AUTOFHE: AUTOMATED ADAPTION OF CNNS FOR EFFICIENT EVALUATION OVER FHE

Anonymous authors
Paper under double-blind review

ABSTRACT

Secure inference of deep convolutional neural networks (CNNs) was recently demonstrated under the RNS-CKKS fully homomorphic encryption (FHE) scheme. The state-of-the-art solution uses a high-order composite polynomial to approximate non-arithmetic ReLUs and refreshes zero-level ciphertext through bootstrapping. However, this solution suffers from prohibitively high latency, both due to the number of levels consumed by the polynomials (47%) and the inference time consumed by bootstrapping operations (70%). Furthermore, it requires a hand-crafted architecture for homomorphically evaluating CNNs by placing a bootstrapping operation after every Conv-BN layer. To accelerate CNNs on FHE and automatically design a homomorphic evaluation architecture, we propose *AutoFHE: Automated adaption of CNNs for evaluation over FHE*. AutoFHE exploits the varying sensitivity of approximate activations across different layers in a network and jointly evolves polynomial activations (EvoReLUs) and searches for placement of bootstrapping operations for evaluation under RNS-CKKS. The salient features of AutoFHE include: i) a multi-objective co-evolutionary (MOCoEv) search algorithm to maximize validation accuracy and minimize the number of bootstrapping operations, ii) a gradient-free search algorithm, R-CCDE, to optimize EvoReLU coefficients, and iii) polynomial-aware training (PAT) to fine-tune polynomial-only CNNs for one epoch to adapt trainable weights to EvoReLUs. We demonstrate the efficacy of AutoFHE through the evaluation of ResNets on CIFAR-10 and CIFAR-100 under RNS-CKKS. Experimental results on CIFAR-10 indicate that in comparison to the state-of-the-art solution, AutoFHE reduces inference time (50 images on 50 threads) by 1,000 seconds and amortized inference time (per image) by 28% and 17% for ResNet-20 and ResNet-32, respectively.

1 INTRODUCTION

Fully homomorphic encryption (FHE) is a promising solution for secure inference of neural networks (Gilad-Bachrach et al., 2016; Brutzkus et al., 2019; Lou & Jiang, 2021; Lee et al., 2022b;a). Homomorphically evaluating CNNs on encrypted data is challenging in two respects: 1) the design of homomorphic evaluation architecture of deep CNNs with arbitrary depth and 2) non-arithmetic operations like ReLU. Recently, FHE-MP-CNN (Lee et al., 2022a) successfully implemented a homomorphic evaluation architecture of ResNets by using bootstrapping (Cheon et al., 2018a; Bossuat et al., 2021) to refresh zero-level ciphertext under the full residue number system (RNS) variant of Cheon-Kim-Kim-Song (RNS-CKKS) scheme (Cheon et al., 2017; 2018b). Since only homomorphic multiplication and addition are supported by FHE, non-arithmetic operations are approximated by polynomials (Gilad-Bachrach et al., 2016; Chou et al., 2018; Brutzkus et al., 2019; Lee et al., 2021a;c; 2022a). For example, FHE-MP-CNN adopts a high-precision Minimax composite polynomial (Lee et al., 2021a;c) with degree $\{15, 15, 27\}$ to approximate ReLUs (AppReLU).

The state-of-the-art approach FHE-MP-CNN is limited by three main design choices. *First*, high-precision approximations like AppReLU only consider function-level approximation and neglect the potential for end-to-end optimization of the entire network response. As such, the same high-precision AppReLU is used to replace all the ReLU layers in the network, which in turn necessitates evaluation of very deep circuits. *Secondly*, due to the high number of levels required for each AppReLU, ciphertexts encrypted with leveled HE schemes like CKKS quickly exhaust their levels. Therefore, a bootstrapping operation is required for each AppReLU to refresh the level of zero-level ciphertexts.

Collectively, while these design choices are very effective at maintaining the performance of the plaintext networks under FHE, they require a prohibitively large number of multiplicative levels and, consequently, numerous bootstrapping operations. *Thirdly*, due to the constraints imposed by the cryptographic scheme (RNS-CKKS in this case), inference of networks in FHE requires co-design of AppReLU and the homomorphic evaluation architecture. These include careful design of AppReLU (number of composite polynomials and their degrees), cryptographic parameters, placement of bootstrapping operations and choice of network architectures to evaluate. On the other hand, high-degree polynomials introduce significant barriers for training/fine-tuning networks towards achieving high accuracy due to vanishing or exploding gradients. (A more comprehensive discussion of related work can be found in Appendix B.)

In this paper, we relax these design choices and accelerate the inference of CNNs over homormorphically encrypted data. The main premise behind our approach is to *directly optimize the end-to-end function represented by the network, instead of optimizing the function represented by the activation function*. This idea allows us to exploit the varying sensitivity of activation function approximation across different layers in a network. Therefore, theoretically, evolving layerwise polynomial approximations of ReLUs (EvoReLU) should reduce the total multiplicative depth required by the resulting polynomial-only networks. To this end, we propose AutoFHE, a search-driven approach to jointly optimize layerwise polynomial approximations of ReLU and the placement of bootstrapping operations. Specifically, we propose a multi-objective co-evolutionary (MOCoEv) algorithm that seeks to *maximize* accuracy while simultaneously *minimizing* the number of bootstrapping operations. AutoFHE jointly searches for the parameters of the approximate activation functions at all layers, i.e., *degrees* and *coefficients* and the optimal placement of the bootstrapping operations in the network. Our contributions are three-fold[1]:

1. AutoFHE automatically searches for EvoReLUs and bootstrapping operations. It provides a diverse set of Pareto-effective solutions that span the trade-off between accuracy and inference time under RNS-CKKS.
2. From an algorithmic perspective,
   (a) We propose a simple yet effective multi-objective co-evolutionary (MOCoEv) algorithm to effectively explore and optimize over the large search space ($10^{79} \sim 10^{230}$) and optimize high-dimensional vectors ($114 \sim 330$) corresponding to our formulation.
   (b) We design a gradient-free algorithm, regularized co-operative co-evolutionary differentiable evolution (R-CCDE), to *optimize* the coefficients of high-degree composite polynomials.
   (c) We introduce polynomial-aware training (PAT) to fine-tune EvoReLU DNNs for *one* epoch.
3. Experimental results on CIFAR-10 indicate that in comparison to FHE-MP-CNN, AutoFHE reduces inference time (50 images on 50 threads) by 1,000 seconds and amortized inference time (per image) by 28% and 17% for ResNet-20 and ResNet-32, respectively. For instance, employing AutoFHE for ResNet-20 results in a model with a Top-1 accuracy of 90.64% with inference time of 2,699 seconds for 50 images, and another model with accuracy of 91.66% and inference time of 4,753 seconds. The amortized inference time for ResNet-20 is 54 seconds per image, and the amortized inference time for ResNet-32 is 95 seconds per image. On CIFAR-100, ResNet-32 with EvoReLUs can reduce inference time by 1,279 seconds and amortized inference time by 22%, and has 68.75% Top-1 accuracy.

2 Preliminaries

RNS-CKKS: The full residue number system (RNS) variant of Cheon-Kim-Kim-Song (RNS-CKKS) (Cheon et al., 2017; 2018b) is a leveled homomorphic encryption (HE) scheme for approximate arithmetic. Under RNS-CKKS, a ciphertext $\boldsymbol{c} \in \mathcal{R}^2_{Q_\ell}$ satisfies the decryption circuit $[\langle \boldsymbol{c}, sk \rangle]_{Q_\ell} = m + e$, where $\langle \cdot, \cdot \rangle$ is the dot product and $[\cdot]_Q$ is the modular reduction function. $\mathcal{R}_{Q_\ell} = \mathbb{Z}_{Q_\ell}[X]/(X^N + 1)$ is the residue cyclotomic polynomial ring. The modulus is $Q_\ell = \prod_{i=0}^{\ell} q_\ell$, where $0 \leq \ell \leq L$. ℓ is a non-negative integer referred to as *level*, and it denotes the capacity of homomorphic multiplications. sk is the secret key with Hamming weight h. m is the original plaintext message and e is a small error that provides security. A ciphertext has $N/2$ slots to accommodate

[1]An anonymized link to our source code will be provided.

$N/2$ complex or real numbers. RNS-CKKS supports homomorphic addition and multiplication:

$$\begin{aligned} \text{Homomorphic Addition: } \mathrm{Decrypt}(\boldsymbol{c} \oplus \boldsymbol{c}') &= \mathrm{Decrypt}(\boldsymbol{c}) + \mathrm{Decrypt}(\boldsymbol{c}') \approx m + m' \\ \text{Homomorphic Multiplication: } \mathrm{Decrypt}(\boldsymbol{c} \otimes \boldsymbol{c}') &= \mathrm{Decrypt}(\boldsymbol{c}) \times \mathrm{Decrypt}(\boldsymbol{c}') \approx m \times m' \end{aligned} \tag{1}$$

Bootstrapping: Leveled HE only allows a finite number of homomorphic multiplications, with each multiplication consuming one level due to rescaling. Once a ciphertext's level reaches zero, a bootstrapping operation is required to refresh it to a higher level and allow more multiplications. The number of levels needed to evaluate a circuit is known as its *depth*. RNS-CKKS with bootstrapping (Cheon et al., 2018a) is an FHE scheme that can evaluate circuits of arbitrary depth. It enables us to homomorphically evaluate deep CNNs on encrypted data. Conceptually, bootstrapping homomorphically evaluates the decryption circuit and raises the modulus from Q_0 to Q_L by using the isomorphism $\mathcal{R}_{q_0} \cong \mathcal{R}_{q_0} \times \mathcal{R}_{q_1} \times \cdots \times \mathcal{R}_{q_L}$ (Bossuat et al., 2021). Practically, bootstrapping (Cheon et al., 2018a) homomorphically evaluates modular reduction $[\cdot]_Q$ by first approximating it by a scaled sine function, which is further approximated through polynomials (Cheon et al., 2018a; Lee et al., 2020). The refreshed ciphertext has level $\ell = L - K$, where K levels are consumed by bootstrapping (Bossuat et al., 2021) for polynomial approximation of modular reduction.

FHE-MP-CNN (Lee et al., 2022a) is the state-of-the-art framework for homomorphically evaluating deep CNNs on encrypted data under RNS-CKKS with high accuracy. Its salient features include, 1) *Compact Packing:* All channels of a tensor are packed into a single ciphertext. Multiplexed parallel (MP) convolution was proposed to process the ciphertext efficiently. 2) *Homomorphic Evaluation Architecture:* Bootstrapping operations are placed after every Conv-BN, except for the first one, to refresh zero-level ciphertexts. This hand-crafted homomorphic evaluation architecture for ResNets is determined by the choice of cryptographic parameters, the level consumption of operations and ResNet's architecture. 3) *AppReLU:* It replaces all ReLUs with the same high-order Minimax composite polynomial Lee et al. (2021a;c) of degrees $\{15, 15, 27\}$. By noting that $\mathrm{ReLU}(x) = x \cdot (0.5 + 0.5 \cdot \mathrm{sgn}(x))$, where $\mathrm{sgn}(x)$ is the sign function, the approximated ReLU (AppReLU) is modeled as $\mathrm{AppReLU}(x) = x \cdot (0.5 + 0.5 \cdot p_\alpha(x)), x \in [-1, 1]$. $p_\alpha(x)$ is the composite Minimax polynomial. The precision α is defined as $|p_\alpha(x) - \mathrm{sgn(x)}| \leq 2^{-\alpha}$. AppReLU is expanded to arbitrary domains $x \in [-B, B]$ encountered by activation functions in CNNs by scaling it as $B \cdot \mathrm{AppReLU}(x/B)$. However, this reduces approximation precision to $B \cdot 2^{-\alpha}$. To estimate the maximum dynamic range B (40 for CIFAR-10 and 65 for CIFAR-100) of ReLUs, FHE-MP-CNN evaluates the pre-trained network on the training dataset. 4) *Cryptographic Parameters:* FHE-MP-CNN sets $N = 2^{16}$, $L = 30$ and Hamming weight $h = 192$. Please refer to Lee et al. (2022a) for the detailed implementation of FHE-MP-CNN and other parameters. These parameters provide 128-bits of security Cheon et al. (2019). 5) *Depth Consumption:* To reduce level consumption, FHE-MP-CNN integrates scaling parameter B into Conv-BN. The multiplicative depth consumption of Bootstrapping (i.e., K), AppReLU, Conv, DownSampling, AvgPool, FC and BN layers are 14, 14, 2, 1, 1, 1, 0, respectively. Statistically, when using FHE-MP-CNN to homomorphically evaluate ResNet-18/32/44/56 on CIFAR-10 or CIFAR-100, AppReLUs consume $\sim 47\%$ of total levels and bootstrapping operations consume $\sim 70\%$ of inference time.

FHE-MP-CNN is limited in terms of neglecting layer wise approximation sensitivity and applying the same high-order polynomial to approximate all ReLUs. Furthermore, the same scaling parameter B is used for all polynomials, resulting in an approximation precision of $B \cdot 2^{-\alpha}$. It focuses on precise function approximation rather than end-to-end objective of networks, e.g., Top-1 accuracy for classification. Considering different distributions of pre-activations across layers allows us to use smaller B' and α' but with the same precision, i.e., $B' \cdot 2^{-\alpha'} = B \cdot 2^{-\alpha}$, for $B' < B$ and $\alpha' < \alpha$. However, layerwise lower-degree polynomials cannot adapt to the homomorphic evaluation architecture designed by FHE-MP-CNN due to differing depth consumption of layerwise polynomials.

3 AutoFHE: Joint EvoReLU and Bootstrapping Search

To minimize the total latency of secure inference dominated by bootstrapping operations induced by high-degree polynomials and automatically design suitable homomorphic evaluation architecture, we propose AutoFHE. It is designed to search for layerwise polynomial approximation of ReLU jointly with placement of bootstrapping. We directly optimize the end-to-end objective to facilitate finding the optimal combination of layerwise polynomials.

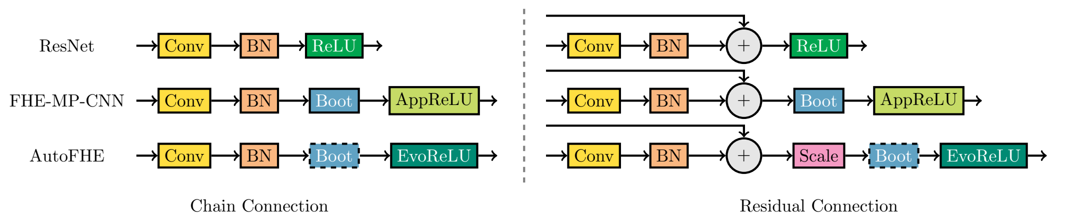

Figure 1: Homomorphic evaluation architectures of the chain connection and the residual connection. Upper: standard ResNet Conv-BN-ReLU triplet He et al. (2016). Middle: FHE-MP-CNN. Bottom: AutoFHE, where dashed rectangles mean the placement of bootstrapping will be searched.

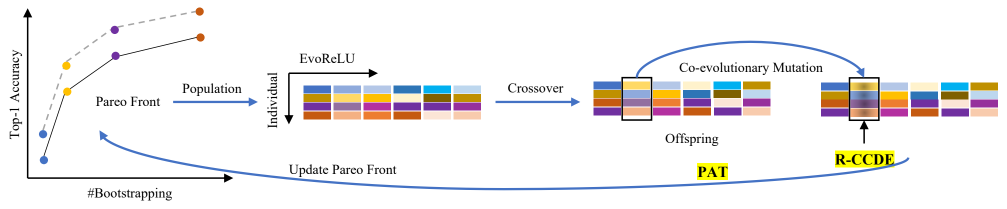

Figure 2: Overview of Multi-objective co-evolutionary (MOCoEv) search algorithm.

3.1 EvoReLU

EvoReLU is defined as $y = \text{EvoReLU}(x) = x \cdot \left(0.5 + p^d(x)\right), x \in [-1, 1], y \in [0, 1]$. The composite polynomial $p^d(x) = (p_K^{d_K} \circ \cdots \circ p_k^{d_k} \circ \cdots \circ p_1^{d_1})(x), 1 \leq k \leq K$ approximates $0.5 \cdot \text{sgn}(x)$. This structure for EvoReLU bears similarity to the Minimax composite polynomial in Lee et al. (2021c; 2022a). However, the objective for optimizing the coefficients is significantly different. We represent the composite polynomial $p^d(x)$ by its degree vector $\boldsymbol{d} = \{d_i\}_{i=1}^{d_K}$, and each sub-polynomial $p_k^{d_k}(x)$ as a linear combination of Chebyshev polynomials of degree d_k, i.e., $p_k^{d_k}(x) = \beta_k \sum_{i=1}^{d_k} \alpha_i \text{T}_i(x)$, where $\text{T}_i(x)$ are the Chebyshev bases of the first kind, α_i are the coefficients for linear combination and β_k is a parameter to scale the output. The coefficients $\boldsymbol{\alpha}_k = \{\alpha_i\}_{i=1}^{d_k}$ control the polynomial's shape, while β_k controls its amplitude. $\boldsymbol{\lambda} = (\boldsymbol{\alpha}_1, \beta_1, \cdots, \boldsymbol{\alpha}_k, \beta_k, \cdots, \boldsymbol{\alpha}_K, \beta_K)$ are the learnable parameters of EvoReLU with the degree $\boldsymbol{d}$.

Homomorphic Evaluation Architecture: The ResNet architecture comprises two types of connections, a chain and a residual connection, as shown in Figure 1. To extend the domain of EvoReLU from $[-1, 1]$ to $[-B, B]$ but avoid extra depth consumption for scaling, we scale the plaintext weight and bias of BatchNorm by $1/B$ in advance for chain connections. But for residual connections, we cannot integrate the scale $1/B$ into BatchNorm's weight and bias. In this case, we scale the ciphertext output of the residual connection by $1/B$ at the expense of one level. Finally, we integrate B into coefficients of $p_K^{d_K}(x)$ to re-scale the output of EvoReLU by B. Given the pre-activation $x \in [-B, B]$, the scaled EvoReLU with the degree $\boldsymbol{d}$ is parameterized by $\boldsymbol{\lambda}$:

$$y = \text{EvoReLU}(x, \boldsymbol{\lambda}; \boldsymbol{d}) = x \cdot (0.5 + p^d(x)), \text{where } x \in [-B, B], y \in [0, B] \tag{2}$$

where we estimate B values for layerwise EvoReLUs on the training dataset. From Figure 1, FHE-MP-CNN places bootstrapping after every Conv-BN, while AutoFHE will search for placement of bootstrapping operations by adapting to different depth consumption of layerwise EvoReLUs.

The Depth Consumption of EvoReLU is $1 + \sum_{k=1}^{K} \lceil \log_2(d_k + 1) \rceil$ when using the Baby-Step Giant-Step (BSGS) algorithm Lee et al. (2020); Bossuat et al. (2021) to evaluate $p^d(x)$.

3.2 MOCoEv: Multi-Objective Co-Evolutionary Search

Search Objectives: Given a neural network function f with L ReLUs and the pre-trained weights $\boldsymbol{\omega}_0$, our goal is to maximize the accuracy of the network while minimizing its inference latency on encrypted data. A possible solution to achieve this goal is to maximize validation accuracy while minimizing the total multiplicative depth of the network with EvoReLUs. This solution does not practically accelerate inference since bootstrapping contributes most to latency, and this solution may not necessarily lead to fewer bootstrapping operations. Therefore, we optimize the parameters of all the EvoReLUs to maximize accuracy and directly seek to minimize the number of bootstrapping layers through a multi-objective optimization problem:

$$\begin{aligned} \min_{\boldsymbol{D}} \quad & \{1 - \mathrm{Acc}_{val}(f(\boldsymbol{\omega^*}); \boldsymbol{\Lambda}^*(\boldsymbol{D}), \boldsymbol{D}), \mathrm{Boot}(\boldsymbol{D})\} \\ \text{s.t.} \quad & \boldsymbol{\Lambda}^* = \mathrm{argmax}_{\boldsymbol{\Lambda}} \{\mathrm{Acc}_{val}(f(\boldsymbol{\omega_0}); \boldsymbol{\Lambda}(\boldsymbol{D}), \boldsymbol{D})\} \\ & \boldsymbol{\omega}^* = \mathrm{argmin}_{\boldsymbol{\omega}} \mathcal{L}_{train}\left(f(\boldsymbol{\omega}); \boldsymbol{\Lambda}^*(\boldsymbol{D}), \boldsymbol{D}\right) \end{aligned} \tag{3}$$

where Acc_{val} is the Top-1 accuracy on a validation dataset val, Boot is the number of bootstrapping operations, $\boldsymbol{D} = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \cdots, \boldsymbol{d}_L\}$ is the degree vector of all EvoReLUs, the corresponding parameters are $\boldsymbol{\Lambda} = \{\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \cdots, \boldsymbol{\lambda}_L\}$, $f(\boldsymbol{\omega}_0)$ is the neural network with the pre-trained weights $\boldsymbol{\omega}_0$, $\mathcal{L}_{train}$ is the training loss. Given a degree vector $\boldsymbol{D}$, the number, and placement of bootstrapping operations can be deterministically determined. Given $\boldsymbol{D}$, we can optimize $\boldsymbol{\Lambda}$ to maximize the validation accuracy. We further fine-tune the network $f(\cdot)$ to minimize the training loss $\mathcal{L}_{train}$. The objectives in equation 3 guide the search algorithm to, i) explore layerwise EvoReLU including its *degrees* and *coefficients*; 2) discover placement of bootstrapping to work well with EvoReLU; 3) trade-off between validation accuracy and inference speed to return a diverse set of Pareto-effective solutions. In this paper, we *propose MOCoEv* to optimize the multi-objective $\min_{\boldsymbol{D}}\{1 - \mathrm{Acc}_{val}(f(\boldsymbol{\omega^*}); \boldsymbol{\Lambda}^*(\boldsymbol{D}), \boldsymbol{D}), \mathrm{Boot}(\boldsymbol{D})\}$. We *propose R-CCDE* and use an evolutionary criterion to maximize $\mathrm{Acc}_{val}(f(\boldsymbol{\omega}_0); \boldsymbol{\Lambda}(\boldsymbol{D}), \boldsymbol{D})$. We *propose PAT* to fine tune approximated networks with EvoReLUs to minimize $\mathcal{L}_{train}\left(f(\boldsymbol{\omega}); \boldsymbol{\Lambda}^*(\boldsymbol{D}), \boldsymbol{D}\right)$.

Search Space: Our search space includes the number of sub-polynomials (K) in our composite polynomial, choice of degrees for each sub-polynomial (d_k) and the coefficients of the polynomials $\boldsymbol{\Lambda}$. Table 1a shows the options for each of these variables. Note that choice $d_k = 0$ corresponds to an identity placeholder, so theoretically, the composite polynomial may have fewer than K sub-polynomials. Furthermore, when the degree of $(p_k^{d_k} \circ p_{k-1}^{d_{k-1}})(x)$ less than or equal to 31 (maximum degree of a polynomial supported on RNS-CKKS Lee et al. (2021a;c)), we merge the two sub-polynomials into a single sub-polynomial $p_k^{d_k}(p_{k-1}^{d_{k-1}})(x)$ with degree $d_k \cdot d_{k-1} \le 31$ before computing its depth. This helps reduce the size of the search space and lead to smoother exploration. Tab. 1b lists the number of ReLUs of our backbone models and the corresponding dimension and size of search space for $\boldsymbol{D}$.

Variable	Option
# polynomials (K)	6
poly degree (d_k)	$\{0, 1, 3, 5, 7\}$
coefficients $(\boldsymbol{\Lambda})$	$\mathbb{R}$

(a) Search variables and options.

Backbone	#ReLUs	Dimension of $\boldsymbol{D}$	Search Space Size
ResNet-20	19	114	10^{79}
ResNet-32	31	186	10^{130}
ResNet-44	43	258	10^{180}
ResNet-56	55	330	10^{230}

(b) AutoFHE search space for ResNets.

MOCoEv: To overcome the challenge of multi-objective search over a high-dimensional $\boldsymbol{D}$ and explore the massive search space, we propose a multi-objective co-evolutionary (MOCoEv) search algorithm. Our approach is inspired by the *divide-and-conquer* strategy of cooperative co-evolution (CC) Yang et al. (2008); Mei et al. (2016); Ma et al. (2018). The key idea of MOCoEv is to decompose the *high*-dimensional multi-objective search problem to multiple *low*-dimensional sub-problems. MOCoEv includes: i) *Decomposition:* given a Pareto-effective solution $\boldsymbol{D} = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \cdots, \boldsymbol{d}_L\}$, MOCoEv improves $\boldsymbol{D}$ by locally mutating $\boldsymbol{d}_\ell, 1 \le \ell \le L$ so that $\boldsymbol{D}' = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \cdots, \boldsymbol{d}'_\ell, \cdots, \boldsymbol{d}_L\}$ dominates $\boldsymbol{D} = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \cdots, \boldsymbol{d}_\ell, \cdots, \boldsymbol{d}_L\}$ in terms of the validation accuracy and the number of bootstrapping; and ii) *Cooperative Evaluation:* we maintain the Pareto front as the context Mei et al. (2016) so we can evaluate the locally mutated solutions cooperatively with each other $\boldsymbol{d}_j, j \neq \ell, 1 \le j \le L$. Figure 2 shows a step of one iteration of MOCoEv. During one iteration, we repeat the step L times until we update all EvoReLUs. We design *crossover* and *co-evolutionary mutation* of MOCoEv to explore and exploit: (1) *Crossover:* given the current Pareto front, we select mating individuals to generate offspring and crossover offspring to exchange *genes* across EvoReLUs. For example, given two mating individuals $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$, we crossover them to obtain $\boldsymbol{D}'_1 = \{\boldsymbol{b}_\ell : \boldsymbol{b}_\ell \in \boldsymbol{D}_1 \cup \boldsymbol{D}_2, 1 \le \ell \le L\}$, $\boldsymbol{D}'_2 = \{\boldsymbol{b}_\ell : \boldsymbol{b}_\ell \in (\boldsymbol{D}_1 \cup \boldsymbol{D}_2)/\boldsymbol{D}'_1, 1 \le \ell \le L\}$; (2) *Co-Evolutionary Mutation:* we mutate the ℓ-th EvoReLU offspring, obtain a new Pareto front from mutated offspring and the current population, and finally update the current population. Then, we move onto the $(\ell+1)$-th EvoReLU and repeat (1)(2) until L EvoReLUs are updated. Therefore, at the end of each iteration, we update the Pareto front L times. We design three types of operators to mutate a composite polynomial function. i) randomly *replace* one polynomial sub-function with a new polynomial. ii) randomly *remove* a sub-function. iii) randomly *insert* a new polynomial. Please refer to Appendix C for background on evolutionary search algorithms. The implementation details of MOCoEv algorithm can be found in Appendix D.1.

3.3 Regularized Cooperative Differentiable Co-evolution

To solve $\mathbf{\Lambda}^* = \text{argmax}_{\mathbf{\Lambda}} \{\text{Acc}_{val}(f(\boldsymbol{\omega_0}); \mathbf{\Lambda}(\boldsymbol{D}), \boldsymbol{D})\}$ in equation 3 where $\boldsymbol{D} = \{\boldsymbol{d}_\ell | 1 \leq \ell \leq L\}$, $\mathbf{\Lambda} = \{\boldsymbol{\lambda}_\ell | 1 \leq \ell \leq L\}$, we propose regularized co-operative co-evolutionary differentiable evolution (R-CCDE). Given degree $\boldsymbol{d}_\ell$, it optimizes $\boldsymbol{\lambda}_\ell$ for function approximation level. However, the function approximation solution $\mathbf{\Lambda}$ maybe not the optimal solution for $\max_{\mathbf{\Lambda}} \{\text{Acc}_{val}(f(\boldsymbol{\omega}); \mathbf{\Lambda}(\boldsymbol{D}), \boldsymbol{D})\}$. So, we use MOCoEv to update the Pareto front in terms of the validation accuracy and the number of bootstrapping. R-CCDE decomposes $\boldsymbol{\lambda}_\ell$ into $\{\boldsymbol{\alpha}_1, \beta_1, \cdots, \boldsymbol{\alpha}_K, \beta_K\}$ corresponding to polynomial sub-functions $y_1 = p_1^{d_1}(x|\boldsymbol{\alpha}_1, \beta_1), y_2 = p_2^{d_2}(y_1|\boldsymbol{\alpha}_2, \beta_2), \cdots, y = p_K^{d_K}(y_{K-1}|\boldsymbol{\alpha}_K, \beta_K)$ by using the forward architecture, $x \mapsto y_1 \mapsto y_2 \cdots \mapsto y_{K-1} \mapsto y$. We adopt gradient-free differentiable evolution (DE) Rauf et al. (2021) to learn $\boldsymbol{\alpha}$ and β. DE uses the difference between individuals for mutation. Given the context vector $\boldsymbol{\lambda}^*$, we optimize $\boldsymbol{\alpha}_k$ and β_k, $1 \leq k \leq K$ alternatively as:

$$\boldsymbol{\alpha}_k^\star = \arg\min_{\boldsymbol{\alpha}_k} \mathcal{L}(\boldsymbol{\alpha}_k|\boldsymbol{\lambda}^*), \ \boldsymbol{\alpha}_k|\boldsymbol{\lambda}^* = (\boldsymbol{\alpha}_1^*, \beta_1^*, \cdots, \boldsymbol{\alpha}_k, \cdots, \boldsymbol{\alpha}_K^*, \beta_K^*) \tag{4}$$

$$\beta_k^\star = \arg\min_{\beta_k} \mathcal{L}(\beta_k|\boldsymbol{\lambda}^*) + \gamma \cdot \beta_k^2, \ \beta_k|\boldsymbol{\lambda}^* = (\boldsymbol{\alpha}_1^*, \beta_1^*, \cdots, \beta_k, \cdots, \boldsymbol{\alpha}_K^*, \beta_K^*) \tag{5}$$

where $\mathcal{L}(\cdot)$ is the ℓ_1 distance between $p^d(x)$ and $0.5 \cdot \text{sgn}(x)$. $\boldsymbol{\alpha}_k^\star$ and $\beta_k^\star$ are then used to update $\boldsymbol{\lambda}^*$. We introduce a *regularization* term for optimizing the scale parameters, where γ is the scaling decay. Scale parameters can prevent polynomials from growing exponentially during the initial iterations. The decay helps guide the parameters gradually toward a value of one and eventually select promising coefficients. A detailed description of our R-CCDE algorithm can be found in Appendix D.2.

3.4 PAT: Polynomial-Aware Training

Replacing ReLU with EvoReLU in pre-trained neural networks injects *small* approximation errors, which leads to performance loss. Fine-tuning can mitigate this performance loss by allowing the learnable weights to adapt to the approximation error. However, backpropagation through EvoReLU leads to exploding gradients due to high-degree polynomials. Thanks to precise *forward* approximation of EvoReLU, we can use gradients from the original non-arithmetic ReLU function for *backpropagation*. Specifically, during *forward* pass, EvoReLU injects slight errors, which are captured by objective functions like cross entropy loss. During the *backward* pass, we bypass EvoReLU and use ReLU to compute gradients to update the weights of the linear trainable layers (e.g., convolution or fully connected). We refer to this procedure, which bears similarity to STE Bengio et al. (2013) and QAT Jacob et al. (2018), as polynomial-aware training (PAT). The following pseudocode illustrates this procedure for a simple example, $\text{EvoReLU}(x) = x(0.5 + (f_3 \circ f_2 \circ f_1)(x))$. In the forward function, we first scale the coefficients of f_3 by B so that the output range of y is $[0, B]$. In backward function, we compute the gradient $\partial y / \partial \text{ReLU(x)}$ instead of $\partial y / \partial \text{EvoReLU(x)}$.

```
def EvoReLU_forward(x, B):
    f3 = f3 * B
    y = f3(f2(f1(x)))
    y = x(0.5 + y)
    return y
```

```
def EvoReLU_backward(x, grad):
    y = ReLU(x)
    grad_y = dy/dx
    grad = grad * grad_y
    return grad
```

4 Experiments

Setup: We benchmark AutoFHE on CIFAR-10 and CIFAR-100 Krizhevsky et al. (2009). Both datasets have 50,000 training and 10,000 validation images at a resolution of 32×32. The validation images are treated as private data and are only used for evaluating the final networks. A randomly selected subset of images (5,120 images for CIFAR-10, and 2048 for CIFAR-100) from the training split are selected as a *minival* Tan & Le (2021) dataset to guide the search process. The Top-1 accuracy on the minival dataset is used for optimizing equation 3. The training split is used by PAT to fine-tune polynomial networks for one epoch. Finally, we report the Top-1 accuracy on the *encrypted validation dataset under RNS-CKKS* as our final result. To evaluate AutoFHE under RNS-CKKS, we adopt the publicly available code of FHE-MP-CNN and adapt it for inference with layerwise EvoReLU. During inference, we simply keep track of the ciphertext levels and call the bootstrapping operation when the level reaches zero, thanks to the optimal placement of bootstrapping operations found by AutoFHE. For fair comparison between AutoFHE and the baseline FHE-MP-CNN, we use the pre-trained network weights provided by FHE-MP-CNN.

Hyperparameters: For MOCoEv, we use a population size of 50 and run it for 20 generations. We set the probability of polynomial replacement to 0.5, probability of polynomial removal to 0.4 and probability of polynomial insertion to 0.1. For R-CCDE, we set the search domain of $\boldsymbol{\alpha}$ to $[-5, 5]$ and that of β to $[1, 5]$. We set the population size for optimizing β to 20. For $\boldsymbol{\alpha}$, we set the population size equal to $10\times$ the number of variables. We set the scaling decay to $\gamma = 0.01$ and the number of iterations to 200. For PAT, we use a batch size of 512 and weight decay of 5×10^{-3}, and clip the gradients to 0.5. We use a learning rate of 8×10^{-5} for CIFAR-10 and 5×10^{-5} for CIFAR-100. On one NVIDIA RTX A6000 GPU, the entire search process for ResNet-20/32/44/56 on CIFAR-10 took 2 days and 17 hours, 6 days and 5 hours, 9 days and 10 hours, 8 days and 19 hours, respectively. The search for ResNet-32 on CIFAR-100 took 5 days and 22 hours.

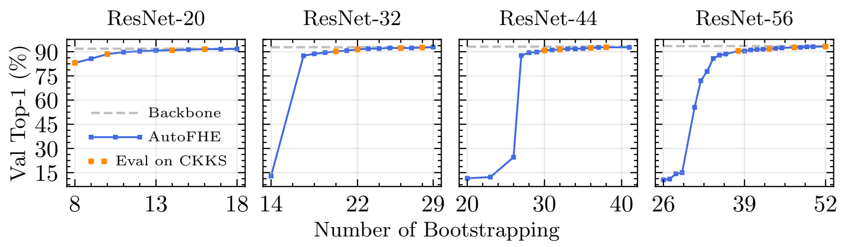

Figure 3: Trade-off between accuracy and number of bootstrapping operations for AutoFHE on plaintext CIFAR-10. Orange points denote solutions that are evaluated on encrypted CIFAR-10 under RNS-CKKS.

4.1 Pareto-Effective Solutions

Figure 3 shows Pareto-effective solutions found by AutoFHE on CIFAR-10 for different ResNet models. The trade-off is between Top-1 validation accuracy on plaintext data and the number of bootstrapping operations required for the corresponding homomorphic evaluation architecture. By optimizing the end-to-end network prediction function, AutoFHE adapts to the differing sensitivity of the activation layers to approximation errors and reduces the number of levels required in comparison to using the same high-degree AppReLU in all the layers. Thus, AutoFHE significantly reduces the number of bootstrapping operations. For example, for ResNet-20, AutoFHE reduces the number of bootstrapping by 22% (compared to FHE-MP-CNN) with the accuracy loss 0.92% (compared to the original network with ReLUs). Lastly, AutoFHE provides a family of solutions offering different trade-offs rather than a single solution, thus providing flexible choice for practical deployments.

4.2 Secure Inference of AutoFHE under RNS-CKKS

Due to the high computation cost of validating networks performance on RNS-CKKS ciphertexts, we select 4 solutions each for ResNet-20/32/44/56 (orange points in Figure 3) for evaluation on a machine with AMD EPYC 7H12 64-Core Processor and 1000 GB RAM. We perform inference on 1000 images, evaluating 50 images at a time using 50 CPU threads. We estimate the inference time for 50 images by averaging over the 20 total evaluation runs. Figure 4 shows the trade-off between Top-1 accuracy over the 1000 images and average inference time for 50 images. We observe that AutoFHE can find Pareto-effective solutions that trade-off accuracy and inference time. The results validate our assumption that directly reducing the number of bootstrapping operations can effectively accelerate inference speed. For shallower networks (ResNet-20 and ResNet-32), AutoFHE reduces

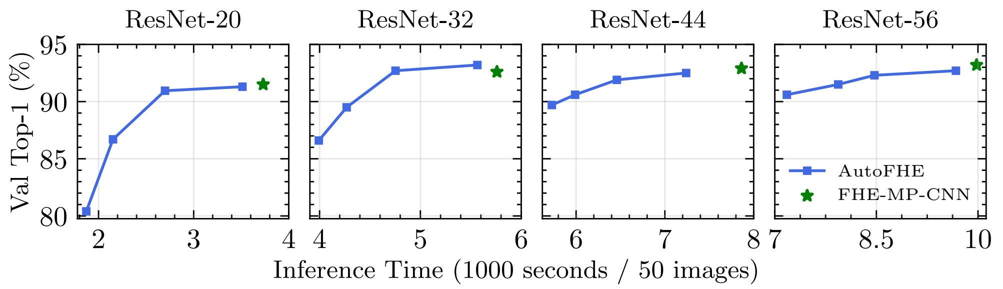

Figure 4: Evaluate AutoFHE on encrypted CIFAR-10 under the RNS-CKKS scheme. We report the inference time of 50 images on 50 CPU threads. See Appendix A for amortized inference time for each image.

Dataset	Backbone Network	Backbone Top-1	FHE-MP-CNN Boot	FHE-MP-CNN Top-1*	FHE-MP-CNN Inference	FHE-MP-CNN Amortized	**AutoFHE** Boot	**AutoFHE** Top-1	**AutoFHE** Inference	**AutoFHE** Amortized	**AutoFHE** #Images
CIFAR-10	ResNet-20	91.86	18	91.31	$3,731 \pm 117$	75 ± 2	14	90.64	$2,699 \pm 59$	54 ± 1	10,000
	ResNet-32	92.80	30	92.40	$5,759 \pm 156$	115 ± 3	26	91.66	$4,753 \pm 159$	95 ± 3	10,000
	ResNet-44	93.02	42	92.65	$7,863 \pm 219$	157 ± 4	38	92.15	$7,239 \pm 114$	145 ± 2	5,000
	ResNet-56	93.49	54	93.07	$9,980 \pm 243$	200 ± 5	52	92.66	$9,674 \pm 209$	193 ± 4	5,000
CIFAR-100	ResNet-32	69.38	30	69.43	$5,846 \pm 127$	117 ± 3	23	67.31	$4,363 \pm 73$	87 ± 1	3,500
							25	68.75	$4,567 \pm 85$	91 ± 2	10,000
							28	68.97	$5,468 \pm 129$	109 ± 3	3,500

Table 2: AutoFHE under the RNS-CKKS scheme. Top-1* accuracy for FHE-MP-CNN, as reported in Lee et al. (2022a). The inference time for 50 images is evaluated on AMD EPYC 7H12 64-core processor using 50 threads.

inference time by $> 1,000$ seconds while preserving performance, and for deeper networks it can still accelerate inference but at the cost of a small drop in accuracy.

We benchmark a select few AutoFHE models on a much larger portion of encrypted CIFAR-10 and CIFAR-100 validation dataset. We estimate inference time for FHE-MP-CNN and AutoFHE on the same hardware platform. We compare AutoFHE models with FHE-MP-CNN in terms of the accuracy and inference time (mean and standard deviation) in Table 2. We also report the number of encrypted validation images used and the corresponding Top-1 accuracy. We observe that, for ResNet-20 and ResNet-32, AutoFHE provides significant acceleration for a small drop in accuracy. Specifically, inference on ResNet-20 and ResNet-32 is lower by **1,032** and **1,006** seconds, respectively. This translates to an amortized inference time reduction of **28**% (75 secs. to **54** secs.) for ResNet-20, and **17**% (115 secs. to **95** secs.) for ResNet-32. In summary, AutoFHE can achieve a Top-1 accuracy of > 90% (90.64%) on encrypted CIFAR-10 under the RNS-CKKS at an amortized inference latency of under one minute (54 secs) per image, which brings us closer towards practically realizing secure inference of deep CNNs under RNS-CKKS. The bottom row in Table 2 shows experimental results on CIFAR-100 for ResNet-32. Here, by reducing the number of bootstrapping operations from 30 to 28, 25 and 23, our inference time reduces by 378, 1279, 1483 seconds, respectively. Especially, the solution with 25 bootstrapping operations reduces inference time by **22**% i.e., **1,279** seconds with accuracy loss of 0.68% compared to FHE-MP-CNN. These results indicate that, i) reducing the number of bootstrapping operations through layerwise mixed-precision EvoReLU is effective in accelerating inference under RNS-CKKS, and 2) AutoFHE can trade off between inference speed and accuracy and provide a portfolio of solutions.

5 Analysis

Depth Distribution of Layer wise EvoReLU: Figure 5 shows the distribution of depth consumption (top) for AutoFHE version of ResNet-56 and the distribution of scaling parameters (bottom), i.e., B. We show three different Pareto-effective solutions corresponding to three solutions in Figure 3 with different number of bootstrapping operations. Chain EvoReLU and residual EvoReLU refer to EvoReLU in the chain connection and the residual connection shown in Figure1. And, since we estimate the dynamic range of the activations on the training dataset, we scale it by $\times 1.5$ as a safety margin. We make two observations; (i) Residual EvoReLUs consistently consume more levels than chain EvoReLUs, suggesting that residual ReLU layers have less tolerance to approximation errors. (ii) Since pre-activations of chain EvoReLUs are normalized, they follow a tighter distribution and need smaller scaling values. However, since the input to the residual ReLU is a superposition of activations from two branches, they are not normalized and thus need much larger scaling values. We plot EvoReLUs of ResNet-56 in Appendix G.

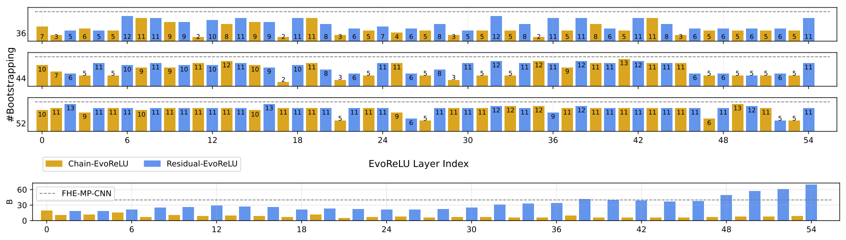

Figure 5: (Top) Distribution of layerwise EvoReLU's depth consumption for ResNet-56 with 36, 44 and 52 bootstrapping operations (distributions for other models can be found in Appendix G). (Bottom) Distribution of scaling parameters (B) for layerwise EvoReLU. Gray dashed lines correspond to AppReLU in FHE-MP-CNN.

Evaluating Co-evolution: To evaluate the effectiveness of *co-evolution* in MOCoEv, we compare MOCoEv with NSGA-II Deb et al. (2002), a standard multi-objective evolution algorithm. Figure 6 (left) shows the trade-off front on for ResNet-20/32 on plaintext CIFAR-10. The results show that co-evolution explores the optimization landscape of high-dimensional variables more effectively. Please refer to Appendix E for more details.

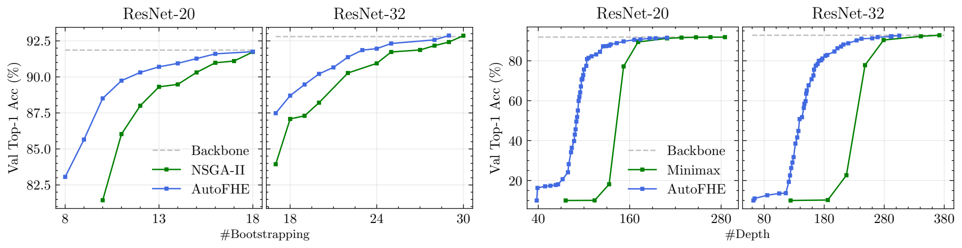

Figure 6: Evaluation of *co-evolution* (left) and *layerwise approximation* (right) on CIFAR-10.

Evaluating Layerwise Approximation: To evaluate the effectiveness of adaptive *layerwise* approximation of AutoFHE, we compare it with uniformly distributed Minimax composite polynomials Lee et al. (2021a;c). Since we cannot design homomorphic evaluation architectures for Minimax polynomials at different precision, we compare Pareto fronts between accuracy and depth consumption (see Appendix F for more details). For fair comparison, we *disable* PAT fine-tuning for AutoFHE. Figure 6 (right) shows that AutoFHE, by virtue of exploiting the varying approximation sensitivity of different layers, has better trade-off than uniformly distributed Minimax polynomials. By minimizing the number of bootstrapping operations, AutoFHE intrinsically reduces overall depth consumption.

6 Conclusion

This paper introduced AutoFHE, an automated approach for accelerating CNNs on FHE and automatically design a homomorphic evaluation architecture. AutoFHE seeks to approximate the end-to-end function represented by the network instead of approximating each activation function. We exploited the varying sensitivity of approximate activations across different layers in a network to jointly evolve composite polynomial activation functions and search for placement of bootstrapping operations for evaluation under RNS-CKKS. Experimental results over multiple ResNet models on CIFAR-10 and CIFAR-100 indicate that AutoFHE can reduce the amortized inference time (per image) by 28% and 17% for ResNet-20 and ResNet-32, respectively, with negligible loss of accuracy. Although our focus in this paper was on ResNets, and consequently ReLU, AutoFHE is a general purpose algorithm that is agnostic to the network architecture or the type of activation function.

REFERENCES

Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 784–796, 2012.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Jean-Philippe Bossuat, Christian Mouchet, Juan Troncoso-Pastoriza, and Jean-Pierre Hubaux. Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 587–617. Springer, 2021.

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

Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *International conference on the theory and application of cryptology and information security*, pp. 409–437. Springer, 2017.

Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. Bootstrapping for approximate homomorphic encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 360–384. Springer, 2018a.

Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. A full rns variant of approximate homomorphic encryption. In *International Conference on Selected Areas in Cryptography*, pp. 347–368. Springer, 2018b.

Jung Hee Cheon, Minki Hhan, Seungwan Hong, and Yongha Son. A hybrid of dual and meet-in-the-middle attack on sparse and ternary secret lwe. *IEEE Access*, 7:89497–89506, 2019.

Edward Chou, Josh Beal, Daniel Levy, Serena Yeung, Albert Haque, and Li Fei-Fei. Faster cryptonets: Leveraging sparsity for real-world encrypted inference. *arXiv preprint arXiv:1811.09953*, 2018.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

Carlos M Fonseca, Luís Paquete, and Manuel López-Ibánez. An improved dimension-sweep algorithm for the hypervolume indicator. In *2006 IEEE international conference on evolutionary computation*, pp. 1157–1163. IEEE, 2006.

Zahra Ghodsi, Nandan Kumar Jha, Brandon Reagen, and Siddharth Garg. Circa: Stochastic relus for private deep learning. *Advances in Neural Information Processing Systems*, 34:2241–2252, 2021.

Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International conference on machine learning*, pp. 201–210. PMLR, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. {GAZELLE}: A low latency framework for secure neural network inference. In *27th USENIX Security Symposium (USENIX Security 18)*, pp. 1651–1669, 2018.

Brian Knott, Shobha Venkataraman, Awni Hannun, Shubho Sengupta, Mark Ibrahim, and Laurens van der Maaten. Crypten: Secure multi-party computation meets machine learning. *Advances in Neural Information Processing Systems*, 34:4961–4973, 2021.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Eunsang Lee, Joon-Woo Lee, Young-Sik Kim, and Jong-Seon No. Minimax approximation of sign function by composite polynomial for homomorphic comparison. *IEEE Transactions on Dependable and Secure Computing*, 2021a.

Eunsang Lee, Joon-Woo Lee, Junghyun Lee, Young-Sik Kim, Yongjune Kim, Jong-Seon No, and Woosuk Choi. Low-complexity deep convolutional neural networks on fully homomorphic encryption using multiplexed parallel convolutions. In *International Conference on Machine Learning*, pp. 12403–12422. PMLR, 2022a.

Joon-Woo Lee, Eunsang Lee, Yongwoo Lee, Young-Sik Kim, and Jong-Seon No. Optimal minimax polynomial approximation of modular reduction for bootstrapping of approximate homomorphic encryption. *IACR Cryptol. ePrint Arch.*, 2020:552, 2020.

Joon-Woo Lee, Eunsang Lee, Yongwoo Lee, Young-Sik Kim, and Jong-Seon No. High-precision bootstrapping of rns-ckks homomorphic encryption using optimal minimax polynomial approximation and inverse sine function. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 618–647. Springer, 2021b.

Joon-Woo Lee, HyungChul Kang, Yongwoo Lee, Woosuk Choi, Jieun Eom, Maxim Deryabin, Eunsang Lee, Junghyun Lee, Donghoon Yoo, Young-Sik Kim, et al. Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. *IEEE Access*, 10:30039–30054, 2022b.

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

Jian Liu, Mika Juuti, Yao Lu, and Nadarajah Asokan. Oblivious neural network predictions via minionn transformations. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pp. 619–631, 2017.

Qian Lou and Lei Jiang. Hemet: A homomorphic-encryption-friendly privacy-preserving mobile neural network architecture. In *International conference on machine learning*, pp. 7102–7110. PMLR, 2021.

Qian Lou, Yilin Shen, Hongxia Jin, and Lei Jiang. Safenet: A secure, accurate and fast neural network inference. In *International Conference on Learning Representations*, 2020.

Xiaoliang Ma, Xiaodong Li, Qingfu Zhang, Ke Tang, Zhengping Liang, Weixin Xie, and Zexuan Zhu. A survey on cooperative co-evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 23(3):421–441, 2018.

Yi Mei, Mohammad Nabi Omidvar, Xiaodong Li, and Xin Yao. A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization. *ACM Transactions on Mathematical Software (TOMS)*, 42(2):1–24, 2016.

Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and Raluca Ada Popa. Delphi: A cryptographic inference service for neural networks. In *USENIX Security Symposium*, 2020.

Jaiyoung Park, Michael Jaemin Kim, Wonkyung Jung, and Jung Ho Ahn. Aespa: Accuracy preserving low-degree polynomial activation for fast private inference. *arXiv preprint arXiv:2201.06699*, 2022.

Deevashwer Rathee, Mayank Rathee, Rahul Kranti Kiran Goli, Divya Gupta, Rahul Sharma, Nishanth Chandran, and Aseem Rastogi. Sirnn: A math library for secure rnn inference. In *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 1003–1020. IEEE, 2021.

Hafiz Tayyab Rauf, Waqas Haider Khan Bangyal, and M Ikramullah Lali. An adaptive hybrid differential evolution algorithm for continuous optimization and classification problems. *Neural Computing and Applications*, 33(17):10841–10867, 2021.

Nidamarthi Srinivas and Kalyanmoy Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3):221–248, 1994.

Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In *International Conference on Machine Learning*, 2021.

Zhenyu Yang, Ke Tang, and Xin Yao. Large scale evolutionary optimization using cooperative coevolution. *Information sciences*, 178(15):2985–2999, 2008.

Andrew Chi-Chih Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pp. 162–167. IEEE, 1986.

APPENDIX

In this appendix, we include the following:

1. Amortized inference trade-offs for secure inference in Section A
2. An expanded discussion of related work for secure inference in Section B.
3. Background and related work for evolutionary algorithms in Section C.
4. Detailed description of the MOCoEv and R-CCDE algorithms in Section D.1 and Section D.2, respectively.
5. Experimental details for evaluating co-evolution in Section E.
6. Experimental details for evaluating layerwise comparison in Section F.
7. EvoReLUs of ResNet-56 on CIFAR-10 in Section G.

A ADDITIONAL RESULTS

Figure 7 shows the trade-offs of AutoFHE evaluated on encrypted CIFAR-10 under the RNS-CKKS scheme. We report the amortized inference time (per image).

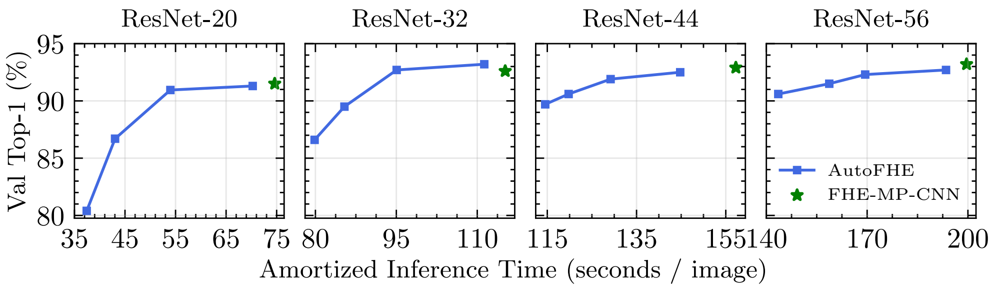

Figure 7: Evaluate AutoFHE on encrypted CIFAR-10 under the RNS-CKKS scheme. We report the amortized inference time (per image).

B RELATED WORK

Secure Inference: Secure inference is a promising solution for resolving the safety and privacy concerns in applications driven by deep learning as a service (DLaaS). Fully homomorphic encryption (FHE) and secure multiparty computation (MPC) are becoming *de-facto* standards of secure inference of deep learning. Secure inference based FHE Gilad-Bachrach et al. (2016); Brutzkus et al. (2019); Lou & Jiang (2021); Lee et al. (2022b;a) better takes advantage of the Cloud service provider's infrastructure. Customers only need to encrypt their private data, sent ciphertexts to the Cloud and decrypt the encrypted result. Secure MPC Liu et al. (2017); Juvekar et al. (2018); Mishra et al. (2020); Lou et al. (2020); Ghodsi et al. (2021); Knott et al. (2021); Rathee et al. (2021) requires regular communication between customers and the Cloud. Regarding the evaluation of non-arithmetic ReLU, FHE cannot directly evaluate ReLU because it only allows arithmetic homomorphic addition and multiplication. However, secure MPC is able to evaluate ReLU using Garbled Circuits (GC) Yao (1986); Bellare et al. (2012) but suffers from high online computation and communication costs Mishra et al. (2020). *Adaption* of CNNs to secure inference by polynomial approximation of non-arithmetic functions is a necessary pre-processing stage. Polynomial approximation enables us to homomorphically evaluate encrypted data on FHE, while it also can greatly reduce online computation and communication costs of secure MPC.

Polynomial Approximation of ReLU: A simple square activation function x^2 is used in CryptoNets Gilad-Bachrach et al. (2016), LoLa Brutzkus et al. (2019) and Deiphi Mishra et al. (2020). Faster CryptoNets Chou et al. (2018) exploits more accurate low-degree approximation $2^{-3}x^2 + 2^{-1}x + 2^{-2}$. SAFENet Lou et al. (2020) adopts $a_1x^3 + a_2x^2 + a_3x + a_4$ or $b_1x^2 + b_2x + b_3$ and uses SGD to train coefficients. When apply SGD to train low-degree polynomial coefficients and network weights simultaneously, polynomials easily lead to gradient exploding problem. On the other hand, low-degree polynomials need train approximated networks from scratch, cannot use pre-trained weights and also has a big accuracy gap comparing with ReLU networks. More recently, AESPA Park et al. (2022) proposes basis-wise normalization to address gradient exploding problem of low-degree polynomial approximated networks. Deiphi and SAFENet apply population-based training (PBT) Jaderberg et al. (2017) to search for placement of polynomials. Because Deiphi and SAFENet are evaluated under secure MPC, they maintain some ReLUs to preserve accuracy. SAFENet also observed layer-wise and channel-wise mixed-precision approximation can better take advantage of varying sensitivity of different layers. Minimax composite polynomials Lee et al. (2021a;c) are especially designed to approximate ReLU under FHE with high precision using composite polynomials. FHE-MP-CNN Lee et al. (2022a) applies the Minimax composite polynomial with degree $\{15, 15, 27\}$ and proves it can maintain performance of pretrained ResNets under the RNS-CKKS FHE scheme. Unlike aforementioned methods learn trainable weights including coefficients of polynomial-only networks by optimizing cross Entropy loss, Minimax is *function-level* approximation by optimizing the polynomial interpolation of ReLU. Given depth, Minimax uses dynamic programming to optimize degrees of composite polynomials and applies improved multi-interval Remez algorithm Lee et al. (2021b) to solve coefficients. So, Minimax can achieve high approximation precision given depth. However, it neglects i) the learning ability of neural networks to adapt to polynomial approximation; 2) the layer-wise varying sensitivity; and 3) the combination of all polynomial activations in a network. In this paper, we take into account both high-precision approximation and network performance. MOCoEv searches for degrees across all layers and directly optimize validation accuracy. We take into account function-level approximation by using R-CCDE to minimize the ℓ_1 distance. We use pre-trained ResNets and propose PAT to fine-tune network trainable weights to adapt to EvoReLUs for just one epoch.

C Evolutionary Search Algorithms

Evolutionary Algorithms (EAs) are a type of search algorithms inspired by Darwin's natural selection. Each candidate solution is a *individual*. NP individuals constitute the *population* with the population size NP. Individuals are assigned *fitness* related to the objective, like validation accuracy for image classification. Based on fitness, we randomly *select* mating individuals. *Crossover* combine mating individuals to generate *offspring*. *Offspring* can be further *mutated* to better exploit current knowledge. Finally, offspring is used to update the current population. We can iteratively repeat this process many times. Each iteration is called *generation*. The number of generations is a simple criterion for stopping EA search.

Multi-Objective EA (MOEA): Given two d-dimensional vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ for a minimization problem, if $\boldsymbol{x}_{1,i} \leq \boldsymbol{x}_{2,i}, \forall i \in \{1, 2, \cdots, d\}$ and $\boldsymbol{x}_{1,j} < \boldsymbol{x}_{2,j}, \exists j \in \{1, 2, \cdots, d\}$, $\boldsymbol{x}_1$ *dominates* $\boldsymbol{x}_2$ Srinivas & Deb (1994). It means $\boldsymbol{x}_1$ is better than $\boldsymbol{x}_2$. It is denoted as $\boldsymbol{x}_1 \prec \boldsymbol{x}_2$. Pareto front or Pareto-effective solutions are referred to those not dominated by others. Deiphi Mishra et al. (2020) and SAFENet Lou et al. (2020) combine two objectives (accuracy and ReLU replacement ratio) to a single objective by weighted sum. It is a widely-used trick to release multi-objective problems to single objective. However, it only obtains a single solution balancing multiple objectives and cannot get Pareto-effective solutions. This is the reason why we apply multi-objective search to obtain Pareto-effective solutions that can be used to different accuracy and latency requirements. EA is naturally well suited for multi-objective search due to population-based optimization. It allows us to obtain the entire set of Pareto-effective solutions in a single run. NSGA-II Deb et al. (2002) is the most well known evolutionary multi-objective algorithm. The proposed MOCoEv adopts *nondominated sorting* and *crowding distance* from NSGA-II. The nondominated sorting can return all Pareto fronts, while crowding distance is used to select the uniformly distributed of individuals within a Pareto front.

Differentiable Evolution (DE) is a gradient-free evolutionary algorithm used to optimize continuous variables Rauf et al. (2021). Given population $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{NP}\}$, where each individual $\boldsymbol{x}_\pi \in \mathbb{R}^d, 1 \leq \pi \leq NP$, the mutation, crossover and selection of DE are defined as:

$$\begin{aligned} &\text{Mutation: } \boldsymbol{v} = \boldsymbol{x}_{\pi_1} + F \cdot (\boldsymbol{x}_{\pi_2} - \boldsymbol{x}_{\pi_3}), 1 \leq \pi_1, \pi_2, \pi_3 \leq NP \\ &\text{Crossover: } \boldsymbol{u}_j = \begin{cases} \boldsymbol{v}_j, & \mathcal{U}(0,1) \leq CR \\ \boldsymbol{x}_{\pi_1, j}, & \text{Otherwise} \end{cases}, \quad 1 \leq j \leq d \\ &\text{Selection: } \boldsymbol{u} = \begin{cases} \boldsymbol{u}, & \mathcal{F}(\boldsymbol{u}) \geq \mathcal{F}(\boldsymbol{x}_{\pi_1}) \\ \boldsymbol{x}_{\pi_1}, & \text{Otherwise} \end{cases} \end{aligned} \tag{6}$$

where F is scaling factor, CR is crossover rate, $\mathcal{F}(\cdot)$ is the fitness evaluation function, and $\mathcal{U}(0, 1)$ is the uniform distribution between 0 and 1.

Cooperative Co-Evolution (CC) algorithms are been proposed to address the challenging to optimize high-dimensional variables Yang et al. (2008); Mei et al. (2016); Ma et al. (2018). Co-evolution decomposes the high-dimensional optimization problem to low-dimensional sub-problems. Then, we can apply EAs or DE to solve sub-problems for discrete or continuous variables, respectively. CC includes two major stages, *decomposition* and *cooperative evaluation*. Decomposition refers to grouping variables. Simple grouping strategies include random grouping or interaction-based (gradient-based) grouping Mei et al. (2016). When the proposed R-CCDE searches for parameters $\boldsymbol{\lambda} = (\boldsymbol{\alpha}_1, \beta_1, \cdots, \boldsymbol{\alpha}_K, \beta_K)$ of $\text{EvoReLU}(x, \boldsymbol{\lambda}|\boldsymbol{d})$, we decompose $\boldsymbol{\lambda}$ to $\boldsymbol{\alpha}_1, \beta_1, \cdots, \boldsymbol{\alpha}_K, \beta_K$ corresponding to polynomial sub-functions $y_1 = p_1^{d_1}(x|\boldsymbol{\alpha}_1, \beta_1), y_2 = p_2^{d_2}(y_1|\boldsymbol{\alpha}_2, \beta_2), \cdots, y = p_K^{d_K}(y_{K-1}|\boldsymbol{\alpha}_K, \beta_K)$. Because the scaling parameter β is used to adjust the amplitude of polynomials, we evolve β followed by $\boldsymbol{\alpha}$. We maintain $2K$ populations for $\boldsymbol{\alpha}_1, \beta_1, \cdots, \boldsymbol{\alpha}_K, \beta_K$, separately. These populations are referred as *sub-populations* or *species* in CC. This decomposition of R-CCDE takes advantage of the forward architecture of composited polynomials, $x \mapsto y_1 \mapsto y_2 \cdots \mapsto y_{K-1} \mapsto y$. When the proposed MOCoEV searches for $\boldsymbol{D} = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \cdots, \boldsymbol{d}_L\}$ to minimize the objective $\{1 - \text{Acc}_{val}(f(\boldsymbol{\omega^*}); \boldsymbol{\Lambda}^*(\boldsymbol{D}), \boldsymbol{D}), \text{Boot}(\boldsymbol{D})\}$, we evolve sub-populations for $\boldsymbol{d}_1, \boldsymbol{d}_2, \cdots, \boldsymbol{d}_L$, separately. It decomposes original problems with dimension $114 \sim 330$ to dimension 6 and greatly reduces the search space size from $10^{79} \sim 10^{230}$ to 10^4. Cooperative evaluation refers to cooperatively evaluate the fitness of an individual of a sub-population. We should take into account other sub-populations when evaluate the individual. R-CCDE is a single objective optimization, so we maintain a *context vector* Mei et al. (2016) $\boldsymbol{\lambda}^* = (\boldsymbol{\alpha}_1^*, \beta_1^*, \cdots, \boldsymbol{\alpha}_K^*, \beta_K^*)$. When evaluate $\boldsymbol{\alpha}_k$ or β_k, we just need to replace the corresponding $\boldsymbol{\alpha}_k^*$ or β^* and assign $\boldsymbol{\alpha}_k$ or β with the fitness of $(\boldsymbol{\alpha}_1^*, \beta_1^*, \cdots \boldsymbol{\alpha}_k \cdots, \boldsymbol{\alpha}_K^*, \beta_K^*)$ or $(\boldsymbol{\alpha}_1^*, \beta_1^*, \cdots \beta_k \cdots, \boldsymbol{\alpha}_K^*, \beta_K^*)$. At the beginning, the context vector is randomly initialized. Finally, R-CCDE outputs the context vector as the searched result. We extend the context vector for multi-objective optimization. MOCoEv maintain context vectors that are the current Pareto Front. Therefore, MOCoEv can effectively improve the Pareto Front using CC.

D The Proposed Search Algorithms

D.1 Multi-Objective Cooperative Evolution

Algorithm 1 shows the details of our proposed MOCoEv search algorithm. MOCoEv takes as input a neural network f with L ReLUs that will be replaced by EvoReLUs, the number of sub-functions a of a composite polynomial K, the population size NP, the number of iterations T, the initial population size N_0. $N_0 \gg N$ because random initialization will generate *invalid* individuals. Invalid individuals refer to those will led to negative levels. The dataset, like CIFAR-10, with training and validation datasets will be used. We will randomly sample a subset from the training dataset as *minival* dataset used for search. The training dataset will be used for fine-tuning. During search and fine-tuning, the validation dataset is strictly unseen. We will report the Top-1 validation accuracy on the validation dataset as final result. MOCoEv will output the Pareto front, namely the population. The population is composed of non-dominated individuals with varying numbers of bootstrapping.

Algorithm 1: MOCoEv

```
input  : The Network $f$ with $L$ ReLUs, the number of sub-functions of a composite polynomial
         $K$, the population size $NP$, the number of iterations $T$, the initial population size
         $N_0 \gg NP$, the replace probability $P_{\text{replace}}$, the remove probability $P_{\text{remove}}$, the insert
         probability $P_{\text{insert}}$, the training dataset Training, the mini-validation dataset Minival;
output : The Pareto front Population;
initial: Population $\{\boldsymbol{D}_1, \boldsymbol{D}_2, \cdots, \boldsymbol{D}_{N_0}\} \leftarrow$ LHS($N_0, L, K$) where $\boldsymbol{D} = \{\boldsymbol{d}_1, \cdots, \boldsymbol{d}_L\}$;
         foreach $\boldsymbol{d}$ in $\boldsymbol{D}$ do $\boldsymbol{\lambda} \leftarrow$ R-CCDE($\boldsymbol{d}$);
         foreach $\boldsymbol{D}$ in Population do Acc $\leftarrow$ Evalaute($f(\boldsymbol{D}, \boldsymbol{\Lambda})$, Minival);
         Population $\leftarrow$ Pareto(Population, Acc, $NP$)
         foreach $\boldsymbol{D}$ in Population do $\boldsymbol{\omega} \leftarrow$ PAT($f(\boldsymbol{D}, \boldsymbol{\Lambda})$, Training);
         foreach $\boldsymbol{D}$ in Population do Acc $\leftarrow$ Evalaute($f(\boldsymbol{\omega}, \boldsymbol{D}, \boldsymbol{\Lambda})$, Minival);
for $t \leftarrow 1$ to $T$ do
    for $i \leftarrow 1$ to $L$ do
        Offspring $\leftarrow$ Select(Population);
        Offspring $\leftarrow$ Crossover(Offspring);
        Offspring $\leftarrow$ Mutate(Offspring $[:, i]$);
        foreach $\boldsymbol{d}$ in Offspring $[:, i]$ do $\boldsymbol{\lambda} \leftarrow$ R-CCDE($\boldsymbol{d}$);
        foreach $\boldsymbol{D}$ in Offspring do $\boldsymbol{\omega} \leftarrow$ PAT($f(\boldsymbol{D}, \boldsymbol{\Lambda})$, Training);
        foreach $\boldsymbol{D}$ in Offspring do Acc $\leftarrow$ Evalaute($f(\boldsymbol{\omega}, \boldsymbol{D}, \boldsymbol{\Lambda})$, Minival);
        Population $\leftarrow$ Pareto(Population +Offspring, Acc, NP);
```

In the initialization, we randomly initialize the population with N_0 individuals, $\{\boldsymbol{D}_1, \boldsymbol{D}_2, \cdots, \boldsymbol{D}_{N_0}\}$ where $\boldsymbol{D}_j = \{\boldsymbol{d}_1, \cdots, \boldsymbol{d}_L\} \in \mathbb{Z}^{L \times K}, 1 \leq j \leq N_0$. $\boldsymbol{d}_i, 1 \leq i \leq L$ is the degrees of a composite polynomial and is randomly sampled using Latin hypercube sampling (LHS) method. The composite polynomials with the layer index i constitute the i-th sub-populations of CC. The proposed R-CCDE searched for coefficients of composite polynomials. The Pareto function is first to use nondominated sorting to find Pareto fronts and then use crowding distance to select individuals given the population size NP. In iteration t, we sequentially evolve EvoReLUs one by one. Given i-th EvoReLU, we first randomly select mating individuals from population based on their accuracy on the minival dataset. We crossover mating individuals to generate offspring. Crossover operates at the network level. Then, we mutate the i-th sub-population. We randomly replace, remove and insert polynomials of the i-th sub-population with the probability P_{replace}, P_{remove} and P_{insert}, respectively. So, we need apply R-CCDE to search coefficients of the i-th sub-population and also use PAT to fine-tune the network. We evaluate the fine-tuned networks on the minival dataset. Finally, we use Pareto function to obtain the new Pareto front given the population size NP. Individuals in the Pareto front will be used to replace population.

D.2 Regularized Co-operative Differential Co-Evolution

Algorithm 2 details the proposed R-CCDE searches for coefficients of a composite polynomial. R-CCDE takes as input a composte polynoimal $p^d(x)$, the target non-arithmetic function $q(x)$, the number of iterations T and scaling decay parameter γ. In this paper, $q(x) = 0.5 \cdot \text{sgn}(x)$. R-CCDE will output the context vector $\boldsymbol{\lambda}^*$ as result. The composite polynomial $p^d(x) = (p_K^{d_K} \circ p_{K-1}^{d_{K-1}} \circ \cdots \circ p_1^{d_1})(x)$ has learnable parameters $\boldsymbol{\lambda} = \{\boldsymbol{\alpha}_1, \beta_1, \cdots, \boldsymbol{\alpha}_K, \beta_K\}$. $\boldsymbol{\alpha}_k = \{\alpha_1, \cdots, \alpha_{d_k}\}$ and $\beta_k, 1 \leq k \leq K$ satisfy $p_k^{d_k}(x) = \beta_k \sum_{i=1}^{d_k} \alpha_i \mathrm{T}_i(x)$. We apply LHS to initialize sub-populations of each $\boldsymbol{\alpha}_k$ and β_k. The population size of $\boldsymbol{\alpha}_k$ sub-populations is equal to $10 \times \lfloor d_k + 1 \rceil / 2$, where β_k is 20. We set $\beta_K = 1$ and not learnable. Given iteration t and sub-population index k, we apply DE to solve $\arg\min_{\boldsymbol{\alpha}_k} \mathcal{L}_{p^d,q}(\boldsymbol{\alpha}_k | \boldsymbol{\lambda}^*)$ where $\mathcal{L}_{p^d,q}$ is the ℓ_1 distance between $p^d(x)$ and $q(x)$. $\boldsymbol{\alpha}_k | \boldsymbol{\lambda}^*$ means to use the current context vector $\boldsymbol{\lambda}^*$ but only search for the corresponding $\boldsymbol{\alpha}_k$. The

Algorithm 2: R-CCDE

```
input  : A composite polynomial $p^d(x) = (p_K^{d_K} \circ p_{K-1}^{d_{K-1}} \circ \cdots \circ p_1^{d_1})(x)$ with parameters
         $\boldsymbol{\lambda} = \{\boldsymbol{\alpha}_1, \beta_1, \cdots, \boldsymbol{\alpha}_K, \beta_K\}$, the target non-arithmetic function $q(x)$, the number of
         iterations $T$, the scaling decay $\gamma$;
output : The context vector $\boldsymbol{\lambda}^*$;
initial : $\boldsymbol{\lambda}^* \leftarrow$ LHS
for $t \leftarrow 1$ to $T$ do
    for $k \leftarrow 1$ to $K$ do
        $\boldsymbol{\alpha}_k^\star = \arg\min_{\boldsymbol{\alpha}_k} \mathcal{L}_{p^d,q}(\boldsymbol{\alpha}_k|\boldsymbol{\lambda}^*),\ \boldsymbol{\alpha}_k|\boldsymbol{\lambda}^* = (\boldsymbol{\alpha}_1^*, \beta_1^*, \cdots \boldsymbol{\alpha}_k, \cdots, \boldsymbol{\alpha}_K^*, \beta_K^*)$;
        $\boldsymbol{\lambda}^* \leftarrow (\boldsymbol{\alpha}_1^*, \beta_1^*, \cdots \boldsymbol{\alpha}_k^\star, \cdots, \boldsymbol{\alpha}_K^*, \beta_K^*)$;
        $\beta_k^\star = \arg\min_{\beta_k} \mathcal{L}_{p^d,q}(\beta_k|\boldsymbol{\lambda}^*) + \gamma \cdot \beta_k^2,\ \beta_k|\boldsymbol{\lambda}^* = (\boldsymbol{\alpha}_1^*, \beta_1^*, \cdots, \beta_k, \cdots, \boldsymbol{\alpha}_K^*, \beta_K^*)$;
        $\boldsymbol{\lambda}^* \leftarrow (\boldsymbol{\alpha}_1^*, \beta_1^*, \cdots \beta_k^\star, \cdots, \boldsymbol{\alpha}_K^*, \beta_K^*)$;
```

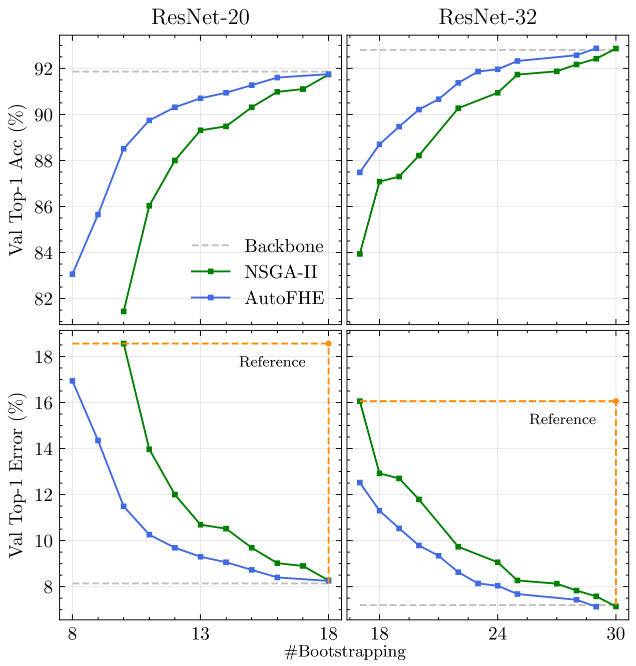

Figure 8: Comparison between AutoFHE with Co-evolution and NSGA-II.

solution $\boldsymbol{\alpha}_k^\star$ evolved by DE is used to update the context vector $\boldsymbol{\lambda}^* = (\boldsymbol{\alpha}_1^*, \beta_1^*, \cdots \boldsymbol{\alpha}_k^\star, \cdots, \boldsymbol{\alpha}_K^*, \beta_K^*)$. Similarly, we also evolve β_k. After obtain $\boldsymbol{\lambda}^*$, we can scale $\boldsymbol{\alpha}_k$ by β_k and obtain the coefficients of the composite polynomial in terms of the first kind of Chebyshev basis.

E Comparison of Search Algorithms

To solve the high-dimensional search problem $\min_{\boldsymbol{D}}\{1 - \mathrm{Acc}_{val}(f(\boldsymbol{\omega}^*); \boldsymbol{\Lambda}^*(\boldsymbol{D}), \boldsymbol{D}), \mathrm{Boot}(\boldsymbol{D})\}$, we propose MoCoEv by using cooperative co-evolution to decompose high-dimensional optimization problem to low-dimensional sub-problems. Because we adopt non-dominated sorting and crowding distance from NSGA-II Deb et al. (2002) to obtain Pareto-effective solutions. NSGA-II is a fair comparison to demonstrate the efficacy of MoCoEv. We conduct the search experiments of ResNet-20 and ResNet-32 on plaintext CIFAR-10. When we use NSGA-II, we set the same hyper-parameters except for increasing the population size by the number of ReLUs. However, we cannot control the number of polynomial being evaluated, the number of evaluation on the minival dataset, and the number of fine-tunning on the training dataset. So, we use the wall-clock search time to make computation comparable, as shown in Table 3.

Backbone		NSGA-II Deb et al. (2002)						**AutoFHE**					
Network	Top1	HV	Time	#Iter	#Poly	#Eval	#Tune	HV	Time	#Iter	#Poly	#Eval	#Tune
ResNet-20	91.86	55.13	2 days 17 hrs	12	219k	9k	8k	72.98	2 days 17 hrs	20	24k	18k	18k
ResNet-32	92.28	74.92	6 days 14 hrs	15	713k	14k	14k	92.89	6 days 5 hrs	20	36k	30k	30k

Table 3: The ablation experiment of search algorithms on plaintext CIFAR-10.

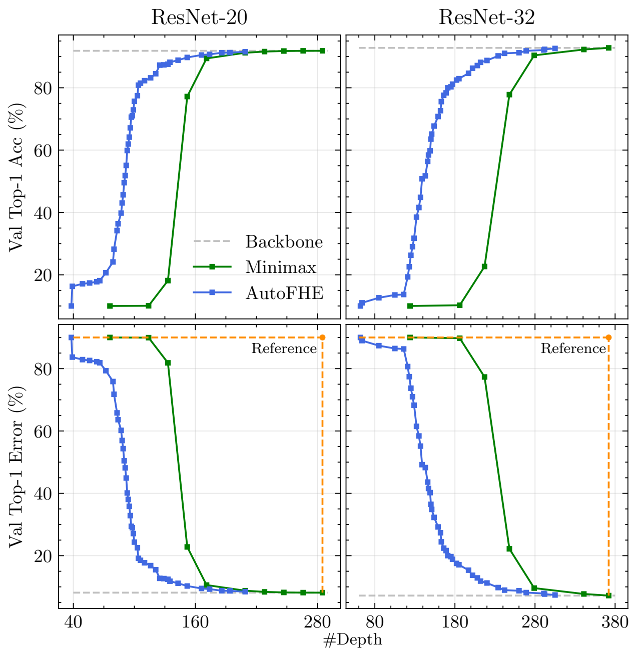

Figure 9: Evaluation of AutoFHE and layerwise Minimax.

The upper row of Figure 8 shows the Pareto fonts of NSGA-II and AutoFHE. In terms of the trade-off of the Top-1 validation accuracy ($> 80\%$) versus the number of bootstrapping operations, NSGA-II is inferior than AutoFHE. It proves the effectiveness of *co-evolution*. Hypervolume (HV) Fonseca et al. (2006) is used to quantitatively compare NSGA-II and AutoFHE. Hypervolume denotes the volume dominated by the Pareto front. The bigger is HV, the better is the Pareto Front. The bottom row of Figure 8 shows the trade-off between the Top-1 validation error and the number of bootstrapping. We compute the HV with respect the reference points, $(18.00, 18.56)$ for ResNet-20 and $(30.00, 16.06)$ for ResNet-32, respectively. Table 3 shows AutoFHE has better HV values than NSGA-II. These ablation experiments prove that co-evolution facilitate the high-dimensional multi-objective search.

F Evaluation of Layerwise ReLU Approximation

To demonstrate the efficacy of *layerwise* approximation of EvoReLUs, we compare AutoFHE with the uniformly approximated networks. We adopt the Minimax composite polynomials Lee et al. (2021a;c) from precision 4 to 14 and use them to replace ReLUs uniformly. However, the Minimax polynomials require re-design of homomorphic evaluation architecture for all composite polynomials with different precision. For example, FHE-MP-CNN uses the polynomial with precision 13 and designs the suitable homomorphic evaluation architecture. To fairly compare the layerwise EvoReLUand the uniformly distributed Minimax polynomial, we use the number of depths consumed by polynomials rather the number of bootstrapping as the criterion. We report the Top-1 validation accuracy on CIFAR-10 as the estimated performance under the RNS-CKKS. Hence, we used the search objective $\min_{\boldsymbol{D}}\{1 - \mathrm{Acc}_{val}(f(\boldsymbol{\omega}^*); \boldsymbol{\Lambda}^*(\boldsymbol{D}), \boldsymbol{D}), \mathrm{Depth}(\boldsymbol{D})$, where $\mathrm{Depth}(\cdot)$ is the total number of depth consumed by polynomials. On the other hand, in this ablation, we do NOT use PAT to fine-tune networks to make the comparison fair.

The upper row of Figure 9 shows Pareto fronts of Minimax and AutoFHE in terms of Top-1 validation accuracy and the number of depth. From the bottom row of Figure 9, we compute the hypervolume

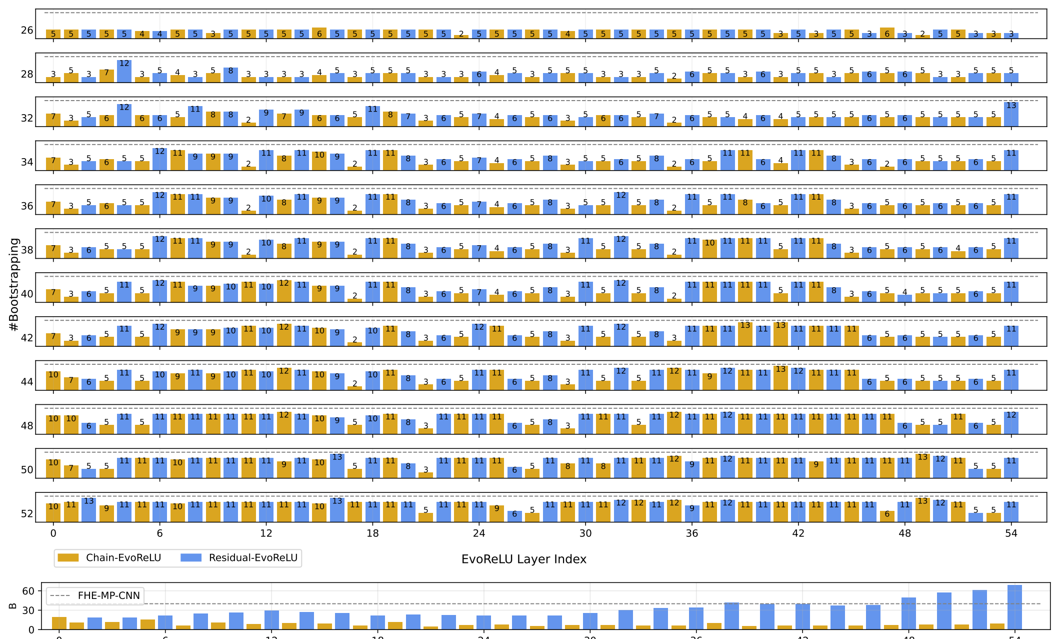

Figure 10: Depth consumption distribution of EvoReLUs of ResNet-56. Upper: depth consumption distributions of layerwise EvoReLUs of different bootstrapping consumption. Bottom: the distribution of scaling parameters (B) of layerwise EvoReLUs. The gray dashed lines show the depth consumption or B of AppReLUs of FHE-MP-CNN.

values with respect to the reference point $(285, 90)$ for ResNet-20 and the reference point $(372, 90)$ for ResNet-32, respectively. AutoFHE has HV 1.58×10^4 better than Minimax HV 1.06×10^4 on ResNet-20, while AutoFHE HV is 1.84×10^4 compared with Minimax HV 1.00×10^4. These experimental results prove: 1) the layerwise approximation is better than uniform approximation; 2) the approximation of AutoFHE is also precise and AutoFHE's performance is not simply because of fine-tuning.

G EvoReLUs of ResNet-56

Figure 10 shows distributions of depth consumption of EvoReLUs of Pareto-effective solutions with varying numbers of bootstrapping operations. From the upper panel of Figure 10, MoCoEv exploits the layerwise varying approximation sensitivity and assigns different depth to each EvoReLU. So, AutoFHE can reduce depth consumption, save the number of bootstrapping operations and further accelerate inference. From the bottom panel of Figure 10, it shows different distributions of pre-activatoins. It proves we can use smaller B values and lower-degree polynomials to have the same precesion, namely $B' \cdot 2^{-\alpha'} = B \cdot 2^{-\alpha}$, where $B' < B$ and $\alpha' < \alpha$. The pre-activations of residual EvoReLU are not normalized so its B values are bigger than chain EvoReLU. So, residual EvoReLUs prefer higher-degree polynomial approximation with more depth consumption to maintain approximation precision.

Figure 11 and Figure 12 show EvoReLUs of different layers of ResNet-56. We include Pareto-effective solutions with the number of bootstrapping operations from 26 to 52. From Figure 11 and Figure 12, high precision solutions consume more depth and approximate ReLUs precisely. Low precision solutions use low-degree polynomials to reduce the depth consumption. From EvoReLU approximation, we learn how AutoFHE can trade-off accuracy and inference speed.

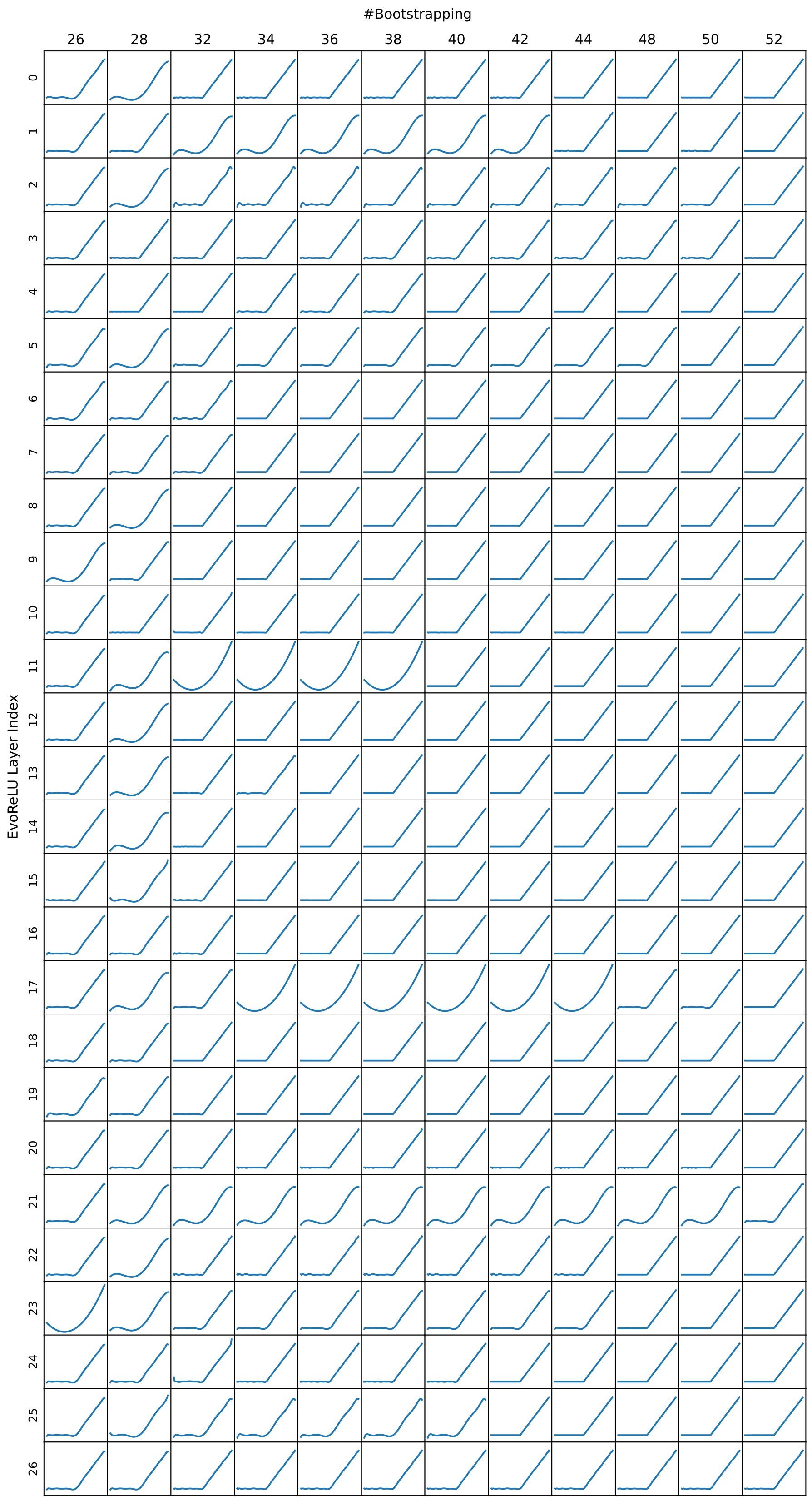

Figure 11: EvoReLUs of ResNet-56 from layer 0 to 26.

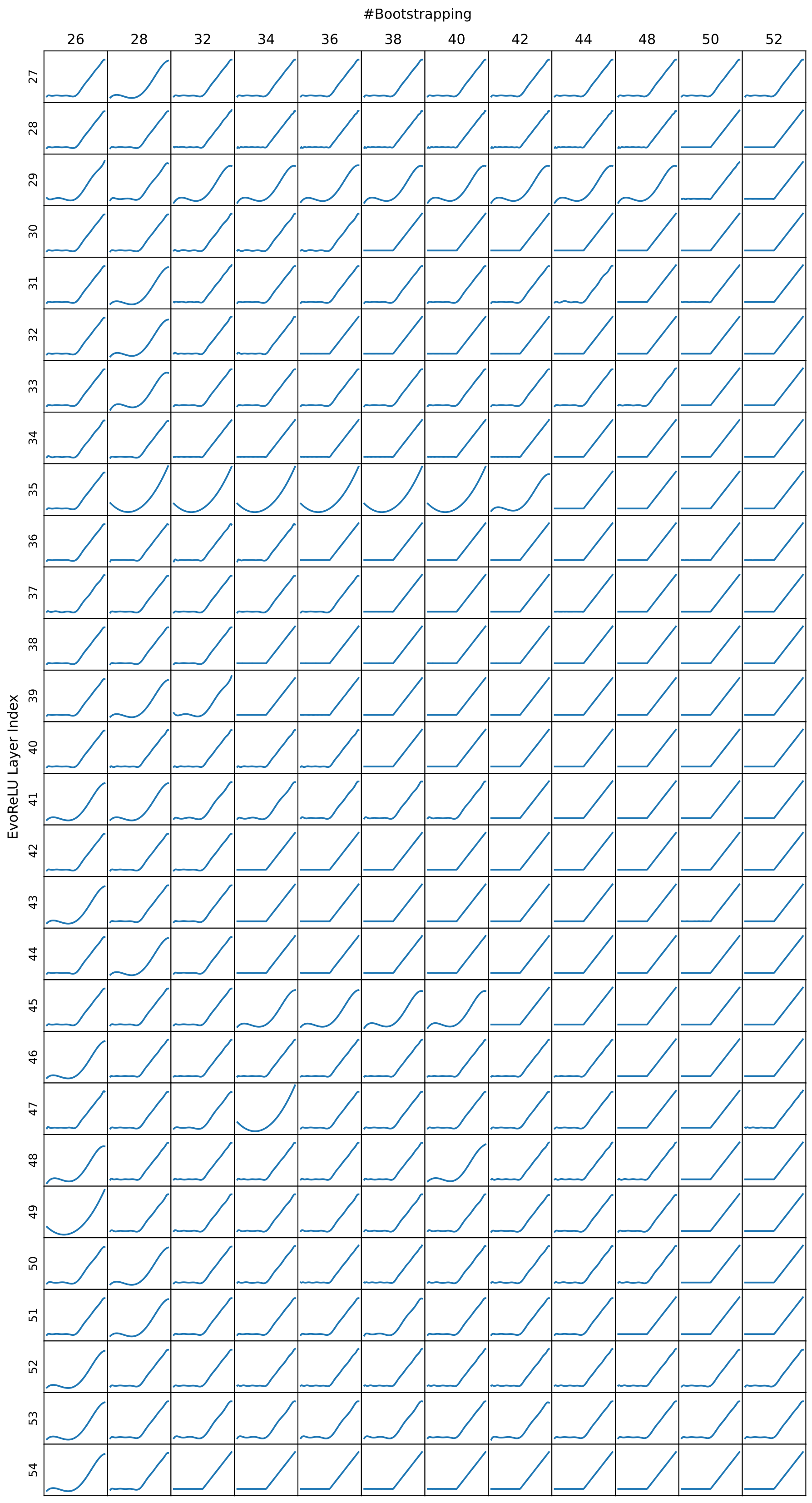

Figure 12: EvoReLUs of ResNet-56 from layer 27 to 54.