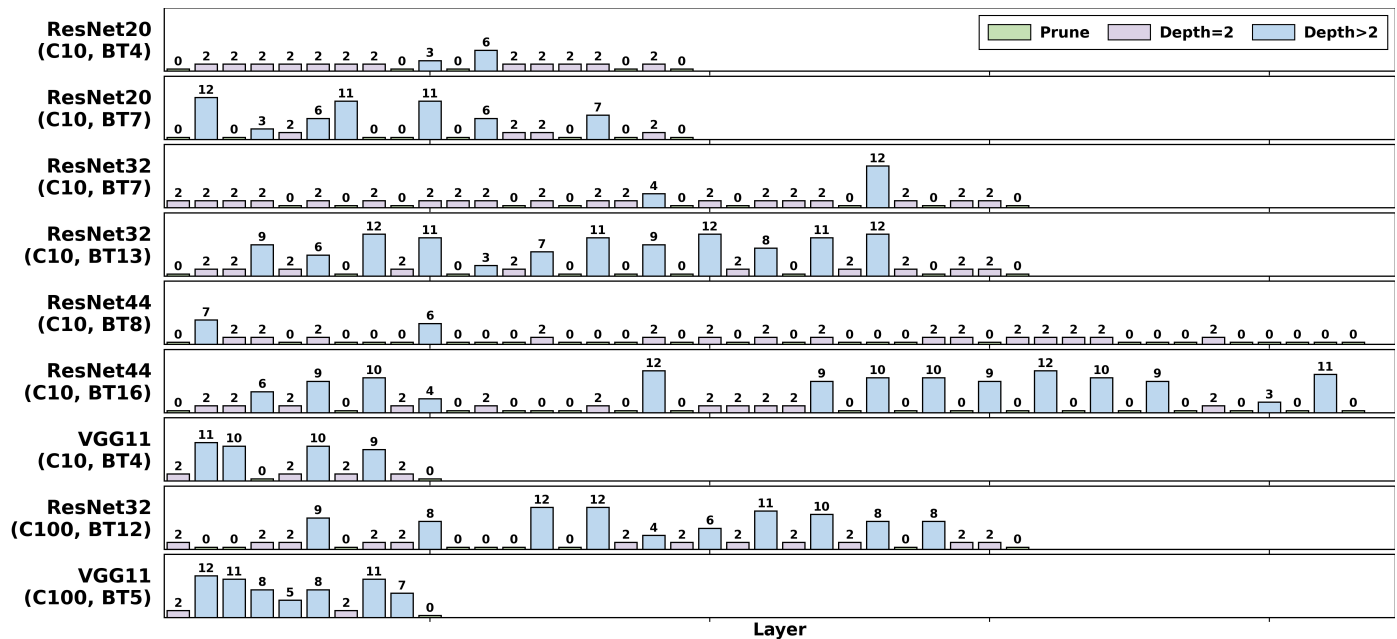


Figure A1. Per-layer multiplicative depths assigned to activation polynomials under representative configurations with different bootstrapping counts. A depth value of 0 denotes a pruned or identity activation, 2 denotes a shallow polynomial replacement, and larger values denote higher-depth composite polynomial approximations.

References

- Hesamian, M.H.; Jia, W.; He, X.; Kennedy, P. Deep learning techniques for medical image segmentation: achievements and challenges. *J. Digit. Imaging* **2019**, *32*, 582–596. [\[CrossRef\]](#) [\[PubMed\]](#)
- Ozbayoglu, A.M.; Gudelek, M.U.; Sezer, O.B. Deep learning for financial applications: A survey. *Appl. Soft Comput.* **2020**, *93*, 106384. [\[CrossRef\]](#)
- Tanuwidjaja, H.C.; Choi, R.; Baek, S.; Kim, K. Privacy-Preserving Deep Learning on Machine Learning as a Service—A Comprehensive Survey. *IEEE Access* **2020**, *8*, 167425–167447. [\[CrossRef\]](#)
- European Union. General Data Protection Regulation. In *Regulation (EU)*; Publications Office of the European Union: Luxembourg, 2016; Volume 679.
- Pardau, S.L. The california consumer privacy act: Towards a european-style privacy regime in the united states. *J. Technol. Law Policy* **2018**, *23*, 68.
- Lam, K.; Lu, X.; Zhang, L.; Wang, X.; Wang, H.; Goh, S.Q. Efficient FHE-Based Privacy-Enhanced Neural Network for Trustworthy AI-as-a-Service. *IEEE Trans. Dependable Secur. Comput.* **2024**, *21*, 4451–4468. [\[CrossRef\]](#)
- Huang, Z.; Lu, W.; Hong, C.; Ding, J. Cheetah: Lean and fast secure Two-Party deep neural network inference. In *Proceedings of the 31st USENIX Security Symposium (USENIX Security 22)*, Boston, MA, USA, 10–12 August 2022; USENIX Association: Berkeley, CA, USA, 2022; pp. 809–826.
- Pang, Q.; Zhu, J.; Möllering, H.; Zheng, W.; Schneider, T. Bolt: Privacy-preserving, accurate and efficient inference for transformers. In *Proceedings of the 2024 IEEE Symposium on Security and Privacy (SP)*; IEEE: New York, NY, USA, 2024; pp. 4753–4771.
- Yao, A.C. How to generate and exchange secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science (SFOCS 1986)*; IEEE: New York, NY, USA, 1986; pp. 162–167.
- Beaver, D. Efficient multiparty protocols using circuit randomization. In *Proceedings of the Advances in Cryptology—CRYPTO’91: Proceedings 11*; Springer: Berlin/Heidelberg, Germany, 1992; pp. 420–432.
- Gentry, C. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, Bethesda, MD, USA, 31 May–2 June 2009*; ACM: San Jose, CA, USA, 2009; pp. 169–178.
- Chillotti, I.; Gama, N.; Georgieva, M.; Izabachène, M. TFHE: Fast fully homomorphic encryption over the torus. *J. Cryptol.* **2020**, *33*, 34–91.
- Cheon, J.H.; Kim, A.; Kim, M.; Song, Y. Homomorphic encryption for arithmetic of approximate numbers. In *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 409–437.
- Marcolla, C.; Sucasas, V.; Manzano, M.; Bassoli, R.; Fitzek, F.H.; Aaraj, N. Survey on fully homomorphic encryption, theory, and applications. *Proc. IEEE* **2022**, *110*, 1572–1609. [\[CrossRef\]](#)

15. Cheon, J.H.; Han, K.; Kim, A.; Kim, M.; Song, Y. A full RNS variant of approximate homomorphic encryption. In *Proceedings of the Selected Areas in Cryptography—SAC 2018: 25th International Conference, Calgary, AB, Canada, 15–17 August, 2018*; Revised Selected Papers 25; Springer: Berlin/Heidelberg, Germany, 2019; pp. 347–368.
16. Lee, E.; Lee, J.; Lee, J.; Kim, Y.; Kim, Y.; No, J.; Choi, W. Low-complexity deep convolutional neural networks on fully homomorphic encryption using multiplexed parallel convolutions. In *Proceedings of the International Conference on Machine Learning, Baltimore, MD, USA, 17–23 July 2022*; PMLR: Cambridge, MA, USA, 2022; pp. 12403–12422.
17. Kim, D.; Guyot, C. Optimized privacy-preserving CNN inference with fully homomorphic encryption. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 2175–2187. [[CrossRef](#)]
18. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016*; IEEE: New York, NY, USA; pp. 770–778.
19. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
20. Gilad-Bachrach, R.; Dowlin, N.; Laine, K.; Lauter, K.; Naehrig, M.; Wernsing, J. CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proceedings of the International Conference on Machine Learning, New York, NY, USA, 20–22 June 2016*; PMLR: Cambridge, MA, USA, 2016; pp. 201–210.
21. Wan, J.; Li, D.; Fang, J.; Jiang, Z.L. LPFHE: Low-Complexity Polynomial CNNs for Secure Inference over FHE. In *Proceedings of the European Symposium on Research in Computer Security*; Springer: Berlin/Heidelberg, Germany, 2024; pp. 403–423.
22. Ao, W.; Boddeti, V.N. AutoFHE: Automated Adaption of CNNs for Efficient Evaluation over FHE. In *Proceedings of the 33rd USENIX Security Symposium (USENIX Security 24), Philadelphia, PA, USA, 14–16 August 2024*; USENIX Association: Berkeley, CA, USA, 2024; pp. 2173–2190.
23. Lee, J.; Lee, E.; Kim, Y.; Lee, Y.; Lee, J.; Kim, Y.; No, J. Optimized Layerwise Approximation for Efficient Private Inference on Fully Homomorphic Encryption. *arXiv* **2023**, arXiv:2310.10349.
24. Cheon, S.; Lee, Y.; Kim, D.; Lee, J.M.; Jung, S.; Kim, T.; Lee, D.; Kim, H. DaCapo: Automatic Bootstrapping Management for Efficient Fully Homomorphic Encryption. In *Proceedings of the 33rd USENIX Security Symposium (USENIX Security 24), Philadelphia, PA, USA, 14–16 August 2024*; USENIX Association: Berkeley, CA, USA, 2024; pp. 6993–7010.
25. Ebel, A.; Garimella, K.; Reagen, B. Orion: A Fully Homomorphic Encryption Framework for Deep Learning. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Rotterdam, The Netherlands, 30 March–3 April 2025*; ACM: San Jose, CA, USA, 2025; Volume 2, pp. 734–749.
26. Liu, Y.; Lai, J.; Li, L.; Sui, T.; Xiao, L.; Yuan, P.; Zhang, X.; Zhu, Q.; Chen, W.; Xue, J. ReSBM: Region-based Scale and Minimal-Level Bootstrapping Management for FHE via Min-Cut. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Rotterdam, The Netherlands, 30 March–3 April 2025*; ACM: San Jose, CA, USA, 2025; Volume 1, pp. 924–939.
27. Bossuat, J.; Mouchet, C.; TroncosoPastoriza, J.; Hubaux, J. Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 587–617.
28. Lee, E.; Lee, J.; No, J.; Kim, Y. Minimax approximation of sign function by composite polynomial for homomorphic comparison. *IEEE Trans. Dependable Secur. Comput.* **2021**, *19*, 3711–3727. [[CrossRef](#)]
29. Paterson, M.S.; Stockmeyer, L.J. On the number of nonscalar multiplications necessary to evaluate polynomials. *SIAM J. Comput.* **1973**, *2*, 60–66. [[CrossRef](#)]
30. Sarkar, S.; Kundu, S.; Beerel, P.A. RLNet: Robust Linearized Networks for Efficient Private Inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, 16–22 June 2024*; IEEE: New York, NY, USA, 2024; pp. 244–253.
31. Xu, X.; Wang, Z.; Ning, R.; Xin, C.; Wu, H. PrivShap: A Finer-granularity Network Linearization Method for Private Inference. *Trans. Mach. Learn. Res.* **2025**, 1–16. Available online: <https://openreview.net/forum?id=7TliYmJr2m> (accessed on 29 April 2026).
32. Molchanov, P.; Mallya, A.; Tyree, S.; Frosio, I.; Kautz, J. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019*; IEEE: New York, NY, USA, 2019; pp. 11264–11272.
33. Otsu, N. A threshold selection method from gray-level histograms. *IEEE Trans. Syst. Man, Cybern.* **1979**, *9*, 62–66. [[CrossRef](#)]
34. Microsoft. Microsoft SEAL Library. GitHub Repository. 2021. Available online: <https://github.com/microsoft/SEAL> (accessed on 1 April 2026).
35. Krizhevsky, A.; Hinton, G. *Learning Multiple Layers of Features from Tiny Images*; Technical report; University of Toronto: Toronto, ON, Canada, 2009.

36. Cheon, J.H.; Hhan, M.; Hong, S.; Son, Y. A Hybrid of Dual and Meet-in-the-Middle Attack on Sparse and Ternary Secret LWE. *IEEE Access* **2019**, *7*, 89497–89506.
37. Lee, J.; Lee, E.; Lee, Y.; Kim, Y.; No, J. High-precision bootstrapping of RNS-CKKS homomorphic encryption using optimal minimax polynomial approximation and inverse sine function. In *Proceedings of the Advances in Cryptology—EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, 17–21 October 2021; Proceedings, Part I* 40; Springer: Berlin/Heidelberg, Germany, 2021; pp. 618–647.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.