AutoReP: Automatic ReLU Replacement for Fast Private Network Inference

+Hongwu Peng 1 +Shaoyi Huang 1 +Tong Zhou 2 Yukui Luo 2 Chenghong Wang 4
Zigeng Wang 1* Jiahui Zhao 1 Xi Xie 1 Ang Li 3 Tony Geng 6 Kaleel Mahmood 1
Wujie Wen 3 Xiaolin Xu 2 Caiwen Ding 1

+These authors contributed equally.

1 University of Connecticut 2 Northeastern University 3 North Carolina State University
4 Duke University 5 Pacific Northwest National Laboratory 6 University of Rochester

{hongwu.peng, shaoyi.huang, zigeng.wang, jiahui.zhao, xi.xie, kaleel.mahmood}@uconn.edu
{zhou.tong1, luo.yuk, x.xu}@northeastern.edu, cw374@duke.edu, ang.li@pnnl.gov
tgeng@ur.rochester.edu, wwen2@ncsu.edu, caiwen.ding@uconn.edu

Abstract

The growth of the Machine-Learning-As-A-Service (MLaaS) market has highlighted clients’ data privacy and security issues. Private inference (PI) techniques using cryptographic primitives offer a solution but often have high computation and communication costs, particularly with non-linear operators like ReLU. Many attempts to reduce ReLU operations exist, but they may need heuristic threshold selection or cause substantial accuracy loss. This work introduces AutoReP, a gradient-based approach to lessen non-linear operators and alleviate these issues. It automates the selection of ReLU and polynomial functions to speed up PI applications and introduces distribution-aware polynomial approximation (DaPa) to maintain model expressivity while accurately approximating ReLUs. Our experimental results demonstrate significant accuracy improvements of 6.12% (94.31%, 12.9K ReLU budget, CIFAR-10), 8.39% (74.92%, 12.9K ReLU budget, CIFAR-100), and 9.45% (63.69%, 55K ReLU budget, Tiny-ImageNet) over current state-of-the-art methods, e.g., SNL. Moreover, AutoReP is applied to EfficientNet-B2 on ImageNet dataset, and achieved 75.55% accuracy with 176.1 \times ReLU budget reduction. The codes are shared on Github1.

1. Introduction

The MLaaS market has seen significant growth in recent years, with many MLaaS platform providers established, e.g., AWS SageMaker [22], Google AI Platform [3], Azure ML [42]. However, most MLaaS solutions require clients to share their private input, compromising data privacy and security. Private inference (PI) techniques, have emerged to preserve data and model confidentiality, providing strong security guarantees. The existing highly-secure PI solutions usually use cryptographic primitives include multi-party computation (MPC) [16, 4, 15] and homomorphic encryption (HE) [23, 5, 11, 26]. Recently, MPC-based PI becomes popular as it supports large-scale networks by partitioning the inference between clients and MLaaS providers.

The main challenge of applying cryptographic primitives in PI comes from the non-linear operators (e.g., ReLU), which introduces ultra-high computation and communication overhead. Fig. 1 (a) shows that ReLU dominates the PI latency, i.e., up to 18.6 \times than the combination of convolutional (Conv) and batch normalization (BN) operations. Reducing these ReLU operators could bring latency reduction, as highlighted in Fig. 1 (b). Centered by this observation, several approaches have been discussed, including replacing ReLUs with linear functions (e.g., SNL [8], DeepRe-
duce [21]) or low degree polynomials (e.g., Delphi [34], SAFE-Net [32]), designing neural architectures with fewer ReLUs (e.g., CryptoNAS [14] and Sphynx [7]), and ultra-low bit representations (TAPAS [39], XONN [38]). However, these techniques (i) require a heuristic threshold selection on ReLU counts, therefore cannot effectively perform design space exploration on ReLU reduction, resulting in sub-optimal solutions, and (ii) result in a significant accuracy drop on large networks and datasets such as ImageNet, hence are not scalable as the number of ReLUs or the number of bits decreases.

We argue that the root cause of the limitations is the disjointedness of non-linear operator reduction and model expressivity in this emerging field. We aim to systematically solve the efficient PI problem by answering two gradually advancing questions: (1) **Which** non-linear operators should be replaced, and (2) **What** to be replaced with to maintain a high model accuracy and expressivity, especially for large DNNs and datasets?

In this work, we introduce a gradient-based automatic ReLU replacement (AutoReP) framework that incorporates joint fine-grained replacement policy (addressing (1)) and polynomial approximation (addressing (2)). Our framework could simultaneously reduce the non-linear operators and maintain high model accuracy and expressivity. In summary, our contributions are as follows:

1. We introduce a **parameterized discrete indicator function**, co-trained with model weights until convergence. Our approach allows for fine-grained selection of ReLU and polynomial functions at the pixel level, resulting in a more optimized and efficient model.

2. We present a **hysteresis loop** update function to enhance the stability of the binarized ReLU replacement training process, which enables a recoverable and stable replacement and leads to better convergence and higher accuracy.

3. Our proposed method, **distribution-aware polynomial approximation (DaPa)**, offers a novel solution to the problem of accurately approximating ReLUs using polynomial functions under specific feature distributions. By minimizing the structural difference between the original and replaced networks and maintaining high model expressivity.

Experimental results show that our AutoReP (ResNet-18) achieves 74.92% accuracy with 12.9K ReLU budget, 8.39% higher than SNL [8], with 1.7x latency reduction, on CIFAR-100. For 73.79% accuracy, AutoReP requires only 6K ReLUs, an 8.2x reduction in ReLU budget vs. SNL [8]. When applied to the larger EfficientNet-B2 on the ImageNet dataset, AutoReP achieved an accuracy of 75.55% with a significant reduction of 176.1 × in ReLU budget.

### 2. Background and Related Work

#### 2.1. Threat Model and Cryptographic Primitives

**2PC setup.** In this paper, we explore a two-party secure computing (2PC) protocol for MLaaS, leveraging prior work [12, 24]. The 2PC protocol lets the client outsource confidential inputs to two servers, who use a 2PC protocol to compute a function securely without revealing intermediate information or results. This approach can scale to enable secure computation for multiple clients with confidential inputs, as demonstrated in [12].

**Threat model.** Here, we focus on an admissible adversary [35] who can compromise one server at a time, which aligns with the non-colluding server assumption in MPC. Our security model assumes semi-honest behavior [36, 9, 20, 46], where the adversary follows the protocol but may perform side calculations to breach security. While not the strongest assumption, this model fits real-world scenarios where trust is established before computation initiation.

**Secret Sharing Basics.** As the most critical operation in multi-party computation, secure sharing bridges the communication between parties while still preserving one’s information secure without the risk of being reasoned by other parties. Specifically, in this work, we adopt the commonly used secret sharing scheme described in CryptTen [27]. An example is given in Fig. 2. As a symbolic representation, \( [x] \leftarrow (x_{S_0}, x_{S_1}) \) denotes the two secret shares, where \( x_{S_i}, i \in \{0, 1\} \), is the share distributed to server \( i \). The share generation and the share recovering adopted in our work are shown below:

- **Share Generation** \( \text{shr}(x) \): A random value \( r \) in \( \mathbb{Z}_m \) is sampled, and shares are generated as \( [x] \leftarrow (r, x - r) \).

- **Share Recovering** \( \text{rec}([x]) \): Given \( [x] \leftarrow (x_{S_0}, x_{S_1}) \), it computes \( x \leftarrow x_{S_0} + x_{S_1} \) to recover \( x \).

As most operators used in DNNs can be implemented through scaling, addition, multiplication, and comparison, here we provide an overview of these basic operations.

**Scaling and Addition.** We denote secret shared matrices as \( [X] \) and \( [Y] \). The encrypted evaluation is given in Eq. 1.
where $a$ is the scaling factor.

\[
[aX + Y] \leftarrow (aX_{S_0} + Y_{S_0}, aX_{S_1} + Y_{S_1})
\]

**Multiplication.** In our work, we consider the use of matrix multiplicative operations in the secret-sharing pattern, specifically $[R] \leftarrow [X] \otimes [Y]$, where $\otimes$ is a general multiplication such as Hadamard product, matrix multiplication, and convolution. To generate the required Beaver triples [2] $[Z] = [A] \otimes [B]$, we utilize an oblivious transfer (OT) [25] based approach, with $A$ and $B$ being randomly initialized. It is important to ensure that the shapes of $[Z]$, $[A]$, and $[B]$ match those of $[R]$, $[X]$, and $[Y]$, respectively, in order to align the matrix computation. Next, each party computes two intermediate matrices, $E_{S_i} = X_{S_i} - A_{S_i}$ and $F_{S_i} = Y_{S_i} - B_{S_i}$, separately. The intermediate shares are then jointly recovered, with $E \leftarrow \text{rec}([E])$ and $F \leftarrow \text{rec}([F])$. Finally, each server $S_i$ calculates the secret-shared $R_{S_i}$ locally to get the result:

\[
R_{S_i} = -i \cdot E \otimes F + X_{S_i} \otimes F + E \otimes Y_{S_i} + Z_{S_i}
\]

**Secure 2PC Comparison.** In the context of secure MPC, the 2PC comparison protocol, also known as the millionaires’ protocol, is designed to determine which of two parties holds a larger value, without revealing the actual value to each other. We use the same protocol as CrypTen [27] to conduct comparison $([X < 0])$ through \(\triangledown\) arithmetic share $[X]$ to binary share $[X]$, conversion, \(\triangledown\) right shift to extract the sign bit $b = (X) >> (L - 1)$ ($L$ is the bit width), and \(\triangledown\) binary share $b$ to arithmetic share $[b]$ conversion for final evaluation result.

**Ciphertext Square Operator.** Considering the element-wise square operator presented in Eq. 3, where $\otimes$ denotes the Hadamard product, it is necessary to generate a Beaver pair, $[Z]$ and $[A]$, such that $[Z] = [A] \otimes [A]$. The pair $[A]$ is randomly initialized and shared among distinct parties through the use of oblivious transfer.

\[
[R] \leftarrow [X] \otimes [X]
\]

Subsequently, the parties compute $[E] = [X] - [A]$ and collaboratively reconstruct $E$ using the recovery operation, $E \leftarrow \text{rec}([E])$. The outcome, $R$, can be derived through the application of Eq. 4.

\[
R_{S_i} = Z_{S_i} + 2E \otimes A_{S_i} + E \otimes E
\]

**2.2. Prior Arts Towards PI Acceleration.**

TAPAS [39] and XONN [38] compressed the model into binary neural network format which has reduced number of bits, and the method can effectively reduce the size of Garbled circuit for MPC comparison protocol implementation. CryptoNAS [14], Sphynx [6] and SafeNet [32] are typical NAS directed works which define searched spaces with reduced count of ReLUs find the suitable architecture. Delphi [34] focuses on partially replacing ReLUs with low order polynomial function to achieve speedup in MPC based PI. SNL [8] and DeepReduce [21] developed frameworks to replace ReLUs with linear function. DeepReduce [21] involves manual design of neural architecture while SNL [8] automates ReLU reduction process. Both prior works follow setting similar to Delphi [34], and exhibit higher 2PC comparison overhead than CrypTen [27] framework which is adopted in our research.

**3. The AutoReP Framework**

As depicted in Fig. 3, we present AutoReP, an automatic replacement approach for accelerating DNN on PI while minimizing the inference accuracy drop. Our approach addresses the challenge of replacing the communication expensive non-linear activation function (i.e., ReLU) with PI-friendly low-order polynomial functions from pre-trained DNNs with less accuracy drop.
We formulate ReLU replacement as a fine-grained feature-level optimization problem. Our solution involves a discrete indicator parameter that determines which ReLU operations should be replaced by polynomial functions to achieve minimal accuracy drop, which will be updated according to a hysteresis function [33]. Our approach improves upon SNL [8] by training a discrete indicator parameter until both indicator and model weight converge, leading to superior convergence accuracy, as opposed to training a continuous slope parameter and fine-tuning the model after fixing the ReLU-polynomial selection in SNL [8].

We propose an approach (DaPa) to determine the suitable polynomial activation functions to replace ReLU, based on the channel-wise feature map distribution. The combination of these techniques enables automatic and efficient acceleration of DNNs for PI for more than 10 times speedup.

3.1. Problem Formulation

Our approach is generalizable to the replacement of ReLU activation functions in any L-layer differentiable neural networks $f^L_W$ parameterized by $W := \{W_i\}_{i=0}^{L-1}$, where the input $X_0 \in \mathbb{R}^{m \times n}$ is mapped to the target $Y \in \mathbb{R}^d$. Our goal is to replace the ReLU (denoted as $g_r$) with the polynomial function (denoted as $g_p$), with the aim of achieving an overall $N$ remaining ReLUs with minimal accuracy drop.

We utilize an indicator parameter $m$ to indicate the replacement position on feature map-level: $m_{ik} = 0$ ($i$th element of $i$th layer), $g_r$ is replaced by $g_p$; $m_{ik} = 1$, use $g_r$. The proposed element-wise discrete indicator parameter $m_{ik}$ gives the expression of the $i$th layer with partially replaced ReLU as $X_{ik} = m_{ik} \odot g_r(Z_{i-1,k}) + (1 - m_{ik}) \odot g_p(Z_{i-1,k})$, where $Z_{i-1}$ is the $i$th layer output. The problem of ReLU replacement can be formulated as follows:

$$\arg\min_W L(f^L_W(X_0), Y) + \mu \cdot \max_{i,j} \left[ \sum_{i=1}^{L-1} \|m_i\|_0 - N, 0 \right] \quad (5)$$

Where $L$ denotes the loss function. However, the second term of Eq. 5 is non-differentiable due to the zero norm regularization applied on discrete indicator parameter $m_i$, making the problem intractable using traditional gradient-based optimization methods. To circumvent the non-differentiable behavior of the discrete indicator parameter, we introduce the utilization of a trainable auxiliary parameter $m_{W_{ik}}$ to parameterize the discrete indicator parameter, as represented by Eq. 6. However, Eq. 6 is still a step function and is non-differentiable. To approximate the gradient of Eq. 6, we adopt straight-through estimator (STE) method which will be discussed in Sec. 3.2.

$$m_{ik} = m_{W_{ik}} > 0 \quad (6)$$

To analyze Eq. 5, we decompose the gradient of auxiliary parameters from Eq. 5 into two parts: accuracy gradient (Eq. 7) from accuracy loss $L_{acc}$ and ReLU count regularization gradient (Eq. 8) from ReLU count penalty $L_N$. Eq. 8 penalizes the auxiliary parameters based on the difference between $g_r(Z)$ and $g_p(Z)$, the term provides recoverability as it allows both gradient directions. Eq. 8 penalizes the ReLU count and ensures the target number of ReLUs is met.

$$\frac{\partial L_{acc}}{\partial m_{W_{ik}}} = \frac{\partial L_{acc}}{\partial X_{ik}} (g_r(Z_{i-1}) - g_p(Z_{i-1})) \frac{\partial m_{ik}}{\partial m_{W_{ik}}} \quad (7)$$

$$\frac{\partial L_N}{\partial m_{W_{ik}}} = \begin{cases} \mu \frac{\partial m_{ik}}{\partial m_{W_{ik}}} , & \|m_i\|_0 - N > 0 \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

3.2. Update Rule of Indicator Parameter

Indicator Parameter Gradient. There are various STE functions for estimating the discrete function’s gradient. While linear STE has been used in previous work [40], recent studies suggest that ReLU-like STE has superior convergence [44]. However, ReLU STE may cause gradient freezing problem [43]. To address this issue, we adopt the softplus function ($f(x) = \log(1 + e^x)$) based STE proposed in [43], which is shown in Fig. 4(a), to estimate the indicator parameter gradient $\frac{\partial L_{acc}}{\partial m_{W_{ik}}}$. We propose the use of a hysteresis indicator parameter update to enhance the stability of the indicator parameter $m_{ik}$ during the forward binarization step with Eq. 9.

$$m_{W_{ik}} + = \eta \frac{\partial L_2}{\partial m_{W_{ik}}} , m_{ik} = m_{W_{ik}} > 0 \quad (9)$$

Figure 4: Hysteresis indicator parameter update.

Stability of Indicator Parameter Update. During each training iteration, the auxiliary parameter and indicator parameter updates are performed in accordance with Eq. 9 with softplus STE. However, there is a potential for indicator parameter instability issues to arise during forward binarization step as the training converges. This instability can occur when some of the $W_{m_{ik}}$ values are close to zero. Any small perturbation of these values during the update process may result in a flip of the indicator $m_{ik}$, thus affecting the training performance.

$$m_{W_{ik}} + = \eta \frac{\partial L_2}{\partial m_{W_{ik}}} , m_{ik} = m_{W_{ik}} > 0 \quad (9)$$

We propose the use of a hysteresis indicator parameter update to enhance the stability of the indicator parameter $m_{ik}$ during the forward binarization process with Eq. 9.

The proposed hysteresis indicator parameter update is depicted in Fig. 4(b). To reduce the possibility of the indicator flip, the hysteresis indicator parameter update utilizes...
the threshold $t_h$ as a hyperparameter and iteratively evaluates the old indicator values $m_{i,k}$ and the updated auxiliary parameter $m_{W,i,k}$ values to determine the new indicator values $m_{i,k}$, as outlined in Eq. 10. The hysteresis indicator parameter $m_{i,k}$ reaches convergence when all auxiliary parameters $m_{W,i,k}$ no longer fluctuate across the adjustable threshold $\pm t_h$.

$$m_{i,k,t+1} = \begin{cases} 1, & m_{i,k,t} = 1 \text{ and } m_{W,i,k} > -t_h \\ 0, & m_{i,k,t} = 1 \text{ and } m_{W,i,k} \leq -t_h \\ 0, & m_{i,k,t} = 0 \text{ and } m_{W,i,k} \leq t_h \\ 1, & m_{i,k,t} = 0 \text{ and } m_{W,i,k} > t_h \end{cases}$$

(10)

Figure 5: Balance the recoverability and stability through tuning hysteresis threshold

**Recoverability and Stability.** The proposed update rule for the indicator parameter improves the recoverability and stability of the automatic ReLU replacement process. During the replacement, the ReLU will be replaced according to the difference between $g_r(Z)$ and $g_p(Z)$. The less difference between $g_r(Z)$ and $g_p(Z)$ on a feature map location, the more likely $g_r(Z)$ will be replaced by $g_p(Z)$. However, important non-linear features $g_r(Z)$ may be replaced due to the ReLU count penalty. Without recoverability, the replacement process will be more likely to be trapped in local optima (will be shown in experiment). The recoverability in this process is achieved through the use of the accuracy loss gradient in Eq. 7 combined with the softplus STE function. This automatic recovery process increases the accuracy of the ReLU replacement process without the need for an additional hard threshold, unlike in previous methods such as [8, 17]. However, unlike weight pruning, the non-linearity replacement process can also be unstable as training converges. To address this issue, we use the hysteresis indicator parameter update to improve the replacement stability.

An example is demonstrated in Fig. 5, the balance between recoverability and stability in the system can be controlled by tuning the hyperparameter $t_h$. A higher threshold results in decreased recoverability but increased stability, whereas a lower threshold results in the opposite effect. The optimal threshold selection, as shown in the example, lies within the range of 0.01 to 0.02. With the combination of learning rate decay and the hysteresis indicator parameter, the replacement process is able to balance exploration and exploitation throughout the training process, thus leading to a higher accuracy.

### 3.3. Polynomial Approximations of ReLU

Low-order polynomial functions, such as first-order and second-order polynomials, can significantly reduce latency and communication volume in DNN PI applications by more than 20 times [34]. However, first-order polynomial function (linear) does not provide non-linearity, so it could lead to significant model expressivity reduction and lower accuracy under a high reduction ratio when used as ReLU replacement. In contrast, the second-order polynomial provides a certain degree of non-linearity and might be a better replacement for ReLU. However, prior works [34, 32, 1, 13] have yet to find an effective method for determining the coefficient of the second-order polynomial functions for ReLU replacement. To improve the performance and training stability of ReLU replacement, we propose a feature map distribution-aware polynomial approximation (DaPa) for our AutoReP framework.

**Distribution-aware approximation.** The intuition behind the distribution-aware approximation is that minimizing the discrepancy between the output before and after ReLU replacement would lead to a smaller decrease in accuracy. This can be achieved by minimizing the minimum square error (MSE) between $g_r(Z)$ and $g_p(Z)$:

$$\min_w \sum_{Z} (g_r(Z) - g_p(Z))^2$$

(11)

Previous approaches [8, 1, 34, 13] adopt a fixed polynomial function for all layers without taking feature map distribution into account, and leads to worse accuracy and training stability. In contrast to those approaches, we propose a distribution-aware approximation method to dynamically adjust the parameters of the polynomial function. Specifically, for a polynomial function of degree $s$ ($s > 0$), denoted as $g_{p,s}(Z, c) = \sum_{i=0}^{s} c_i Z^i$, where $Z$ and $c_i$ are input and polynomial coefficient, the optimization problem in Eq. 11 can be solved by getting the input probability distribution $p(Z)$, and reformulated the problem as follows:

$$\min_c \sum_{c} \int (g_r(Z) - g_{p,s}(Z, c))^2 p(Z) dZ$$

(12)

Eq. 12 can be solved numerically through the Monte Carlo integration and sampling. Here we present an analytical expression of the approximation of the ReLU activation function using second-order polynomial functions. The input $Z$ is assumed to be drawn from a normal distribution with mean $\mu$ and variance $\sigma^2$, i.e., $Z \sim N(\mu, \sigma^2)$. The analytical solution that minimizes Eq. 12 is shown in Eq. 13.
Figure 6: ReLU replacement comparison with $N(0, 2)$ normal distribution. $L(Z)$ donates square error of approximation. (a) Proposed second order approximation: \( g = 0.14Z^2 + 0.5Z + 0.28 \). (b) Identity [8]: \( g = Z \). (c) Second order [1]: \( g = Z^2 + 1 \). (d) Second order [34, 1, 13]: \( g = Z^2 \).

\[
\begin{align*}
c_0 &= \frac{\sqrt{2}\mu^2 e^{-\frac{\mu^2}{2\sigma^2}}}{4\sqrt{\pi}\sigma} + \frac{\sqrt{2}\sigma e^{-\frac{\sigma^2}{2\mu^2}}}{4\sqrt{\pi}\mu}, \\
c_1 &= -\frac{\sqrt{2}\mu e^{-\frac{\mu^2}{2\sigma^2}}}{2\sqrt{\pi}\sigma} - \frac{\text{erfc}(\frac{\sqrt{2\mu}}{2\sigma})}{2} + 1
\end{align*}
\]

The minimum approximation loss is given in Eq. 14.

\[
\begin{align*}
\min_c L &= \frac{\mu^2 \text{erfc}^2 \left( \frac{\sqrt{2\mu}}{2\sigma} \right)}{4} + \frac{\mu^2 \text{erfc} \left( \frac{\sqrt{2\mu}}{2\sigma} \right)}{2} + \\
&\quad -\frac{\sqrt{2}\mu e^{-\frac{\mu^2}{2\sigma^2}} \text{erfc} \left( \frac{\sqrt{2\mu}}{2\sigma} \right)}{2\sqrt{\pi}} - \frac{\sqrt{2}\sigma e^{-\frac{\sigma^2}{2\mu^2}}}{2\sqrt{\pi}} \\
&\quad + \frac{\sigma^2 \text{erfc}^2 \left( \frac{\sqrt{\sigma^2}}{2\sigma} \right)}{4} + \frac{\sigma^2 \text{erfc} \left( \frac{\sqrt{\sigma^2}}{2\sigma} \right)}{2} - \frac{3\sigma^2 e^{-\frac{\sigma^2}{4\pi}}}{4\pi}
\end{align*}
\]

We illustrate the effectiveness of our proposed approximation method with a second-order approximation of ReLU function for inputs drawn from a normal distribution $Z \sim N(0, 2)$. It is important to note that the feature map distribution may vary, and this is just an illustrative example. These results are illustrated in Fig. 6. It can be observed that the proposed second-order approximation results in the lowest error to approximate ReLU function.

**Channel-wise approximation.** For most CNNs, the distribution of intermediate feature map follows a channel-wise manner due to the batch normalization module. Inspired by this fact, we propose a more fine-grained and accurate ReLU replacement method by adopting channel-wise polynomial approximation. Unlike prior works which approximate the ReLU function using identical polynomial function across entire CNNs [8, 34, 1, 1, 13], the proposed channel-wise approximation gives a smaller accuracy loss.

4. Experiments

4.1. Experimental Setup

PI system setup. Our platform comprises two servers equipped with RTX6000, which are connected to a router with a bandwidth of 1 GB/s via a local area network (LAN). To implement secure computation for PI, we utilize the CrypTen [27] framework with the admissible adversary assumption [35] (refer to Sec. 2).

**Architectures and datasets** To enable cross-work comparison with state-of-the-art approaches, we evaluate AutoReP using second-order polynomial replacement on ResNet-18 [18] and WideResNet-22-8 [45] architectures on CIFAR-10/CIFAR-100 [28] and Tiny-ImageNet [10] datasets. To ensure a fair comparison with previous works [21, 8], we replace ReLU layers from the first convolutional layer. For scalability evaluation, we apply AutoReP to EfficientNet-B2 [41] with ReLU activation function on ImageNet [29]. See Table 2 for more dataset information.

**Baselines.** For ResNet-18 and WideResNet-22-8 on CIFAR-10/100 and Tiny-ImageNet datasets, we pre-train the models using SGD with an initial learning rate (LR) of 0.1 and momentum of 0.9 for 400 epochs. The LR is scheduled using a standard cosine annealing LR scheduler. In the case of EfficientNet-B2 trained on ImageNet, we utilize the PyTorch pre-trained weights [37]. As the EfficientNet-B2 model uses SiLU as the default non-linear activation function, we replace the SiLU activation with ReLU and fine-tuned the model using SGD with a LR of 0.01, momentum of 0.9, and cosine annealing LR scheduler. See Table 1 for the accuracy and number of ReLUs for the baseline models.

**AutoReP algorithm settings** We employ the AutoReP algorithm with a second-order polynomial replacement on the aforementioned pre-trained models. For the indicator parameter update, we use the Adam optimizer with a LR of 0.001, while for the model weight, we use the Adam optimizer with a LR of 0.0001. The LR for both parameters are scheduled to decay based on the cosine annealing decay function. We conduct the majority of the replacement experiments using 150 replacement epochs. To balance recoverability and stability, we set the hysteresis threshold to $t_h = 0.003$. The details of hysteresis threshold selection are in the ablation study (Section 4.3). As described in Section 3.3, we capture the channel-wise running mean and
Table 1: Baselines accuracy, ReLU counts, and latency

<table>
<thead>
<tr>
<th>Datasets</th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
<th>Tiny-ImageNet</th>
<th>ImageNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (%)</td>
<td>95.55</td>
<td>96.29</td>
<td>77.8</td>
<td>80.2</td>
</tr>
<tr>
<td>ReLUs (K)</td>
<td>491.52</td>
<td>1359.87</td>
<td>491.52</td>
<td>1359.87</td>
</tr>
<tr>
<td>Latency (s)</td>
<td>1.234</td>
<td>3.219</td>
<td>1.242</td>
<td>3.230</td>
</tr>
</tbody>
</table>

Table 2: Image classification datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Image size</th>
<th>Class</th>
<th>Training Samples per class</th>
<th>Test Samples per class</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10 [28]</td>
<td>32 × 32</td>
<td>10</td>
<td>5000</td>
<td>1000</td>
</tr>
<tr>
<td>CIFAR-100 [28]</td>
<td>32 × 32</td>
<td>100</td>
<td>500</td>
<td>100</td>
</tr>
<tr>
<td>Tiny-ImageNet [10]</td>
<td>64 × 64</td>
<td>200</td>
<td>500</td>
<td>50</td>
</tr>
<tr>
<td>ImageNet [29]</td>
<td>224 × 224</td>
<td>1000</td>
<td>~1282</td>
<td>50</td>
</tr>
</tbody>
</table>

Table 3: Cross-work comparison on CIFAR-100

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CIFAR-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
<td>#ReLUs</td>
</tr>
<tr>
<td></td>
<td>(K)</td>
</tr>
<tr>
<td>CryptoNAS</td>
<td>100.0</td>
</tr>
<tr>
<td>Sphynx</td>
<td>51.2</td>
</tr>
<tr>
<td>Sphynx</td>
<td>25.6</td>
</tr>
<tr>
<td>DeepReduce</td>
<td>49.2</td>
</tr>
<tr>
<td>DeepReduce</td>
<td>12.3</td>
</tr>
<tr>
<td>SNL*</td>
<td>49.9</td>
</tr>
<tr>
<td>AutoReP (Ours)*</td>
<td>12.9</td>
</tr>
<tr>
<td>AutoReP (Ours)*</td>
<td>50</td>
</tr>
<tr>
<td>AutoReP (Ours)*</td>
<td>12.9</td>
</tr>
<tr>
<td>AutoReP (Ours)*</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CIFAR-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
<td>ReLU ≤ 100 K</td>
</tr>
<tr>
<td></td>
<td>#ReLUs</td>
</tr>
<tr>
<td>CryptoNAS</td>
<td>344.0</td>
</tr>
<tr>
<td>Sphynx</td>
<td>230.0</td>
</tr>
<tr>
<td>DeepReduce</td>
<td>229.4</td>
</tr>
<tr>
<td>DeepReduce</td>
<td>197.0</td>
</tr>
<tr>
<td>SNL</td>
<td>180.0</td>
</tr>
<tr>
<td>AutoReP (Ours)*</td>
<td>120.0</td>
</tr>
<tr>
<td>AutoReP (Ours)*</td>
<td>180</td>
</tr>
<tr>
<td>AutoReP (Ours)*</td>
<td>150</td>
</tr>
<tr>
<td>AutoReP (Ours)*</td>
<td>120</td>
</tr>
</tbody>
</table>


4.2. Experimental Results

AutoReP is evaluated and compared with SOTA works [8, 21, 7, 14, 32, 30, 34, 31, 19], and results are presented in Fig. 7. Our proposed framework achieves significantly better results than the other approaches on the CIFAR-10, CIFAR-100, and Tiny-ImageNet datasets.

4.2.1 Pareto Frontier and Cross-work comparison

Table 3 and Table 4 provide detailed information on the accuracy and latency trade-offs for different ReLU budgets on the CIFAR-100 and Tiny-ImageNet, respectively. It is worth noting that previous works [14, 7, 21, 8] use DELPHI [34] as the PI framework, which employs a garbled circuit implementation that is more computationally and communicationally expensive than the CrypTen [27] used in our evaluation. As a result, their reported latencies may be higher than those presented in our study.

On the CIFAR-100 dataset using ResNet-18, AutoReP achieves 74.92% accuracy with 12.9K ReLU budget, which is 8.39% higher than SNL [8], with the same ReLU budget, while reducing the inference latency by 1.7 times. Our experiments demonstrate that AutoReP has less accuracy drop for ReLU replacement compared to SNL [8]. To achieve a similar accuracy level of 73.79%, AutoReP requires only 6K ReLUs, resulting in an 8.2 times reduction in ReLU budget when compared to SNL [8]. For higher ReLU budgets, AutoReP on WideResNet-22-8 achieves an accuracy improvement of 0.7% to 1.2% compared to SNL [8] on 120K to 180K ReLU counts, while also outperforming DeepReduce [21] with fewer ReLU budgets.

Our AutoReP achieves stronger performance than prior SOTA methods on the Tiny-ImageNet dataset. Starting from ResNet-18 with a ReLU budget of 55K, our AutoReP achieves 63.69% accuracy, outperforming SNL [8] with a 59.1K ReLU budget by 9.45%, while reducing the ReLU budget by a factor of 3.6 compared to SNL with a similar accuracy of 63.39% on a ReLU budget of 198.1K. Moreover, AutoReP achieves a 3.39 × latency reduction with a similar 55K ReLU budget, demonstrating that second-order polynomial replacement does not result in higher latency. For higher ReLU budgets, AutoReP starting from WideResNet-22-8 achieves more than 2.5% average accuracy improvement compared to SNL [8] and DeepReduce [21].

4.2.2 AutoReP with Linear Replacement

As discussed in Section 3, AutoReP outperforms previous works in two key aspects: (1) fine-grained replacement policy using discrete indicator parameter and (2) DaPa activation function. To evaluate the contribution of each aspect, we conduct experiments using a first-order polynomial function and compare it with AutoReP using a second-order DaPa function and SNL [8]. The evaluation results on CIFAR-10 and CIFAR-100 datasets are given in Fig. 8. Note that AutoReP and SNL use a similar DNN setting.

Our experiments on CIFAR-10 demonstrate that Au-
Figure 7: AutoRep achieves Pareto frontiers of ReLU counts vs. test accuracy on CIFAR-10, CIFAR-100, and Tiny-ImageNet. AutoRep outperforms the state-of-the-art methods (SNL [8], DeepReDuce [21], Sphynx [7], CryptoNAS [14], SAFENet [32], SNIP [30], DELPHI [34], L1 filter pruning [31] and LFPC [19]) in all range of ReLU counts on all datasets.

Table 4: Cross-work comparison on Tiny-ImageNet

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Tiny-ImageNet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Methods</td>
</tr>
<tr>
<td></td>
<td>Sphynx</td>
</tr>
<tr>
<td></td>
<td>DeepReduce</td>
</tr>
<tr>
<td></td>
<td>SNL+</td>
</tr>
<tr>
<td></td>
<td>AutoRep (Ours)*</td>
</tr>
<tr>
<td></td>
<td>AutoRep (Ours)*</td>
</tr>
</tbody>
</table>

|               | Sphynx        | 204.8       | 53.51         | 4.401        | 0.261     |
|               | DeepReduce    | 196.6       | 57.51         | 4.61         | 0.293     |
|               | SNL+          | 393.2       | 61.65         | 7.77         | 0.157     |
|               | AutoRep (Ours)* | 204.8      | 53.51         | 4.401        | 0.261     |
|               | AutoRep (Ours)* | 290         | 64.74         | 0.723        | 0.223     |
|               | AutoRep (Ours)* | 190         | 64.32         | 0.574        | 0.338     |

|               | Sphynx        | 614.4       | 60.76         | 12.548       | 0.099     |
|               | DeepReduce    | 917.5       | 64.66         | 17.16        | 0.070     |
|               | SNL+          | 848.8       | 64.42         | 10.281       | 0.132     |
|               | AutoRep (Ours)* | 300        | 67.04         | 1.094        | 0.223     |


Figure 8: AutoRep w. Dapa-2nd, AutoRep w. Dapa-1st, and SNL comparisons on (a) CIFAR-10 (b) CIFAR-100 with first-order polynomial replacement is substantial compared to AutoRep with Dapa-2nd polynomial replacement, exhibiting an 11.6% accuracy drop under the extreme case of low ReLU budgets (6K).

The experimental results indicate that our proposed AutoRep outperforms SNL [8] in both first and second-order replacements, providing evidence to support the two main claims of our framework: 1⃝ better replacement policy, which leads to better convergence, and 2⃝ improved model expressivity through the deployment of second-order DaPa polynomial function.

4.2.3 ImageNet Evaluation

To showcase the effectiveness of AutoRep on larger models and datasets, we conduct experiments on EfficientNet-B2 with ImageNet using DaPa-2nd replacement. We run the replacement policy for 20 epochs with the same LR setting as previously. Results are shown in Table 5. Compared to the baseline model’s 78.158% accuracy, AutoRep achieves a 17.6 × reduction in ReLU budget with only 1.592% accu-
accuracy drop for the 500K ReLU case. Moreover, we achieve a 5.8 × speedup for private inference in this case. In the case of an extremely low ReLU budget of 50K, our AutoReP achieves a 176.1 × reduction in ReLU budget and a 7.8% × speedup compared to the baseline model, with only a 2.61% accuracy drop. The results demonstrate that our AutoReP with DaPa-2nd achieves a significant reduction in ReLU budget and inference speedup while preserving good model accuracy, even for relatively larger models and datasets.

Table 5: AutoReP for ImageNet

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Methods</th>
<th>#ReLUs (K)</th>
<th>Test Acc. (%)</th>
<th>Latency (s)</th>
<th>Acc./ReLU</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoReP</td>
<td>500</td>
<td>76.566</td>
<td>1.945</td>
<td>0.153</td>
<td></td>
</tr>
<tr>
<td>AutoReP</td>
<td>400</td>
<td>76.368</td>
<td>1.832</td>
<td>0.191</td>
<td></td>
</tr>
<tr>
<td>AutoReP</td>
<td>300</td>
<td>76.216</td>
<td>1.72</td>
<td>0.254</td>
<td></td>
</tr>
<tr>
<td>AutoReP</td>
<td>200</td>
<td>76.176</td>
<td>1.608</td>
<td>0.381</td>
<td></td>
</tr>
<tr>
<td>AutoReP</td>
<td>100</td>
<td>75.766</td>
<td>1.495</td>
<td>0.758</td>
<td></td>
</tr>
<tr>
<td>AutoReP</td>
<td>50</td>
<td>75.548</td>
<td>1.439</td>
<td>1.511</td>
<td></td>
</tr>
</tbody>
</table>

All start from EfficientNet-B2

4.3. Ablation Study

Threshold sensitivity. To investigate the impact of hyperparameters on the proposed AutoReP, we conduct experiments on ResNet-18 architecture trained on CIFAR-100 dataset while varying the hysteresis threshold \( t_h \). Specifically, we compare the accuracy performance under different \( t_h \) settings while keeping other hyperparameters fixed. The results are presented in Fig. 9. A lower hysteresis threshold may cause the binarized indicator parameter to flip frequently during convergence, leading to a decrease in accuracy. A hysteresis threshold of 0 causes a decrease in accuracy of 0.45% compared to a threshold of 0.003 under 6K ReLU budgets. Our experiments show that \( t_h = 0.003 \) strikes a good balance between the stability of the binarized indicator parameter and recoverable training, resulting in higher overall accuracy under most cases. Therefore, we adopt \( t_h = 0.003 \) for other experiments.

Parameter sensitivity. To substantiate the assertion that the training of the proposed AutoReP is robust to variations in the ReLU count penalty parameter \( \mu \), as stated in the primary problem formulation of the paper, we conduct a parameter sensitivity analysis. The results are depicted in Fig. 10, where \( \mu \) is normalized by multiplying it with the original number of ReLUs. Our findings reveal that, under the same training setup, the overall accuracy exhibits only minor fluctuations across different values of \( \mu \). This implies that careful tuning of the \( \mu \) parameter is not necessary for the AutoReP framework to achieve good performance.

Figure 10: AutoReP for Wide-ResNet-22-8 on Tiny-ImageNet dataset with different penalty \( \mu \).

5. Conclusion

We propose the AutoReP framework, designed to be seamlessly integrated into MPC-based PI systems for MLaaS provider, and compatible with pre-trained CNN models on datasets of varying scales. The framework’s fine-grained ReLU replacement policy and the distribution-aware polynomial approximation (DaPa) activation function enable it to achieve a 74.92% accuracy on the CIFAR-100 dataset with 12.9K ReLU budget, outperforming the SOTA SNL [8] framework by 8.39%. AutoReP achieves 75.55% accuracy when applied to EfficientNet-B2 on ImageNet and achieve a 176.1 × ReLU reduction.

Acknowledgement

This work was in part supported by the NSF CNS-2247891, 2247892, 2247893, CNS-2153690, CNS-2239672, US DOE Office of Science and Office of Advanced Scientific Computing Research under award 66150: “CENATE - Center for Advanced Architecture Evaluation”. The Pacific Northwest National Laboratory is operated by Battelle for the U.S. Department of Energy under Contract DE-AC05-76RL01830. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.
References


