PrivCirNet: Efficient Private Inference via Block Circulant Transformation

Tianshi Xu
Peking University
tianshixu@stu.pku.edu.cn

Lemeng Wu
Meta, Inc.
lmwu@meta

Runsheng Wang
Peking University
r.wang@pku.edu.cn

Meng Li*
Peking University
meng.li@pku.edu.cn

Abstract

Homomorphic encryption (HE)-based deep neural network (DNN) inference protects data and model privacy but suffers from significant computation overhead. We observe transforming the DNN weights into circulant matrices converts general matrix-vector multiplications into HE-friendly 1-dimensional convolutions, drastically reducing the HE computation cost. Hence, in this paper, we propose PrivCirNet, a protocol/network co-optimization framework based on block circulant transformation. At the protocol level, PrivCirNet customizes the HE encoding algorithm that is fully compatible with the block circulant transformation and reduces the computation latency in proportion to the block size. At the network level, we propose a latency-aware formulation to search for the layer-wise block size assignment based on second-order information. PrivCirNet also leverages layer fusion to further reduce the inference cost. We compare PrivCirNet with the state-of-the-art HE-based framework Bolt (IEEE S&P 2024) and HE-friendly pruning method SpENCNN (ICML 2023). For ResNet-18 and Vision Transformer (ViT) on Tiny ImageNet, PrivCirNet reduces latency by $5.0\times$ and $1.3\times$ with iso-accuracy over Bolt, respectively, and improves accuracy by 4.1% and 12% over SpENCNN, respectively. For MobileNetV2 on ImageNet, PrivCirNet achieves $1.7\times$ lower latency and 4.2% better accuracy over Bolt and SpENCNN, respectively. Our code and checkpoints are available on Git Hub.

1 Introduction

The past few years have witnessed the rapid evolution of deep learning (DL) as well as its increasing adoption in sensitive and private applications, including face authentication [1], medical diagnosis [2], code auto-completion [3], etc. Privacy emerges as a major concern and leads to a growing demand for privacy-preserving DL [4, 5, 6, 7]. Homomorphic encryption (HE) is proposed as a promising technology for privacy protection and attracts a lot of attention [8, 9, 10, 7]. By encrypting the data into ciphertexts, HE allows computation over the encrypted data directly and produces encrypted results, without leaking any knowledge of the data itself [8].

To apply HE for private deep neural network (DNN) inference, there are two main approaches, including the end-to-end HE-based schemes [8, 11, 12, 13, 14, 15, 16, 17, 18] and the **hybrid HE/multi-party computation (MPC)-based schemes** [7, 10, 19, 20, 21, 22, 23]. As shown in Figure 1 (a), the hybrid HE/MPC scheme leverages HE and MPC protocols to evaluate the linear

*Corresponding author: meng.li@pku.edu.cn

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

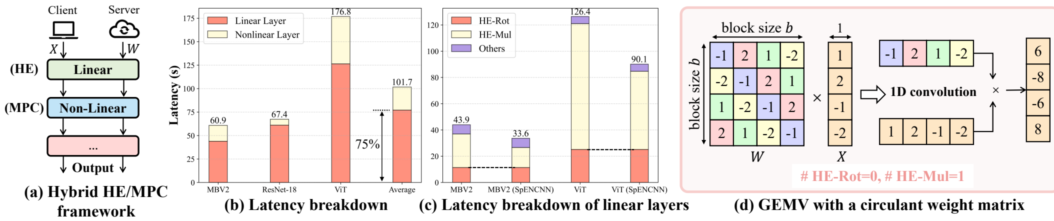

Figure 1: (a) Illustration of Hybrid HE/MPC-based private inference; (b) latency breakdown of linear layers and nonlinear layers based on Bolt's protocol; (c) latency breakdown of linear layers of the original model and SpENCNN with 50% sparsity; (d) GEMV with a circulant weight matrix.

and nonlinear layers separately, which usually demonstrates better accuracy due to its ability to realize accurate activation functions [24]. In contrast, the end-to-end scheme relies on polynomial approximation or TFHE schemes for activation functions, which either suffer from low accuracy or low computation efficiency [25, 11]. Hence, **we focus on the hybrid scheme in our paper.**

While formal privacy protection can be achieved, HE-based DNN inference suffers from high computation cost and orders of magnitude latency overhead [7, 10]. Previous works have proposed algorithm-level optimizations on HE encoding and DNN architectures. HE encoding translates high-dimensional tensor operations of DNNs into 1-dimensional polynomial operations of HE and directly impacts the computation efficiency. For example, Cheetah [10] and Falcon [26] propose efficient encoding algorithms for convolutions while Iron [24] and BubbleBee [22] optimize for general matrix multiplications (GEMMs). Neujeans [27] and Bolt [7] further introduce the baby-step giant-step (BSGS) algorithm to reduce the number of HE rotations and achieve state-of-the-art (SOTA) performance. While significant speedup has been achieved, the overall latency of MobileNetV2 [28] and Vision Transformer (ViT) [29] still exceeds 60s and 170s with Bolt, respectively, as shown in Figure 1 (b) and (c). Meanwhile, linear layers account for more than 75% of total latency due to HE multiplications and rotations, thus, becoming the main optimization target of PrivCirNet.

DNN model optimizations focus on developing HE-friendly architectures. [30, 31, 32, 33, 34, 35, 36] optimize the activation functions for communication and computation reduction, which is orthogonal to our work. [37, 38, 39] propose HE-friendly structured pruning to reduce both HE rotations and multiplications. However, as shown in Figure 1 (c), as these methods are not fully compatible with the SOTA protocols, their latency reduction remains limited, especially for HE rotations[1].

To further reduce the computation cost of linear layers and bridge the latency gap, in this paper, we propose PrivCirNet. *Our key observation is that the circulant transformation of weight matrices enables to convert a general matrix-vector multiplication (GEMV) into a HE-friendly 1-dimensional convolution, simultaneously reducing the HE multiplications and rotations,* as shown in Figure 1 (d). While directly transforming the whole weight matrix into a circulant matrix incurs high accuracy degradation, we propose block circulant transformation and answer the following two questions. First, existing HE encoding algorithms are not fully compatible with block circulant weight matrices, limiting the efficiency gain. How to co-design the encoding algorithm to fully unleash the potential is the first question. Meanwhile, as block circulant transformation introduces structure constraints to weight matrices and inevitably impacts the accuracy, how to determine the layer-wise block sizes for better accuracy/efficiency trade-off becomes the second question.

PrivCirNet features a novel encoding algorithm optimized for block circulant weight matrices, dubbed CirEncode, that reduces the HE computation in proportion to $block\ size$. PrivCirNet also co-design a latency-aware optimization formulation for layer-wise block size assignment based on second-order information. PrivCirNet further leverages layer fusion to reduce the inference cost. With extensive experiments across different DNN architectures (i.e., MobileNetV2, ResNet-18 and ViT) and datasets (i.e, CIFAR, Tiny ImageNet, and ImageNet), we demonstrate PrivCirNet reduces the latency of MobileNetV2, ResNet-18, and ViT by $1.7\times$, $5.0\times$ and $1.3\times$ compared with Bolt [7], respectively. Compared with SOTA HE-friendly pruning method SpENCNN [37], PrivCirNet achieves 4.2%, 4.1%, and 12% better accuracy on MobileNetV2, ResNet-18, and ViT, respectively, demonstrating great capability to accelerate private inference for both ConvNets and Transformers.

[1]The incompatibility is due to the BSGS algorithm and is explained in Appendix D in detail.

2 Preliminaries

Notations. We represent matrices with upper-case letters (e.g., X) and vectors with lower-case letters (e.g., x). We also use lower-case letters with a "hat" symbol (e.g., $\hat{x}$) to represent a polynomial, and $\hat{x}[i]$ to denote the i-th coefficient of $\hat{x}$. We use $\times$ to represent polynomial multiplication and $\odot$ to denote element-wise multiplication. Let $\lceil\cdot\rceil$ denote ceiling operations and $[n]$ denote the set $\{0,\ldots,n-1\}$ for $n \in \mathbb{Z}^+$, where $\mathbb{Z}$ denotes the integer domain. We also denote the set of integer polynomials with $\mathbb{A}_n = \mathbb{Z}[X]/(X^n - 1)$, whose degree n is a power-of-two integer (e.g., 2^{13} following Bolt [7]). We use (d_1, d_2, d_3) to denote the input, hidden, and output dimensions of a GEMM, respectively. For convolution, we use (H, W, C) to represent the input height, width, and number of input channels, and (R, K) to denote the kernel size and number of output channels.

2.1 Cryptographic Primitives

BFV HE Scheme. Following most hybrid HE/MPC schemes [7, 8, 9, 10, 20], PrivCirNet leverages the lattice-based Brakerski-Fan-Vercauteren (BFV) HE scheme [40] and mainly involves the following HE operations, including ciphertext addition (denoted as HE-Add), ciphertext-plaintext multiplication (denoted as HE-Pmult), and ciphertext rotation (denoted as HE-Rot). While HE-Pmult and HE-Rot dominate the overall computation cost, each HE-Rot operation is usually an order of magnitude slower than HE-Pmult [37, 41].

HE Encoding Methods. HE operates over polynomials with 1-dimensional coefficient vectors while DNNs compute over tensors. Encoding is the procedure to map a tensor to a polynomial and directly determines the computation efficiency. Existing encoding methods can be classified into two categories: coefficient encoding [10, 24, 26, 22] and single instruction multiple data (SIMD) encoding [9, 6, 42, 27, 7]. Coefficient encoding can support convolutions efficiently with a single HE-Pmult [10]. In contrast, SIMD encoding only supports element-wise multiplications and requires multiple HE-Rot for convolutions [9]. For GEMMs, either coefficient encoding [22] or SIMD encoding [7] requires HE-Pmult and HE-Rot, while the SIMD encoding algorithm Bolt [7] achieves the SOTA computation efficiency.

The two encoding methods can be transformed to each other through the discrete Fourier transform (DFT) as shown in Lemma 1 [27]. The main reason is that polynomial multiplication implements convolutions in the coefficient domain and is equivalent to element-wise multiplications in the frequency domain, leading to Lemma 1 [27]. While [27] only leverages such nested encoding for convolutions, we show how such schemes can be improved to support block circulant GEMMs and convolutions. We refer interested readers to [27] for a more detailed description.

Lemma 1. $\langle \mathrm{DFT}(w)\rangle_{\mathrm{SIMD}} \times \langle \mathrm{DFT}(x)\rangle_{\mathrm{SIMD}} = \langle \mathrm{DFT}(w) \odot \mathrm{DFT}(x)\rangle_{\mathrm{SIMD}} = \mathrm{DFT}(\langle w\rangle_{\mathrm{Coeff}} \times \langle x\rangle_{\mathrm{Coeff}})$

2.2 Threat Model and Security Guarantee

PrivCirNet works in a general private inference scenario that involves two parties, i.e., server and client. A server holds the proprietary DNN model and a client owns private data [10, 24]. PrivCirNet enables the client to obtain the inference results while keeping the server's model weights and the client's data private. Consistent with previous works [7, 9, 10, 24], we assume the DNN architecture (including the block sizes) is known to both sides and adopt an *honest-but-curious* security model in which both parties follow the specification of the protocol but also try to learn more from than allowed. Following [7, 10], PrivCirNet is built upon cryptographic primitives, including BFV and MPC protocols, and focuses on co-optimizing the DNN architecture and the HE encoding algorithm. The security can hence be guaranteed following [40, 43].

2.3 Related Works

To improve the efficiency of HE-based DNN inference, existing works mainly focus on optimizing the HE encoding algorithm [10, 24, 26, 9, 6, 42, 27, 7] and the DNN architectures [31, 30, 32, 33, 34, 35, 36, 38, 39, 37, 25]. In Table 1, we compare PrivCirNet with prior-art works qualitatively. As can be observed, PrivCirNet features network and encoding co-optimization to improve the efficiency of both GEMMs and convolutions.

Table 1: Comparison with existing private inference works.

Method	HE Encoding Optimization			Target Ops	Network Optimization
	Encoding	# HE-Rot Reduction	# HE-Pmult Reduction		
[31, 30, 35, 33]	✗	✗	✗	ReLU/GELU	ReLU/GELU Pruning
Cheetah [10]	Sparse	✓	✗	GEMV, Conv	/
Iron [24]	Sparse	✓	✗	GEMM	/
Neujeans [27]	Dense	✓	✗	Conv	/
Bolt [7]	Dense	✓	✗	GEMM	Token Pruning
[38, 39, 37]	Dense	✗	✓	GEMM, Conv	Weight Pruning
PrivCirNet (ours)	Dense	✓	✓	GEMM, Conv	Block Circulant Transformation

Table 2: Comparison between PrivCirNet and previous works that use circulant matrix.

Method	Application	Initialization method	Variable block size	Block size assignment	Customized Encoding Method	Network
CirCNN [44] CirConv [45]	Convolution in plaintext	Forbenius norm	✓	Uniform/Manually set	/	ConvNets
Falcon [25]	End-to-end HE-based private inference	Forbenius norm	✗	Uniform	✗	Three-layer network
PrivCirNet (ours)	Hybrid HE+MPC private inference	Loss-aware	✓	Latency-aware block size assignment	✓	ConvNets, Transformers

Attempts have been made to use the circulant matrix to accelerate inference in plaintext [44, 45] and ciphertext [25] domains. However, two unresolved problems remain in both domains: 1) how to initialize circulant matrices, and 2) determining block sizes for each layer. As a result, it is hard for [44, 45, 25] to be applied to more efficient networks, e.g., MobileNetV2, Transformers, etc. Additionally, in the ciphertext domain, [25] cannot fully leverage block circulant matrices, resulting in limited or even increased latency. In contrast, PrivCirNet maximizes the potential of block circulant matrices by customizing the HE encoding algorithm and proposing new initialization and block size assignment algorithms, achieving a superior accuracy-latency trade-off. We give a comprehensive comparison between PrivCirNet and [44, 45, 25] in Table 2. We leave a more detailed review of existing works in Appendix A.

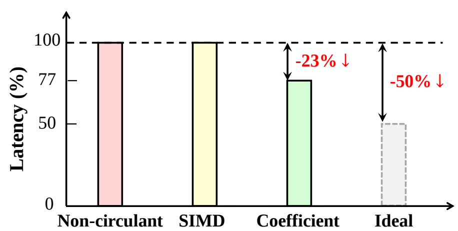

Figure 2: Directly using coefficient or SIMD encoding to block circulant GEMMs $((d_1, d_2, d_3, b) = (256, 192, 576, 2))$ leads to limited efficiency improvement.

Layer-wise block sizes	Top-1 Acc.	Latency
1-1-1-1	66.13	42 s
16-16-16-1	64.51	25 s
16-16-1-16	64.16	19 s
16-1-16-16	63.23	16 s
1-16-16-16	62.17	16 s

Table 3: Accuracy and latency impact of applying block circulant transformation to different layers of MobileNetV2 on Tiny ImageNet. 32 layers are partitioned into 4 groups.

3 PrivCirNet Framework

3.1 Motivation

While the circulant transformation enables to convert a GEMV into a HE-friendly 1-dimensional convolution, directly transforming the whole weight into a circulant matrix introduces large accuracy degradation due to the high compression ratio. We propose to leverage block circulant transformation and to trade off accuracy with efficiency by controlling the block sizes. However, we observe the following challenges that need to be addressed.

Challenge 1: existing encoding algorithms are incompatible with block circulant weight matrices. The computation of a GEMM with a block circulant weight matrix can be naturally decomposed into two steps, i.e., a circulant GEMV within each block and a general GEMM across blocks. Within each block, a circulant GEMV can be converted to a 1-dimensional convolution and be computed with a single HE-Pmult through coefficient encoding. However, when processing the GEMM across blocks, coefficient encoding suffers from either high communication cost [10, 24] or extensive HE rotations [22]. In contrast, while SIMD encoding can process the GEMM across blocks more efficiently [7], it still requires HE rotations to process the convolution within each block. As shown in

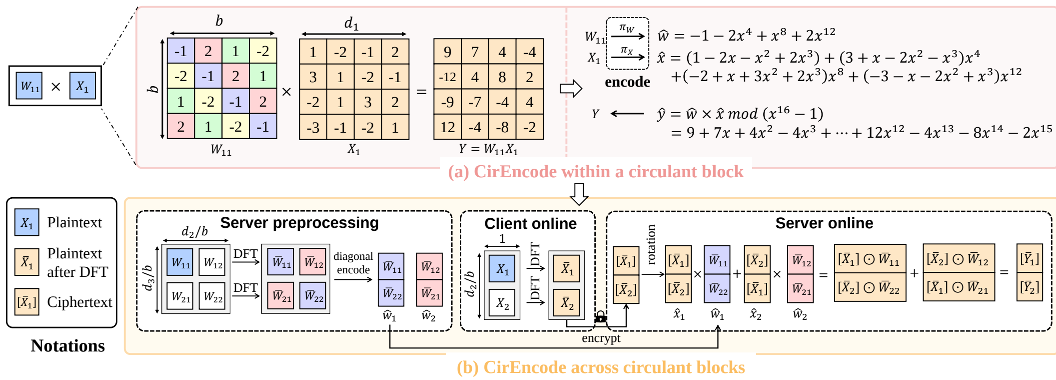

Figure 4: An example of CirEncode for block circulant GEMM where $(d_1, d_2, d_3, b) = (4, 8, 8, 4)$.

Figure 2, with existing encoding algorithms, block circulant transformation only introduces limited efficiency improvement. *Therefore, it is important to design the encoding algorithm to fully unleash the efficiency potential of the block circulant transformation.*

Challenge 2: accuracy and latency impact of block circulant transformation varies across layers. We apply the block circulant weight transformation with different block sizes to different layers of a MobileNetV2 on Tiny ImageNet. As shown in Table 3, the accuracy and latency impact on the MobileNetV2 varies significantly. *Hence, to better explore the Pareto optimal of efficiency and accuracy, layer-wise block size assignment becomes important*.

PrivCirNet Overview. In this paper, we introduce PrivCirNet, which features a joint optimization of the block circulant network and the private inference protocol. Figure 3 provides an overview of PrivCirNet. We first propose CirEncode for the GEMMs with block circulant weights in Section 3.2. Then, we develop a latency-aware optimization algorithm to determine the block sizes for each layer based on second-order information in Section 3.3. We also propose network-protocol co-fusion methods to further boost the inference efficiency in Section 3.4.

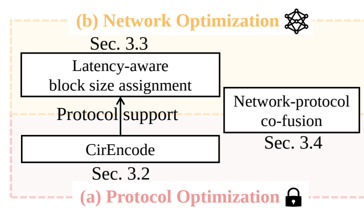

Figure 3: Overview of PrivCirNet.

3.2 CirEncode: nested encoding for block circulant GEMMs

High-level idea. Consider a GEMM $Y = WX$, where $Y \in \mathbb{Z}^{d_3 \times d_1}, W \in \mathbb{Z}^{d_3 \times d_2}, X \in \mathbb{Z}^{d_2 \times d_1}$. W is a block circulant matrix with block size b. Then, CirEncode encodes the GEMM following two steps: for each block with $W \in \mathbb{Z}^{b \times b}$ and $X \in \mathbb{Z}^{b \times d_1}$, we convert the computation into d_1 parallel GEMVs and leverage the coefficient encoding to avoid HE-Rot as shown in Figure 4 (a); then, for across blocks, we regard it as a GEMM and leverage the SIMD encoding to further reduce the HE-Rot as shown in Figure 4 (b). Thereby, CirEncode combines the advantages of both encoding schemes.

Encoding within a circulant block. We elaborately design the encoding rule for a circulant GEMM. Formally, we define two encoding functions $\pi_{\mathrm{W}} : \mathbb{Z}^{b \times b} \to \mathbb{A}_n$ and $\pi_{\mathrm{X}} : \mathbb{Z}^{b \times d_1} \to \mathbb{A}_n$ as follows:

$$\hat{w} = \pi_{\mathrm{W}}(W), \text{ where } \hat{w}[id_1] = W[i, 0], \quad \forall i \in [b], j \in [d_1]$$
$$\hat{x} = \pi_{\mathrm{X}}(X), \text{ where } \hat{x}[id_1 + j] = X[i, j], \quad \forall i \in [b], j \in [d_1]$$

where other coefficients of $\hat{w}$ are set to 0. $\hat{y} = \hat{w} \times \hat{x}$ directly gives the result of $Y = WX$ as described in Theorem 1 and we defer the proof to Appendix I.1. [2]

Theorem 1. *Given a circulant matrix $W \in \mathbb{Z}^{b \times b}$ and an input matrix $X \in \mathbb{Z}^{b \times d_1}$, where $bd_1 \leq n$, define two polynomials $\hat{w} = \pi_{\mathrm{W}}(W)$ and $\hat{x} = \pi_{\mathrm{X}}(X)$. Then, a GEMM $Y = WX \in \mathbb{Z}^{b \times d_1}$ can be evaluated by the polynomial multiplication $\hat{y} = \hat{w} \times \hat{x}$, where $Y[i, j] = \hat{y}[id_1 + j], \forall i \in [b], j \in [d_1]$.*

Compared with prior-art coefficient encoding algorithms for a GEMM, e.g., Iron [24], CirEncode features two key advantages: **1)** the encoding density, i.e., number of useful elements encoded per

[2] CirEncode uses $\bmod\ x^n - 1$ which is different from [10], the explanation is in Appendix I.1.

Table 4: Theoretical complexity comparison of CirEncode with prior works. The data of GEMM is measured with dimension $(d_1, d_2, d_3) = (512, 768, 3072)$, and that of convolution is $(H, W, C, K, R) = (16, 16, 128, 128, 3)$. The polynomial degree $n = 8192$ and block size $b = 8$.

Framework	GEMM			Convolution		
	# HE-Pmult	# HE-Rot	# Ciphertexts	# HE-Pmult	# HE-Rot	# Ciphertexts
CrypTFlow2 [6]	$O(d_1d_2d_3/n)$ 147456	$O(d_1(d_2+d_3)/n+d_3)$ 3312	$O(d_1(d_2+d_3)/n)$ 240	$O(HWCK/n)$ 9216	$O(HW(C+K)/n+K)$ 208	$O(HW(C+K)/n)$ 16
Cheetah [10]	$O(d_1d_2d_3/n)$ 147456	0	$O(d_1d_2/n+\lceil d_1/n\rceil d_3)$ 3120	$O(HWCK/n)$ 1408	0	$O(HWC/n+\lceil HW/n\rceil K)$ 134
Iron [24]	$O(d_1d_2d_3/n)$ 147456	0	$O(\sqrt{d_1d_2d_3/n})$ 960	$O(HWCKR^2/n)$ 12672	0	$O(\sqrt{HWCKR^2/n})$ 257
Bumblebee [22]	$O(d_1d_2d_3/n)$ 147456	$O(d_1d_3\log_2 n/(2\sqrt{n}))$ 6144	$O(d_1(d_2+d_3)/n)$ 240	$O(HWCK/n)$ 1408	$O(HWK\log_2 n/(2\sqrt{n}))$ 256	$O(HW(C+K)/n)$ 16
Neujeans+BSGS [27]	$O(d_1d_2d_3/n)$ 147456	$O(\sqrt{d_1d_2d_3/n})$ 588	$O(d_1(d_2+d_3)/n)$ 240	$O(HWCK/n)$ 1024	$O(\sqrt{HWCK/n})$ 48	$O(HW(C+K)/n)$ 16
Bolt+BSGS [7]	$O(d_1d_2d_3/n)$ 147456	$O(\sqrt{d_1d_2d_3/n})$ 528	$O(d_1(d_2+d_3)/n)$ 240	$O(HWCKR^2/n)$ 11700	$O(\sqrt{HWCKR^2/n})$ 106	$O(HW(CR^2+K)/n)$ 100
CirEncode (ours)	GEMM with circulant weight matrix			Convolution with circulant weight kernel		
	$O(d_1d_2d_3/(nb))$ 18432	$O(\sqrt{d_1d_2d_3/(nb)})$ 48	$O(d_1(d_2+d_3)/n)$ 240	$O(HWCK/(nb))$ 128	$O(\sqrt{HWCK/(nb)})$ 8	$O(HW(C+K)/n)$ 16

polynomial, is much higher, minimizing the communication cost; **2)** the input and output of a GEMM follow the same encoding rule described above, enabling layer fusion in Section 3.4.

Encoding across circulant blocks. Consider each circulant block as a unit, the computation across blocks can be regarded as a GEMM with dimension $(1, \frac{d_2}{b}, \frac{d_3}{b})$. We apply the SIMD diagonal encoding to pack different circulant blocks in parallel and use DFT for each block to transform the coefficient encoding into the SIMD encoding format, as shown in Figure 4 (b). Similar to Lemma 1, the correctness is given by Theorem 2 and we defer the proof to Appendix I.2.

Theorem 2. *Given M circulant weight matrices $W_0, \dots, W_{M-1} \in \mathbb{Z}^{b\times b}$ and input matrices $X_0, \dots, X_{M-1} \in \mathbb{Z}^{b\times d_1}$, define polynomials $\hat{w}_m$ and $\hat{x}_m$ with $m \in [M]$ following the coefficient packing in Theorem 1. Then, $Y_m = W_m X_m$ can be evaluated simultaneously through the polynomial multiplication in SIMD encoding:*

$$\begin{aligned} &\langle \mathrm{DFT}(\hat{y}_0) | \dots | \mathrm{DFT}(\hat{y}_{M-1}) \rangle_{\mathrm{Coeff}} \\ =& \langle \mathrm{DFT}(\hat{w}_0) | \dots | \mathrm{DFT}(\hat{w}_{M-1}) \rangle_{\mathrm{SIMD}} \times \langle \mathrm{DFT}(\hat{x}_0) | \dots | \mathrm{DFT}(\hat{x}_{M-1}) \rangle_{\mathrm{SIMD}} , \end{aligned}$$

where $|$ represents concatenation of polynomial coefficients and $Y_m[i,j] = \hat{y}_m[id_1 + j], \forall i \in [b], j \in [d_1], m \in [M]$.

We further extend the BSGS algorithm [7] to CirEncode with details in Appendix B. We also design CirEncode for block circulant convolutions as described in Appendix C.

Theoretical complexity analysis. Table 4 shows the theoretical complexity comparison of CirEncode with prior-art encoding methods in the number of HE-Pmult, HE-Rot, and ciphertexts. CirEncode computes a (d_1, b, b) circulant GEMM with only $O(bd_1/n)$ HE-Pmult and 0 HE-Rot. Therefore, compared to the SOTA scheme, i.e., Bolt and Neujeans, CirEncode reduces the number of HE-Pmult and HE-Rot by a factor of b and $\sqrt{b}$, respectively, for both GEMM and convolution. A detailed proof of theoretical complexity is available in Appendix B.

3.3 Latency-aware block size assignment with loss-aware initialization

Previous works use uniform block size [44, 25] or manually set the block sizes [45] for each layer, resulting in sub-optimal performance. We now propose a novel latency-aware block size assignment algorithm based on second-order information together with loss-aware initialization, which achieves a superior Pareto front of latency and accuracy.

Loss-aware initialization for circulant matrices. Previously, circulant matrices were initialized by minimizing the Frobenius norm between the non-circulant and circulant matrices [46, 45], i.e., $\min \|W_i' - W_i\|_2^2$, where W_i' represents the weight matrix after the circulant transformation of layer i. While this method minimizes the min square error (MSE) of the weight matrix, it overlooks that *the network accuracy has different sensitivity towards the MSE of different layers.* Therefore, we propose to directly assess the final loss instead of MSE for the transformation with Taylor expansion:

$$\mathcal{L}_{W_i'}(\mathcal{D}) - \mathcal{L}_{W_i}(\mathcal{D}) = \frac{\partial \mathcal{L}^\top(\mathcal{D})}{\partial W_i} \Delta W_i + \frac{1}{2} \Delta W_i^\top H \Delta W_i + \mathcal{O}\left(\|\Delta W_i\|^3\right), \tag{1}$$

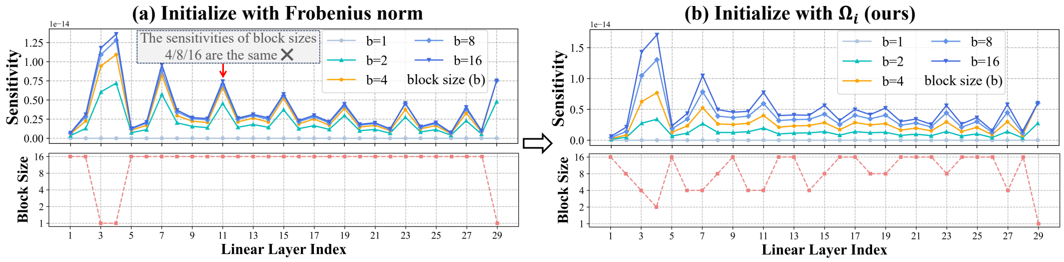

Figure 5: Layer-wise sensitivity and block size visualization for ViT on CIFAR-100.

where $\mathcal{L}$ is the task loss, $\mathcal{D}$ is the training dataset, H is the Hessian matrix and $\Delta W_i = W_i' - W_i$. The first term can be neglected as the model has already converged on the training dataset [47]. The Hessian matrix can be approximated using diagonal Fisher information matrix [48]. We then define the sensitivity of layer i as Ω_i:

$$\Omega_i = \Delta W_i^\top H \Delta W_i \approx \Delta W_i^\top \operatorname{diag}\left(\left(\frac{\partial \mathcal{L}(\mathcal{D})}{\partial W_i}\right)^2\right) \Delta W_i \tag{2}$$

Hence, we propose initializing the circulant matrix by minimizing Ω_i instead of the Frobenius norm, which can be solved analytically as $W_i' = \mathbb{E}\left[W_i \odot \left(\frac{\partial \mathcal{L}(\mathcal{D})}{\partial W_i}\right)^2\right]_{diag}$. $\mathbb{E}_{diag}$ is the expectation of each diagonal of a matrix. An example is provided in Appendix E.

Latency-aware block size assignment. Given an L-layer network, we denote the block size of each layer as $\{b_1, \dots, b_L\}$, where $b_i \in \{2^0, \dots, 2^{k-1}\}$. The search space contains k^L possible architectures, which can be extremely large, e.g., 2×10^{22} for $k = 5, L = 32$, rendering exhaustive search impractical. Therefore, we propose to formulate the search problem as an integer linear programming (ILP) problem, aiming to minimize the overall network sensitivity under the latency constraint [49, 50, 51]:

$$\text{Objective: } \min_{\{b_i\}_{i=1}^{L}} \quad \sum_{i=1}^{L} \Omega_i^{b_i}, \quad \text{Subject to: } \quad \sum_{i=1}^{L} \text{LAT}_i^{b_i} \le \text{Latency Limit} \tag{3}$$

Here, $\Omega_i^{b_i}$ is the i-th layer's sensitivity with block size b_i, $\text{LAT}_i^{b_i}$ is the associated latency in private inference. $\text{LAT}_i^{b_i}$ can be pre-computed given the dimension of the layer.

Visualization analysis. We visualize the layer-wise sensitivity and the searched structure of different initialization methods in Figure 5. As we can observe in Figure 5 (a), the previous method fails to tell the different sensitivity of block size 4, 8, and 16 for most of the layers. In contrast, our method, depicted in Figure 5 (b), better captures the effects of varying block sizes on task loss.

3.4 Network-Protocol Co-Fusion

Circulant ConvBN Fusion. During the inference, convolution (conv) and batch normalization (bn) layers are usually fused for lower latency. However, naïve fusion destroys the block circulant structure. Hence, we propose a fusion method for circulant conv and bn. Consider the learnable scaling factor $\gamma \in \mathbb{Z}^C$ for a bn layer. We combine the elements of γ into groups of size b and set $\gamma' \in \mathbb{Z}^C$ such that $\gamma'[i] = \frac{\sum_{j=0}^{b-1} \gamma[i+j-(i \mod b)]}{b}, \forall i \in [C]$. We use the same strategy for the learnable bias, running mean and variance, which maintains the circulant structure after fusion.

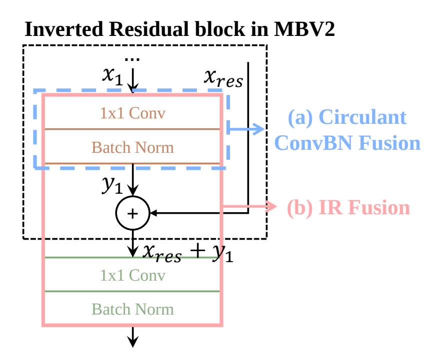

Figure 6: Network-Protocol Co-Fusion.

Inverted Residual (IR) Fusion Protocol. In the hybrid HE/MPC-based DNN inference, the network is

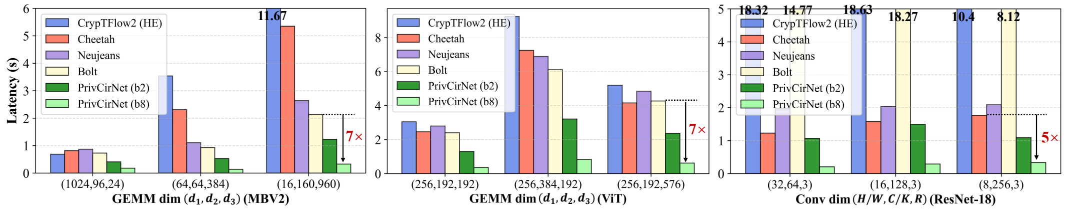

Figure 7: Latency comparison of different protocols for GEMMs and convolutions. PrivCirNet use circulant weight with block size b.

evaluated layer by layer. We identify the potential for layer fusion of consecutive linear layers in MobileNetV2 [28]. Figure 6 (b) shows where we implement fusion, aiming to compute $\text{convbn}(x_{res} + \text{convbn}(x_1))$ all together. Thanks to the encoding consistency provided by CirEncode, we can fuse layers with equal block size. Details of the fusion algorithm are in Appendix F.

4 Experiments

4.1 Experimental Setup

Implementation. PrivCirNet is built on top of the SEAL library [52] in C++. We use the OpenCheetah [10] to evaluate Cheetah [10] and CrypTFlow2 [6]. We also implement Falcon [26], Neujeans [27] and Bolt [7] protocols. Following [10, 53, 54], we simulate a LAN network setting via Linux Traffic Control, where the bandwidth is 384 MBps and the echo latency is 0.3ms. All the experiments are performed on a machine with 2.4 GHz Intel Xeon CPU. Following [7], we set $n = 8192$, security parameter $\lambda = 128$, plaintext bitwidth $p = 41$ and ciphertext bitwidth $q = 218$, which is also the default setting in SEAL library [52].

Datasets and Models. We evaluate PrivCirNet on MobileNetV2 [28], ResNet-18 [55], and ViT [29] across four datasets: CIFAR-10, CIFAR-100, Tiny ImageNet and ImageNet.[3] Detailed model architectures and training settings can be found in Appendix G.

Baselines. We compare PrivCirNet with prior-art HE-based DNN inference frameworks, including CrypTFlow2 [6], Cheetah [10], Falcon [26], Neujeans [27] and Bolt [7]. We also compare with SpENCNN [37] which is the SOTA HE-friendly pruning method.

4.2 Micro-Benchmark on Single GEMM and Convolution

Latency comparison. In Figure 7, we benchmark PrivCirNet on both GEMMs and convolutions with different block sizes. The layer dimensions are chosen from MobileNetV2, ResNet-18, and ViT. It can be observed that PrivCirNet supports both GEMMs and convolutions efficiently. Compared with Bolt and Cheetah, PrivCirNet (b8), i.e., block size of 8, achieves $5 \sim 7\times$ latency reduction. With PrivCirNet (b2), we can reduce latency by $1.7\times$ on average.

The number of HE-Pmult and HE-Rot comparison. In Table 5, we show the number of HE-Rot and HE-Pmult comparisons with different protocols. The layer dimensions are chosen from MobileNetV2, ResNet-18, and ViT. It can be observed that: **1)** Compared with SOTA algorithms Bolt and Neujeans, PrivCirNet (b8) achieves on average $7\times$ HE-Rot reduction and $8.5\times$ HE-Pmult reduction. And PrivCirNet (b2) achieves on average $2.1\times$ HE-Rot reduction and $1.9\times$ HE-Pmult reduction which is consistent with the theoretical complexity. **2)** PrivCirNet supports both GEMM and convolution efficiently. On the contrary, Neujeans performs worse in GEMM while Bolt performs worse in convolution.

4.3 End-to-End Inference Evaluation

In Figure 8 and Figure 9, we benchmark PrivCirNet at the full network scale and plot the Pareto front of accuracy and **latency of linear layers**. We make the following observation:

[3] Each of the models in the paper is capable of only classifying to the ImageNet 1k categories.

Table 5: The number of HE-Rot / HE-Pmult comparisons of different protocols for GEMMs and convolutions with different dimensions.

Method	MobileNetV2			ViT			ResNet-18			Average
	(1024,96,24)	(64,64,384)	(16,160,960)	(256,192,192)	(256,192,576)	(256,384,192)	(32,64,3)	(16,128,3)	(8,256,3)	
Neujeans+BSGS [27]	32 / 288	44 / 384	88 / 1024	90 / 1152	150 / 3456	120 / 2304	32 / 1024	48 / 1024	42 / 1134	72 / 1310
Bolt+BSGS [7]	21 / 288	33 / 384	55 / 960	60 / 1152	94 / 3456	78 / 2304	63 / 9216	106 / 11700	116 / 4608	70 / 2504
PrivCirNet (b2)	9 / 144	21 / 192	37 / 480	36 / 576	60 / 1728	54 / 1152	16 / 512	32 / 726	28 / 567	**33 / 675**
PrivCirNet (b8)	0 / 36	7 / 48	15 / 120	12 / 144	18 / 432	18 / 288	0 / 64	8 / 128	12 / 135	**10 / 155**

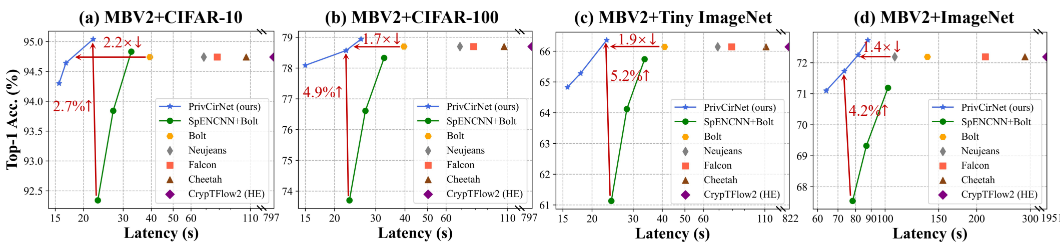

Figure 8: Comparison with SpENCNN and other prior-art protocols on MobileNetV2.

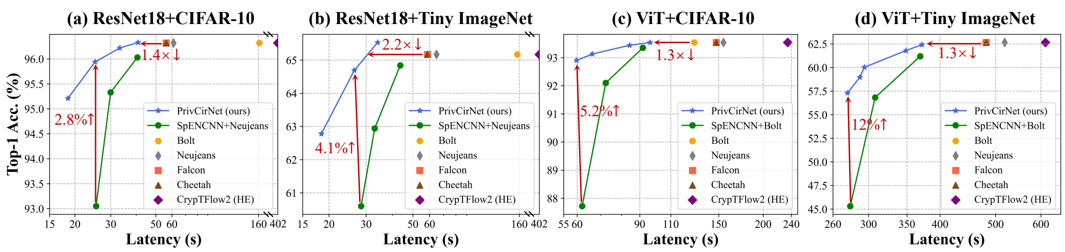

Figure 9: Comparison with SpENCNN and other prior-art protocols on ResNet-18 and ViT.

Comparison with prior-art HE-based frameworks. PrivCirNet consistently outperforms prior-art frameworks, including Bolt, Neujeans, Falcon, etc, in both ConvNets and Transformers. Specifically, on Tiny ImageNet, compared with Bolt, PrivCirNet achieves $1.9\times, 5.0\times, 1.3\times$ latency reduction with iso-accuracy on MobileNetV2, ResNet-18, and ViT, respectively. Compared to Cheetah, PrivCirNet achieves $1.3 \sim 4.8\times$ latency reduction with iso-accuracy across three models.

Comparison with prior-art structured pruning method SpENCNN. PrivCirNet achieves SOTA accuracy/latency Pareto front across different datasets and models. Especially in larger compression ratios, SpENCNN suffers from huge accuracy loss. In comparison, PrivCirNet outperforms SpENCNN by 5.2% on MobileNetV2, 4.1% on ResNet-18, and 12% on ViT on Tiny ImageNet.

Benchmark on ImageNet. We benchmark PrivCirNet on ImageNet with MobileNetV2 in Figure 8 (d). PrivCirNet achieves $1.4\times$ latency reduction compared with prior-art framework Neujeans and achieves 4.2% accuracy improvement over SpENCNN with lower latency.

4.4 Ablation Study

Effectiveness of latency-aware block size assignment. Table 6 shows the comparison of different block size assignment methods, including uniform block size, mixed block sizes with Frobenius norm initialization [46, 45], and mixed block sizes with loss-aware initialization. According to the results, we find that: **1)** PrivCirNet achieves the highest accuracy across most datasets and models. **2)** PrivCirNet exhibits enhanced performance at higher compression ratios, emphasizing the importance of latency-aware block size assignment in networks with limited capacity.

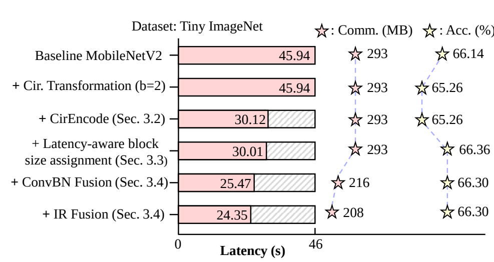

Figure 10: Ablation study of our proposed optimizations in PrivCirNet on MobileNetV2.

Table 6: Accuracy comparison of different block size assignment methods. Latency limitation represents the proportion of latency relative to the original uncompressed model.

Method	Latency Limitation	Top-1 Acc. ↑					
		MobileNetV2			ViT		
		CIFAR-10	CIFAR-100	Tiny ImageNet	CIFAR-10	CIFAR-100	Tiny ImageNet
Uncompressed	100%	94.74	78.70	66.14	93.54	74.77	62.65
Uniform	50%	94.81 (-0.23)	77.98 (-0.60)	65.26 (-1.10)	93.38 (-0.06)	74.41 (-0.30)	**61.87** (+0.08)
	25%	93.97 (-0.33)	76.30 (-1.41)	62.76 (-2.07)	92.57 (-0.33)	72.00 (-0.76)	58.11 (-0.85)
	12.5%	92.71 (-0.55)	73.89 (-0.96)	60.34 (-1.50)	90.98 (-0.46)	67.51 (-2.22)	51.90 (-2.16)
Frobenius	50%	94.71 (-0.35)	78.28 (-0.30)	65.98 (-0.38)	93.40 (-0.04)	74.58 (-0.13)	61.33 (-0.46)
	25%	94.23 (-0.07)	76.38 (-1.33)	63.76 (-1.07)	92.40 (-0.50)	72.07 (-0.69)	58.00 (-0.96)
	12.5%	92.65 (-0.61)	74.32 (-0.53)	61.14 (-0.70)	90.32 (-1.12)	68.02 (-1.71)	51.92 (-2.14)
Loss-aware (PrivCirNet)	50%	**95.04**	**78.58**	**66.36**	**93.44**	**74.71**	61.79
	25%	**94.30**	**77.71**	**64.83**	**92.90**	**72.76**	**58.96**
	12.5%	**93.26**	**74.85**	**61.84**	**91.44**	**69.73**	**54.06**

Effectiveness of different optimizations in PrivCirNet. We demonstrate the effectiveness of the proposed optimizations by adding them step by step on MobileNetV2 and Tiny ImageNet. As in Figure 10, we observe that: **1)** without CirEncode, circulant transformation harms the accuracy and cannot reduce latency due to incompatibility with existing encoding algorithms; **2)** latency-aware block size assignment significantly improves the accuracy and even outperforms the uncompressed model; **3)** the fusion methods reduce both the latency and communication with negligible accuracy loss.

Additional Results. We present extra experiments to show 1) latency breakdown, and 2) comparison on more networks in Appendix H.

5 Limitation and Future Work

PrivCirNet focuses on improving the HE computation efficiency, which accounts for 75% total latency and is the bottleneck in the hybrid HE/MPC scheme. We can also extend PrivCirNet with activation function optimization methods, e.g., ReLU pruning method SNL [30]. As shown in Table 7, we prune 50% ReLUs in PrivCirNet (b2) without accuracy loss, achieving $2\times$ latency reduction in nonlinear layers. We regard a more in-depth study of joint linear/nonlinear layer optimization as our future work.

Method (CIFAR-100)	Top-1 Acc.	Nonlinear latency	Total latency
Original ResNet-18	76.52	12.64 s	73.72 s
PrivCirNet (b2)	76.93	12.64 s	45.76 s
+SNL(-50% ReLU)	76.72	6.32 s	39.44 s
+SNL(-60% ReLU)	76.27	5.06 s	38.18 s

Table 7: Extend PrivCirNet with nonlinear layer optimization method SNL.

6 Conclusion

In this paper, we introduce PrivCirNet, a network/protocol co-optimization framework to enhance the efficiency of HE-based DNN inference. PrivCirNet leverages block circulant transformation to reduce the HE computation. PrivCirNet features a novel encoding method, CirEncode, and a latency-aware block size assignment algorithm. PrivCirNet significantly improves the network-level inference efficiency while maintaining a high accuracy. PrivCirNet achieves a latency reduction of $1.3 \sim 5.0\times$ compared to Bolt in MobileNetV2, ResNet-18 and ViT. Moreover, when compared with SpENCNN, PrivCirNet attains up to 12% higher accuracy, demonstrating a high potential to accelerate private inference across both ConvNets and Transformers.

7 Acknowledgement

This work was partly supported by Beijing Municipal Science and Technology Program (No. Z241100004224015), Ant Group, and the 111 Project (B18001).

References

[1] Neda Azouji, Ashkan Sami, and Mohammad Taheri. Efficientmask-net for face authentication in the era of covid-19 pandemic. *Signal, Image and Video Processing*, 16(7):1991–1999, 2022.

[2] Georgios Kaissis, Alexander Ziller, Jonathan Passerat-Palmbach, Théo Ryffel, Dmitrii Usynin, Andrew Trask, Ionésio Lima Jr, Jason Mancuso, Friederike Jungmann, Marc-Matthias Steinborn, et al. End-to-end privacy preserving deep learning on multi-institutional medical imaging. *Nature Machine Intelligence*, 3(6):473–484, 2021.

[3] Frank F Xu, Uri Alon, Graham Neubig, and Vincent Josua Hellendoorn. A systematic evaluation of large language models of code. In *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming*, pages 1–10, 2022.

[4] Woo-Seok Choi, Brandon Reagen, Gu-Yeon Wei, and David Brooks. Impala: Low-latency, communication-efficient private deep learning inference, May 2022.

[5] Kanav Gupta, Deepak Kumaraswamy, Nishanth Chandran, and Divya Gupta. Llama: A low latency math library for secure inference. *Proceedings on Privacy Enhancing Technologies*, 2022(4):274–294, Sep 2022.

[6] Deevashwer Rathee, Mayank Rathee, Nishant Kumar, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. Cryptflow2: Practical 2-party secure inference. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 325–342, 2020.

[7] Q. Pang, J. Zhu, H. Möllering, W. Zheng, and T. Schneider. Bolt: Privacy-preserving, accurate and efficient inference for transformers. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 133–133, Los Alamitos, CA, USA, may 2024. IEEE Computer Society.

[8] 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*, pages 201–210. PMLR, 2016.

[9] Chiraag Juvekar, Vinod Vaikuntanathan, and AnanthaP. Chandrakasan. GAZELLE: A low latency framework for secure neural network inference, Jan 2018.

[10] Zhicong Huang, Wen-jie Lu, Cheng Hong, and Jiansheng Ding. Cheetah: Lean and fast secure {Two-Party} deep neural network inference. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 809–826, 2022.

[11] Qian Lou and Lei Jiang. She: A fast and accurate deep neural network for encrypted data. *Advances in neural information processing systems*, 32, 2019.

[12] 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, 2022.

[13] 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*, pages 12403–12422. PMLR, 2022.

[14] Jaiyoung Park, Donghwan Kim, Jongmin Kim, Sangpyo Kim, Wonkyung Jung, Jung Hee Cheon, and Jung Ho Ahn. Toward practical privacy-preserving convolutional neural networks exploiting fully homomorphic encryption. *arXiv preprint arXiv:2310.16530*, 2023.

[15] Dongwoo Kim and Cyril Guyot. Optimized privacy-preserving cnn inference with fully homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 18:2175–2187, 2023.

[16] Shengyu Fan, Zhiwei Wang, Weizhi Xu, Rui Hou, Dan Meng, and Mingzhe Zhang. Tensorfhe: Achieving practical computation on encrypted data using gpgpu. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 922–934. IEEE, 2023.

[17] Jongmin Kim, Gwangho Lee, Sangpyo Kim, Gina Sohn, Minsoo Rhu, John Kim, and Jung Ho Ahn. Ark: Fully homomorphic encryption accelerator with runtime data generation and inter-operation key reuse. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1237–1254. IEEE, 2022.

[18] George Onoufriou, Paul Mayfield, and Georgios Leontidis. Fully homomorphically encrypted deep learning as a service. *Machine Learning and Knowledge Extraction*, 3(4):819–834, 2021.

[19] Jian Liu, Mika Juuti, Yao Lu, and N. Asokan. Oblivious neural network predictions via minionn transformations. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, Oct 2017.

[20] Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and RalucaAda Popa. Delphi: A cryptographic inference service for neural networks, Jan 2020.

[21] Karthik Garimella, Zahra Ghodsi, NandanKumar Jha, Siddharth Garg, and Brandon Reagen. Characterizing and optimizing end-to-end systems for private inference, Jul 2022.

[22] Wen-jie Lu, Zhicong Huang, Zhen Gu, Jingyu Li, Jian Liu, Kui Ren, Cheng Hong, Tao Wei, and WenGuang Chen. Bumblebee: Secure two-party inference framework for large transformers. *Cryptology ePrint Archive*, 2023.

[23] Donghwan Kim, Jaiyoung Park, Jongmin Kim, Sangpyo Kim, and Jung Ho Ahn. Hyphen: A hybrid packing method and optimizations for homomorphic encryption-based neural networks. *arXiv preprint arXiv:2302.02407*, 2023.

[24] Meng Hao, Hongwei Li, Hanxiao Chen, Pengzhi Xing, Guowen Xu, and Tianwei Zhang. Iron: Private inference on transformers. In *Advances in Neural Information Processing Systems*, 2022.

[25] Qian Lou, Wen-jie Lu, Cheng Hong, and Lei Jiang. Falcon: Fast spectral inference on encrypted data. *Advances in Neural Information Processing Systems*, 33:2364–2374, 2020.

[26] Tianshi Xu, Meng Li, Runsheng Wang, and Ru Huang. Falcon: Accelerating homomorphically encrypted convolutions for efficient private mobile network inference. *arXiv preprint arXiv:2308.13189*, 2023.

[27] Jae Hyung Ju, Jaiyoung Park, Jongmin Kim, Donghwan Kim, and Jung Ho Ahn. Neujeans: Private neural network inference with joint optimization of convolution and bootstrapping. *The ACM Conference on Computer and Communications Security (CCS)*, 2024.

[28] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[29] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[30] Minsu Cho, Ameya Joshi, Brandon Reagen, Siddharth Garg, and Chinmay Hegde. Selective network linearization for efficient private inference. In *International Conference on Machine Learning*, pages 3947–3961. PMLR, 2022.

[31] Nandan Kumar Jha, Zahra Ghodsi, Siddharth Garg, and Brandon Reagen. Deepreduce: Relu reduction for fast private inference. In *International Conference on Machine Learning*, pages 4839–4849. PMLR, 2021.

[32] Souvik Kundu, Shunlin Lu, Yuke Zhang, Jacqueline Liu, and Peter A Beerel. Learning to linearize deep neural networks for secure and efficient private inference. *arXiv preprint arXiv:2301.09254*, 2023.

[33] Minsu Cho, Zahra Ghodsi, Brandon Reagen, Siddharth Garg, and Chinmay Hegde. Sphynx: A deep neural network design for private inference. *IEEE Security & Privacy*, 20(5):22–34, 2022.

[34] Hongwu Peng, Shaoyi Huang, Tong Zhou, Yukui Luo, Chenghong Wang, Zigeng Wang, Jiahui Zhao, Xi Xie, Ang Li, Tony Geng, et al. Autorep: Automatic relu replacement for fast private network inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5178–5188, 2023.

[35] Wenxuan Zeng, Meng Li, Wenjie Xiong, Wenjie Lu, Jin Tan, Runsheng Wang, and Ru Huang. Mpcvit: Searching for mpc-friendly vision transformer with heterogeneous attention. *arXiv preprint arXiv:2211.13955*, 2022.

[36] Souvik Kundu, Yuke Zhang, Dake Chen, and Peter A Beerel. Making models shallow again: Jointly learning to reduce non-linearity and depth for latency-efficient private inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4685–4689, 2023.

[37] Ran Ran, Xinwei Luo, Wei Wang, Tao Liu, Gang Quan, Xiaolin Xu, Caiwen Ding, and Wujie Wen. Spencnn: orchestrating encoding and sparsity for fast homomorphically encrypted neural network inference. In *International Conference on Machine Learning*, pages 28718–28728. PMLR, 2023.

[38] Yifei Cai, Qiao Zhang, Rui Ning, Chunsheng Xin, and Hongyi Wu. Hunter: He-friendly structured pruning for efficient privacy-preserving deep learning. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, pages 931–945, 2022.

[39] Ehud Aharoni, Moran Baruch, Pradip Bose, Alper Buyuktosunoglu, Nir Drucker, Subhankar Pal, Tomer Pelleg, Kanthi Sarpatwar, Hayim Shaul, Omri Soceanu, et al. He-pex: Efficient machine learning under homomorphic encryption using pruning, permutation and expansion. *arXiv preprint arXiv:2207.03384*, 2022.

[40] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive*, 2012.

[41] Ran Ran, Nuo Xu, Tao Liu, Wei Wang, Gang Quan, and Wujie Wen. Penguin: Parallel-packed homomorphic encryption for fast graph convolutional network inference. *Advances in Neural Information Processing Systems*, 36, 2024.

[42] Qiao Zhang, Chunsheng Xin, and Hongyi Wu. Gala: Greedy computation for linear algebra in privacy-preserved neural networks. *arXiv preprint arXiv:2105.01827*, 2021.

[43] Oded Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, 78(110):1–108, 1998.

[44] Caiwen Ding, Siyu Liao, Yanzhi Wang, Zhe Li, Ning Liu, Youwei Zhuo, Chao Wang, Xuehai Qian, Yu Bai, Geng Yuan, et al. Circnn: accelerating and compressing deep neural networks using block-circulant weight matrices. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 395–408, 2017.

[45] Siyu Liao and Bo Yuan. Circconv: A structured convolution with low complexity. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4287–4294, 2019.

[46] Moody T. Chu and Robert J. Plemmons. Real-valued, low rank, circulant approximation. *SIAM Journal on Matrix Analysis and Applications*, page 645–659, Jan 2003.

[47] Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. *arXiv preprint arXiv:2102.05426*, 2021.

[48] Genevieve B Orr and Klaus-Robert Müller. *Neural networks: tricks of the trade*. Springer, 1998.

[49] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 293–302, 2019.

[50] Zhen Dong, Zhewei Yao, Daiyaan Arfeen, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq-v2: Hessian aware trace-weighted quantization of neural networks. *Advances in neural information processing systems*, 33:18518–18529, 2020.

[51] Zhewei Yao, Zhen Dong, Zhangcheng Zheng, Amir Gholami, Jiali Yu, Eric Tan, Leyuan Wang, Qijing Huang, Yida Wang, Michael Mahoney, et al. Hawq-v3: Dyadic neural network quantization. In *International Conference on Machine Learning*, pages 11875–11886. PMLR, 2021.

[52] Microsoft SEAL (release 3.6). https://github.com/Microsoft/SEAL, November 2020. Microsoft Research, Redmond, WA.

[53] Liyan Shen, Ye Dong, Binxing Fang, Jinqiao Shi, Xuebin Wang, Shengli Pan, and Ruisheng Shi. Abnn 2. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, Aug 2022.

[54] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE symposium on security and privacy (SP)*, pages 19–38. IEEE, 2017.

[55] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[56] Wenxuan Zeng, Meng Li, Haichuan Yang, Wen-jie Lu, Runsheng Wang, and Ru Huang. Copriv: Network/protocol co-optimization for communication-efficient private inference. *arXiv preprint arXiv:2311.01737*, 2023.

[57] Dacheng Li, Rulin Shao, Hongyi Wang, Han Guo, Eric P Xing, and Hao Zhang. Mpcformer: fast, performant and private transformer inference with mpc. *arXiv preprint arXiv:2211.01452*, 2022.

[58] Ali Hassani, Steven Walton, Nikhil Shah, Abulikemu Abuduweili, Jiachen Li, and Humphrey Shi. Escaping the big data paradigm with compact transformers. *arXiv preprint arXiv:2104.05704*, 2021.

[59] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10428–10436, 2020.

[60] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022.

[61] Nandan Kumar Jha and Brandon Reagen. Deepreshape: Redesigning neural networks for efficient private inference. *arXiv preprint arXiv:2304.10593*, 2023.

A Related Works

To improve the efficiency of HE-based DNN inference, existing works mainly focus on optimizing the HE encoding algorithm [10, 24, 26, 9, 6, 42, 27, 7] and the DNN architectures [31, 30, 56, 32, 33, 34, 35, 38, 39, 37, 25]. HE encoding optimizations focus on improving the encoding density (i.e., useful elements per polynomial) to reduce communication [24, 26, 22] and HE computations [10, 7, 27]. For example, Cheetah [10] proposes an efficient rotation free encoding algorithm for convolutions and Falcon [26] further improve the communication efficiency for group-wise convolution. Iron [24] and BubbleBee [22] optimize the encoding algorithm for general matrix multiplications (GEMMs). Neujeans [27] and Bolt [7] further introduce the baby-step giant-step (BSGS) algorithm to reduce the number of HE rotations.

DNN architecture optimizations focus on developing HE-friendly architectures to improve inference efficiency including HE-friendly activation approximation or pruning [31, 30, 56, 32, 33, 34], weight pruning [38, 39, 37], etc. [30, 31, 32, 33, 34] optimize the ReLU functions through pruning and approximation for communication and computation reduction. [35, 57] propose to prune and approximate GeLU functions for efficient private transformer inference. [37, 38, 39] propose HE-friendly structured pruning to reduce both HE rotations and multiplications.

B Baby-step Giant-step (BSGS) Algorithm for CirEncode

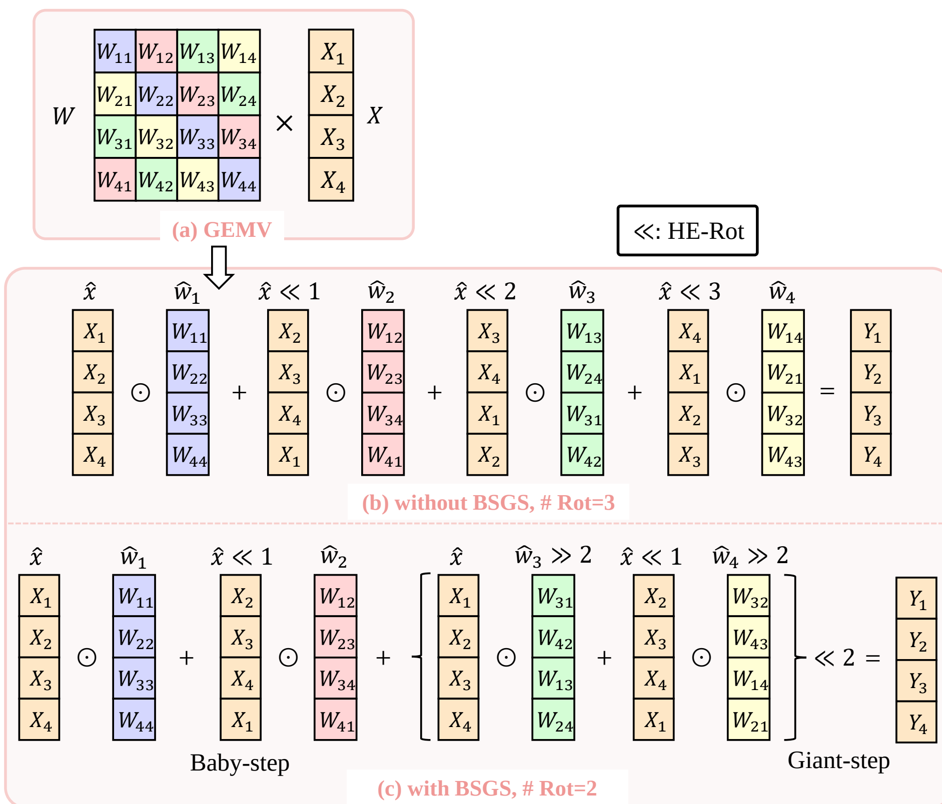

Figure 11: An example of GEMV using BSGS algorithm.

The BSGS algorithm is used for GEMV and GEMM to reduce the number of HE rotations [7, 27]. We visualize the high-level idea of the BSGS algorithm in Figure 11. Instead of rotating each input polynomial once, the BSGS algorithm divides the rotations into two steps: baby-step and giant-step which can be formulated as

$$\sum_{j=1}^{G}\left(\sum_{i=1}^{B}\hat{w}^{\text{diag}}_{(j-1)B+i}\odot(\hat{x} << (i-1))\right) << (j-1)B \tag{4}$$

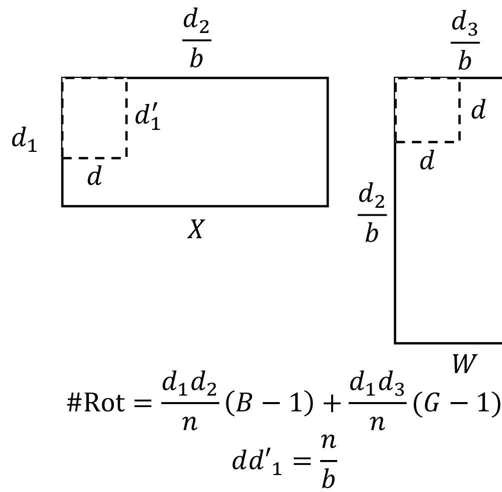

Figure 12: Illustration of our BSGS algorithm for block circulant GEMM with tiling.

Here, G, B are the number of giant-step and baby-step, respectively. The total number of rotations is reduced to $B + G - 2$. In GEMM with dimension (d_1, d_2, d_3), tiling is needed to split matrices into smaller blocks whose maximum size is HE polynomial degree n. Moreover, when extend BSGS to CirEncode, the dimension of GEMM becomes $(d_1, \frac{d_2}{b}, \frac{d_3}{b})$ and the polynomial degree becomes $\frac{n}{b}$. We do not encode the d_1 dimension into each circulant block, instead, we treat the computation cross blocks as a GEMM and use the BSGS algorithm to determine the tiling size of the d_1 dimension. Therefore, how to tile and choose B, G is crucial to minimize the number of rotations. We propose to formulate this optimization problem as a nonlinear programming problem as

$$\begin{aligned} \min \quad & \#\,\text{Rot} = \frac{d_1 d_2}{n}(B-1) + \frac{d_1 d_3}{n}(G-1) \\ \text{s.t.} \quad & B * G = d \\ & d_1' d = \frac{n}{b} \\ & d_1' \le d_1 \\ & d \le \min(\frac{d_3}{b}, \frac{d_2}{b}) \end{aligned} \tag{5}$$

We give an illustration of our BSGS algorithm in Figure 12. The tile sizes of input and weight are (d_1', d) and (d, d), respectively. The constraints in Equation 5 are derived from a tile containing at most n elements and a tile size cannot exceed the size of the matrix. This problem has a small solution space. With $B, G \le \min(\frac{d_3}{b}, \frac{d_2}{b})$, The solution space is at most $\min(\frac{d_3}{b}, \frac{d_2}{b})^2$, allowing us to directly solve it using a search algorithm with the complexity of $O((\min(\frac{d_3}{b}, \frac{d_2}{b})^2))$. Our experiments show that the search algorithm can find the optimal solution within milliseconds for all cases.

Complexity analysis of # Rot. We proof in Equation 6 that the complexity of #Rot with our BSGS algorithm is $O(\sqrt{d_1 d_2 d_3/(nb)})$.

$$\begin{aligned} \#\,\text{Rot} &= \frac{d_1 d_2}{n}(B-1) + \frac{d_1 d_3}{n}(G-1) \\ &\ge 2\frac{d_1}{n}\sqrt{d_2 d_3 (B-1)(G-1)} \\ &\iff d_2(B-1) = d_3(G-1) \\ O(\#\,\text{Rot}) &= O(\frac{d_1}{n}\sqrt{d_2 d_3 d}) \\ &= O(\frac{d_1}{n}\sqrt{d_2 d_3 n / b d_1}) \\ &= O(\sqrt{d_1 d_2 d_3/(nb)}) \end{aligned} \tag{6}$$

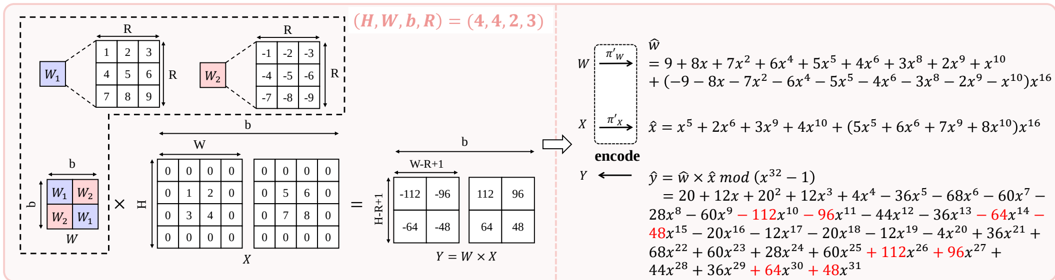

Figure 13: A toy example of CirEncode within a circulant convolution where $(H, W, b, R) = (4, 4, 2, 3)$.

Here we omit the last constraint in Equation 5 for simplicity.

Complexity analysis of # Mul. The complexity of # Mul is given by Equation 7.

$$
\begin{aligned}
&O(\text{\# Mul}) = O(\frac{d_2}{b} \cdot \frac{d_3}{b} \cdot \frac{bd_1}{n}) \\
&= O(d_1 d_2 d_3 / (nb))
\end{aligned}
\tag{7}
$$

Boundary cases. When $d_1 \min(\frac{d_3}{b}, \frac{d_2}{b}) < \frac{n}{b}$, the tile size of input will be $d_1 \min(\frac{d_3}{b}, \frac{d_2}{b})$ although it's not often the case. In addition, the second constraint in Equation 5 should actually be $[d_1' b]_{2^k}\, d = n$. $[\cdot]_{2^k}$ means the next nearest power of 2. This is because NTT requires the input size to be a power of 2. Consequently, we consider all these boundary conditions in the search algorithm in practice.

C CirEncode for Convolutions

In this section, we extend CirEncode to convolutions. We denote the input, weight and output of a block circulant convolution operation as $X \in \mathbb{Z}^{C \times H \times W}, W \in \mathbb{Z}^{K \times C \times R \times R}, Y = W \circledast X \in \mathbb{Z}^{K \times (H-R+1) \times (W-R+1)}$. Here $\circledast$ represents the convolution operation. We assume stride=1 for simplicity. W is a block circulant matrix with respect to the first two dimensions with block size b.

Encoding within a circulant block. For each circulant block, we define two encoding functions $\pi'_{\mathrm{X}} : \mathbb{Z}^{b \times H \times W} \to \mathbb{A}_n$ and $\pi'_{\mathrm{W}} : \mathbb{Z}^{b \times b \times R \times R} \to \mathbb{A}_n$ as follows:

$$
\begin{aligned}
&\hat{x} = \pi'_{\mathrm{X}}(X) \quad \text{s.t.} \quad \hat{x}[iHW + jW + k] = X[i, j, k], i \in [b], j \in [H], k \in [W] \\
&\hat{w} = \pi'_{\mathrm{W}}(W) \quad \text{s.t.} \quad \hat{w}[iHW + (W+1)(R-1) - jW - k] = W[i, 0, j, k], i \in [b], j \in [R], k \in [R]
\end{aligned}
$$

where other coefficients of $\hat{w}$ are set to 0. Multiplication of polynomials $\hat{y} = \hat{w} \times \hat{x}$ directly gives the result of $Y = W \circledast X$ as described in Theorem 3. We defer the proof to Appendix I.3.

Theorem 3. *Assuming $HWb \le n$, given a circulant convolution kernel $W \in \mathbb{Z}^{b \times b \times R \times R}$ and input tensor $X \in \mathbb{Z}^{b \times H \times W}$. Define two polynomials $\hat{w} = \pi'_{\mathrm{W}}(W)$ and $\hat{x} = \pi'_{\mathrm{X}}(X)$. The polynomial multiplication result $\hat{y} = \hat{w} \times \hat{x}$ directly maps to the result of $Y = W \circledast X \in \mathbb{Z}^{b \times (H-R+1) \times (W-R+1)}$ where $Y[i, j, k] = \hat{y}[iHW + (W+1)(R-1) + jW + k]$.*

We show a toy example of CirEncode for circulant convolution in Figure 13.

Encoding across circulant blocks. Consider each circulant block with input dimension (b, H, W) and weight dimension (b, b, R, R) as a basic unit. The computation across circulant blocks can be regarded as a GEMV with dimension $(1, \frac{C}{b}, \frac{K}{b})$. Then we leverage SIMD diagonal encoding which is the same as the block circulant GEMM.

BSGS algorithm for block circulant convolution. Similar to block circulant matrix multiplication, the BSGS algorithm for block circulant convolution can be formulated as an non-linear programming

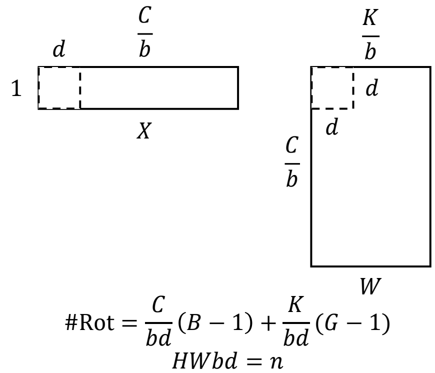

Figure 14: Illustration of our BSGS algorithm for block circulant convolution with tiling.

problem as

$$\begin{aligned} \min \quad & \text{\# Rot} = \frac{HWC}{n}(B-1) + \frac{HWK}{n}(G-1) \\ \text{s.t.} \quad & B * G = d \\ & HWbd = n \\ & d \le \min(\frac{C}{b}, \frac{K}{b}) \end{aligned} \tag{8}$$

We give an illustration in Figure 14 where the tile sizes of input and weight are $(1, d)$ and (d, d), respectively. This problem has a small solution space. With $B, G \le \min(\frac{C}{b}, \frac{K}{b})$, The solution space is at most $(\min(\frac{C}{b}, \frac{K}{b}))^2$, allowing us to directly solve it using a search algorithm with a complexity of $O((\min(\frac{C}{b}, \frac{K}{b}))^2)$. Our experiments show that the search algorithm can find the optimal solution within milliseconds for all cases.

Complexity analysis of # Rot. We proof in Equation 9 that the complexity of # Rot in block circulant convolution with our BSGS algorithm is $O(\sqrt{HWCK/(nb)})$.

$$\begin{aligned} \text{\# Rot} &= \frac{HWC}{n}(B-1) + \frac{HWK}{n}(G-1) \\ &\ge 2\frac{HW}{n}\sqrt{CK(B-1)(G-1)} \\ &\iff C(B-1) = K(G-1) \\ O(\text{\# Rot}) &= O(\frac{HW}{n}\sqrt{CKd}) \\ &= O(\frac{HW}{n}\sqrt{\frac{CKn}{HWb}}) \\ &= O(\sqrt{\frac{HWCK}{nb}}) \end{aligned} \tag{9}$$

Here we omit the last constraint in Equation 8 for simplicity.

Complexity analysis of # Mul. The complexity of # Mul is given by Equation 10.

$$\begin{aligned} O(\text{\# Mul}) &= O(\frac{C}{b} \cdot \frac{K}{b} \cdot \frac{HWb}{n}) \\ &= O(HWCK/(nb)) \end{aligned} \tag{10}$$

D Why does structured pruning fail in BSGS algorithm?

HE-friendly structured pruning [38, 37] reduces the number of rotations by pruning the diagonals of weight matrices. However, this technique is not feasible in the BSGS algorithm. Figure 15 demonstrates the limitations of structured pruning in BSGS. To illustrate, consider a GEMM where

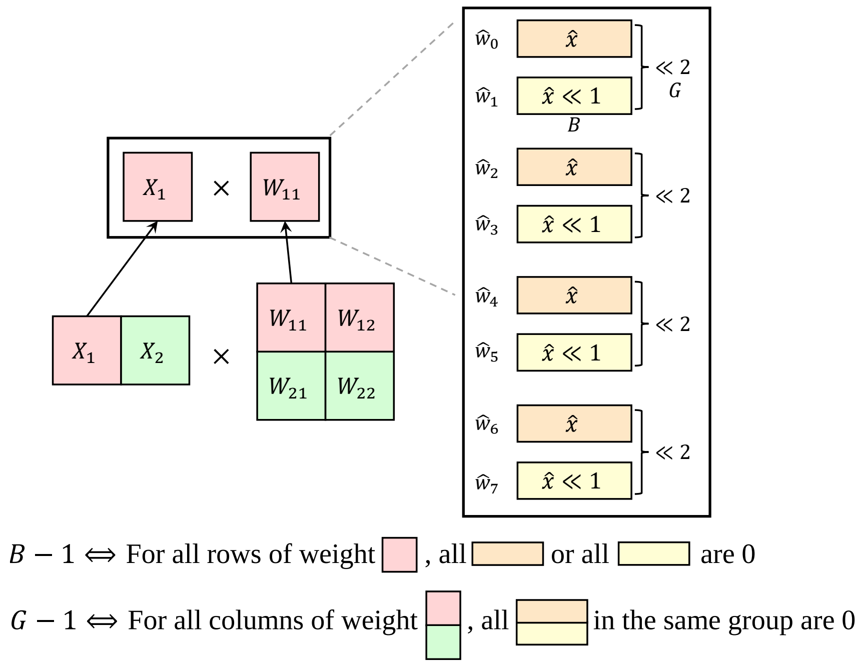

Figure 15: Illustration of the limitation of structured pruning in BSGS algorithm.

input and weight matrices are tiled into smaller blocks, such as X_1, X_2 and $W_{11}, W_{12}, W_{21}, W_{22}$. First focusing on the multiplication between X_1 and W_{11}, note that in BSGS, rotations are split into baby-step and giant-step. Assuming $B = 2, G = 4$, there are four groups, each containing two ciphertexts $(\hat{x}, \hat{x} \ll 1)$, and eight weight polynomials $\hat{w}_0, \ldots, \hat{w}_7$ which are the eight diagonals of the weight matrix W_{11}. Each group requires one baby-step rotation to achieve $\hat{x} \ll 1$ and one giant-step rotation. Pruning diagonals to reduce rotations in BSGS is challenging. For instance, to reduce a baby-step rotation, all diagonals in the same position across different groups, such as $\hat{w}_1, \hat{w}_3, \hat{w}_5, \hat{w}_7$, must be pruned. Additionally, considering tiling, X_1 must multiply with all weight matrices in the first row, i.e., W_{11}, W_{12}. Thus, to decrease a single baby-step rotation, diagonals in the same position across all groups for all first-row weight matrices must be pruned. A similar challenge exists for giant-step rotations; to reduce one giant-step rotation, entire groups like $\hat{w}_0, \hat{w}_1$, in all first-column of the weight matrices must be pruned. Consequently, it is difficult for existing structured pruning methods to meet these constraints, leading to the limitation of reducing the number of rotations.

E An example of our loss-aware initialization for circulant matrices

We give an example of our circulant transformation initialization in Equation 11. The input matrix W is a 2×2 matrix and the values of W and $\frac{\partial \mathcal{L}(\mathcal{D})}{\partial W}$ are artificial for simplicity.

$$\begin{aligned} W = \begin{bmatrix} 1 & 2 \\ 4 & 3 \end{bmatrix}, \left(\frac{\partial \mathcal{L}(\mathcal{D})}{\partial W}\right) &= \begin{bmatrix} 1 & 2 \\ 3 & 5 \end{bmatrix} \\ \min \|W' - W\|_2^2 \Rightarrow W' &= \begin{bmatrix} 2 & 3 \\ 3 & 2 \end{bmatrix} \\ \min \Omega_i \Rightarrow W' = \mathbb{E}\begin{bmatrix} 1*1^2 & 2*2^2 \\ 4*3^2 & 3*5^2 \end{bmatrix}_{diag} &= \begin{bmatrix} \frac{1*1^2+3*5^2}{1^2+5^2} & \frac{2*2^2+4*3^2}{2^2+3^2} \\ \frac{2*2^2+4*3^2}{2^2+3^2} & \frac{1*1^2+3*5^2}{1^2+5^2} \end{bmatrix} = \begin{bmatrix} 2.92 & 3.38 \\ 3.38 & 2.92 \end{bmatrix} \end{aligned} \tag{11}$$

F Inverted Residual Fusion Algorithm

The key idea of the inverted residual fusion is to compute consecutive linear layers at once with one round communication. The algorithm is described in Algorithm 1 where $\langle\cdot\rangle^C, \langle\cdot\rangle^S$ are the secret shares held by the client and the server. $\boxplus, \boxminus, \boxtimes$ represent homomorphic addition, subtraction, and multiplication, respectively.

Algorithm 1: Inverted Residual Fusion Algorithm

Input: Client holds $\langle \boldsymbol{X}_1 \rangle^C$, and Server holds $\langle \boldsymbol{X}_1 \rangle^S$, $\text{Enc}(\boldsymbol{X}_{res})$, $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$.
Output: Client and Server get $\langle \boldsymbol{Y}_2 \rangle^C$, $\langle \boldsymbol{Y}_2 \rangle^S$, respectively, where
$\boldsymbol{Y}_2 = \text{ConvBN}(\boldsymbol{W}_2, \boldsymbol{X}_{res} + \text{ConvBN}(\boldsymbol{W}_1, \boldsymbol{X}_1))$.

Client encodes and encrypts $\langle \boldsymbol{X}_1 \rangle^C$ as $\text{Enc}(\langle \boldsymbol{X}_1 \rangle^C)$ and sends it to Server.
Server computes $\text{Enc}(\boldsymbol{Y}_1) = \boldsymbol{W}_1 \boxtimes [\text{Enc}(\langle \boldsymbol{X}_1 \rangle^C) \boxplus \langle \boldsymbol{X}_1 \rangle^S]$.
Server computes $\text{Enc}(\boldsymbol{X}_{res} + \boldsymbol{Y}_1) = \text{Enc}(\boldsymbol{X}_{res}) \boxplus \text{Enc}(\boldsymbol{Y}_1)$.
Server computes $\text{Enc}(\boldsymbol{Y}_2) = \boldsymbol{W}_2 \boxtimes \text{Enc}(\boldsymbol{X}_{res} + \boldsymbol{Y}_1)$.
Server samples random noise $\boldsymbol{R}$ which has the same shape as $\boldsymbol{Y}_2$. Server then computes $\text{Enc}(\boldsymbol{Y}_2 - \boldsymbol{R}) = \text{Enc}(\boldsymbol{Y}_2) \boxminus \boldsymbol{R}$.
Server sends $\text{Enc}(\boldsymbol{Y}_2 - \boldsymbol{R})$ to Client and sets $\langle \boldsymbol{Y}_2 \rangle^S = \boldsymbol{R}$.
Client decrypts $\text{Enc}(\boldsymbol{Y}_2 - \boldsymbol{R})$ to get $\langle \boldsymbol{Y}_2 \rangle^C = \boldsymbol{Y}_2 - \boldsymbol{R}$.

G Details of Experimental Setup

G.1 Network Architectures

We evaluate PrivCirNet on MobileNetV2 [28], ResNet-18 [55], and ViT [58]. The detailed architectures across different datasets are in Table 8. It should be noted that for ViT, we use ViT-lite architectures from [58].

G.2 Training Details

All baseline methods and PrivCirNet are trained using identical hyper-parameters, including data augmentation, number of epochs, and others. These hyper-parameters are detailed in the 'configs' folder within our codebase. We also leverage self knowledge distillation to guide the training of the circulant networks and the pruned networks.

G.3 Computational Resources in Experiments

For CIFAR and Tiny ImageNet datasets, we train all models on a single NVIDIA RTX4090 GPU and a single NVIDIA A6000 GPU. For ImageNet, we train all models on 8 NVIDIA A100 GPUs. The epochs are 300 and the total training time is around 1 day for CIFAR and Tiny ImageNet as well as ImageNet datasets.

H Additional Experimental Results

H.1 Latency breakdown of PrivCirNet

In Figure 16, we present the latency breakdown of PrivCirNet (b8) applied to MobileNetV2 and ViT on CIFAR-10. It is observed that PrivCirNetsignificantly reduces the latency associated with HE rotations and multiplications, shifting the bottleneck to nonlinear layers. Furthermore, in ViT, the

Table 8: PrivCirNet evaluation benchmarks.

Model	Layers	# Params (M)	MACs (G)	Dataset
MobileNetV2	52 CONV, 1 FC, 1 AP, 35 ReLU	2.24	0.093	CIFAR/Tiny ImageNet
MobileNetV2	52 CONV, 1 FC, 1 AP, 35 ReLU	3.5	0.32	ImageNet
ResNet-18	52 CONV, 1 FC, 1 AP, 35 ReLU	11.17	0.558	CIFAR/Tiny ImageNet
ViT	Hidden Dim=256, Number of blocks=7	3.72	0.24	CIFAR
ViT	Hidden Dim=192, Number of blocks=9	2.77	0.69	Tiny ImageNet

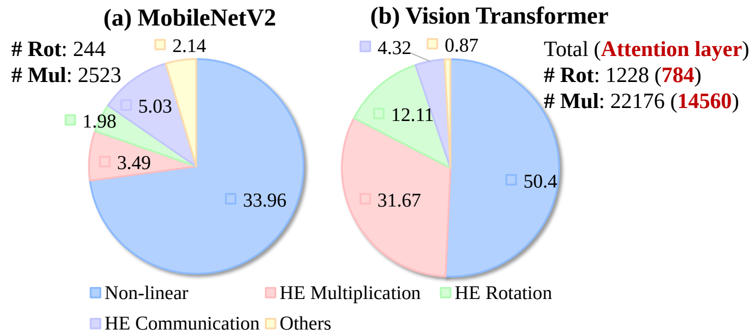

Figure 16: Latency (s) breakdown of PrivCirNet (b8) on MobileNetV2 and ViT on CIFAR-10.

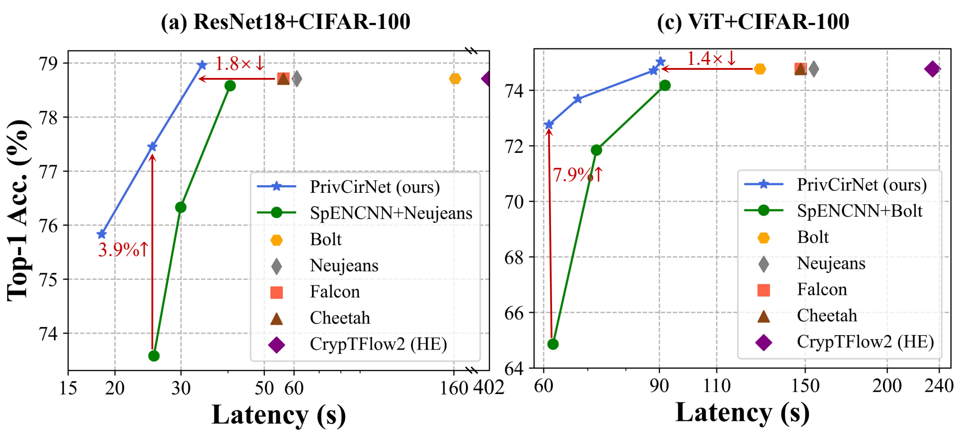

Figure 17: Comparison with SpENCNN and other prior-art protocols on ResNet-18 and ViT on CIFAR-100.

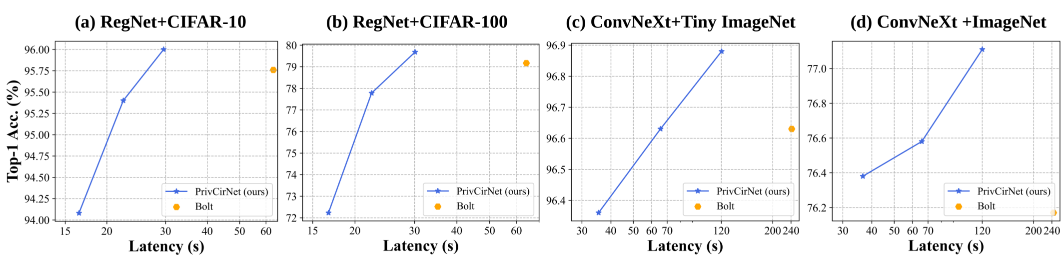

Figure 18: Comparison with Bolt on RegNet and ConvNeXt.

self-attention layers account for a large proportion of the total HE operations. Since these layers lack weight matrices, they cannot benefit from block circulant transformations.

H.2 Results on more networks

Results of ResNet-18 and ViT on CIFAR-100 In Figure 17, we show the results of ResNet-18 and ViT on CIFAR-100. We compare PrivCirNet with SOTA HE-based DNN inference frameworks and HE-friendly structured pruning method SpENCNN. We find that: **1)** PrivCirNet achieves $1.8\times$ latency reduction on ResNet-18 and $1.4\times$ latency reduction on ViT compared with SOTA frameworks Cheetah and Bolt with iso-accuracy. **2)** Compared with SpENCNN, PrivCirNet achieves 3.9% and 7.9% higher accuracy on ResNet-18 and ViT with lower latency, respectively. **3)** Bolt performs worse than Cheetah in ResNet-18 because Bolt needs to transform convolution into GEMM which increases the hidden dimension by $9\times$ in 3×3 convolutions. By contrast, PrivCirNet support both convolution and GEMM efficiently.

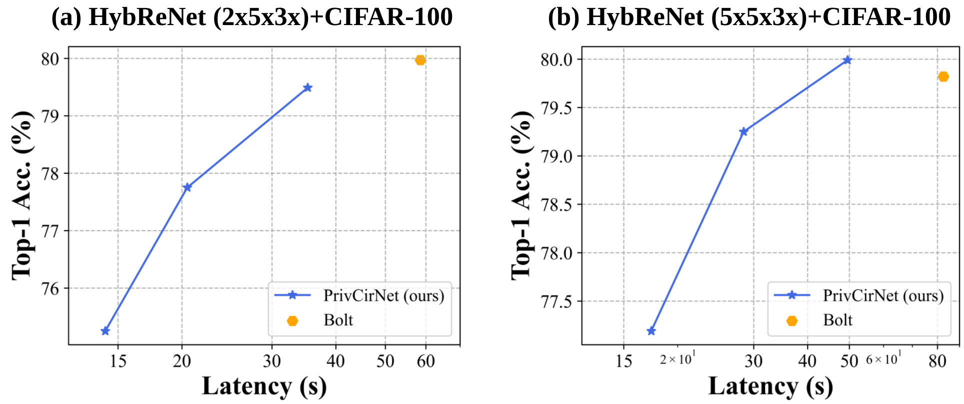

Figure 19: Result of applying PrivCirNeton DeepReshape.

Table 9: Accuracy and latency results when combining PrivCirNet and DeepReshape (ReLU reduction).

Method	Top-1 Acc.	Linear layers' latency (s)	Nonlinear layers' latency (s)
HybReNet (5x5x3x) + PrivCirNet (b2)	79.99	49.52	9.4
+DeepReshape (-53% ReLU)	79.67	49.52	4.4
HybReNet (2x5x3x) + PrivCirNet (b2)	79.49	35.25	9.0
+DeepReshape (-50% ReLU)	79.03	35.25	4.5
+DeepReshape (-72% ReLU)	77.91	35.25	2.5

Results of RegNet and ConvNeXt In Figure 18, we show the results of RegNet [59] and ConvNeXt [60] on CIFAR. We compare PrivCirNet with SOTA HE-based DNN inference framework Bolt.

Comparion with DeepReshape [61] DeepReshape optimizes ReLUs and FLOPs by designing a series of more FLOPs-efficient networks, dubbed HybReNets while pruning the ReLU layers. DeepReshape achieves a better latency-ReLU trade-off than SENet [32], SNL [30], etc. DeepReshape and PrivCirNet are orthogonal and can be applied together to further reduce the inference latency.

In Figure 19 and Table 9, we show the application of PrivCirNet to HybReNets on CIFAR-100, which also yields promising results. We apply the ReLU pruning method proposed in DeepReshape to reduce the latency of nonlinear layers.

From the results, we can see that PrivCirNet is effective when combined with DeepReshape, achieving significant latency reduction in both linear and nonlinear layers.

Discussion on the impact of selecting different baseline networks The varying accuracy degradation observed across different baseline networks (MobileNetV2, ResNet, HybReNet, RegNet, ConvNeXt) can be partly attributed to the differing proportions of parameters occupied by standard convolutional layers. For instance, in ConvNeXt, 98% of the parameters are derived from standard convolution, with less than 2% from depth-wise/group-wise convolution, providing significant compression potential using PrivCirNet. In contrast, standard convolution parameters account for only 64% and 78% of RegNet and MobileNetV2, respectively. As a result, RegNet and MobileNetV2 exhibit larger accuracy degradation at higher compression rates.

I Proofs

I.1 Proof of Theorem 1

For a given input matrix X and a circulant matrix W, we have

$$\begin{aligned} &W \in \mathbb{Z}^{b\times b}, W[i,j] = W[0,(b-i+j) \mod b], \forall i \in [b], \forall j \in [b] \\ &X \in \mathbb{Z}^{b\times d_1}, X[i,j], \forall i \in [b], \forall j \in [d_1] \end{aligned} \tag{12}$$

The matrix multiplication result Y is

$$Y = WX \in \mathbb{Z}^{b\times d_1}, Y[i,j] = \sum_{k=0}^{b-1} W[i,k]X[k,j] = \sum_{k=0}^{b-1} W[0,(b-i+k) \mod b]X[k,j] \tag{13}$$

The polynomials $\hat{x} = \pi_{\mathrm{X}}(X), \hat{w} = \pi_{\mathrm{W}}(W)$ after CirEncode are

$$\begin{aligned} &\hat{x} \in \mathbb{A}_n, \hat{x}[id_1 + j] = X[i,j], \forall i \in [b], \forall j \in [d_1] \\ &\hat{w} \in \mathbb{A}_n, \hat{w}[id_1] = W[i,0] = W[0,(b-i) \mod b], \forall i \in [b], \end{aligned} \tag{14}$$

The other slots of $\hat{w}$ are set to 0. The polynomial multiplication result $\hat{y} = \hat{w} \times \hat{x}$ directly gives the matrix multiplication result Y as

$$\begin{aligned} \hat{y} &= \hat{w} \times \hat{x} \in \mathbb{A}_n \\ \hat{y}[id_1 + j] &= \sum_{k=0}^{b-1} \hat{w}[(i-k)d_1]\hat{x}[kd_1 + j] \\ &= \sum_{k=0}^{b-1} W[0,(b-i+k) \mod b]X[k,j] \\ &= \sum_{k=0}^{b-1} W[i,k]X[k,j] = Y[i,j] \end{aligned} \tag{15}$$

Besides, we extend the definition of $\hat{w}[i] = \hat{w}[bd_1 + i], \forall i < 0$.

Explanation of CirEncode Modulo $x^n - 1$. CirEncode performs Discrete Fourier Transform (DFT) modulo $x^n - 1$ on the plaintext. After the DFT, it applies SIMD encoding to enable element-wise multiplication. The correctness is demonstrated by the equation $\mathrm{DFT}(\hat{w}) \odot \mathrm{DFT}(\hat{x}) = \mathrm{DFT}(\hat{w} \times \hat{x} \mod x^n - 1)$

I.2 Proof of Theorem 2

Given M circulant weight matrices $W_0, \ldots, W_{M-1} \in \mathbb{Z}^{b\times b}$ and input matrices $X_0, \ldots, X_{M-1} \in \mathbb{Z}^{b\times d_1}$, define the polynomials $\hat{w}_m = \pi_{\mathrm{W}}(W_m)$ and $\hat{x}_m = \pi_{\mathrm{X}}(X_m)$ with $m \in [M]$ following the coefficient packing in Theorem 1. We have:

$$\begin{aligned} &\langle \mathrm{DFT}(\hat{w}_0) | \ldots | \mathrm{DFT}(\hat{w}_{M-1}) \rangle_{\mathrm{SIMD}} \times \langle \mathrm{DFT}(\hat{x}_0) | \ldots | \mathrm{DFT}(\hat{x}_{M-1}) \rangle_{\mathrm{SIMD}} \\ &= \langle \mathrm{DFT}(\hat{w}_0) \odot \mathrm{DFT}(\hat{x}_0) | \ldots | \mathrm{DFT}(\hat{w}_{M-1}) \odot \mathrm{DFT}(\hat{x}_{M-1}) \rangle_{\mathrm{SIMD}} \\ &= \langle \mathrm{DFT}(\hat{w}_0 \times \hat{x}_0) | \ldots | \mathrm{DFT}(\hat{w}_{M-1} \times \hat{x}_{M-1}) \rangle_{\mathrm{Coeff}} \\ &= \langle \mathrm{DFT}(\hat{y}_0) | \ldots | \mathrm{DFT}(\hat{y}_{M-1}) \rangle_{\mathrm{Coeff}} \end{aligned} \tag{16}$$

Then we can perform inverse DFT and directly extract elements following Theorem 1 from $\hat{y}_m$ to get $Y_m, \forall m \in [M]$. The second and the third lines of Equation 16 follow directly from Lemma 1 while the last line is derived from Theorem 1. Through Equation 16, we simultaneously evaluate M circulant GEMMs with CirEncode.

I.3 Proof of Theorem 3

For a given input X and a circulant weight W of a convolution, we have

$$\begin{aligned} W \in \mathbb{Z}^{b\times b\times R\times R}, W[i,j,:,:] &= W[0,(b-i+j) \mod b,:,:] \\ &= W[(b-j+i) \mod b, 0,:,:], \forall i \in [b], \forall j \in [b] \\ X \in \mathbb{Z}^{b\times H\times W}, X[i,j,k]&, \forall i \in [b], \forall j \in [H], \forall k \in [W] \end{aligned} \tag{17}$$

The convolution result Y is

$$\begin{aligned} Y &= W \circledast X \in \mathbb{Z}^{b\times(H-R+1)\times(W-R+1)} \\ Y[i,j,k] &= \sum_{l=0}^{b-1}\sum_{m=0}^{R-1}\sum_{h=0}^{R-1} W[i,l,m,h]X[l,j+m,k+h] \end{aligned} \tag{18}$$

The polynomials $\hat{x} = \pi'_{\mathrm{X}}(X), \hat{w} = \pi'_{\mathrm{W}}(W)$ after CirEncode are

$$\begin{aligned} &\hat{x} \in \mathbb{A}_n, \hat{x}[iHW + jW + k] = X[i,j,k] \\ &\hat{w} \in \mathbb{A}_n, \hat{w}[iHW + (W+1)(R-1) - jW - k] = W[i,0,j,k] \end{aligned} \tag{19}$$

The other slots of $\hat{w}$ are set to 0. The polynomial multiplication result $\hat{y} = \hat{w} \times \hat{x}$ directly gives the convolution result Y as

$$\begin{aligned} \hat{y} &= \hat{w} \times \hat{x} \in \mathbb{A}_n \\ \hat{y}[iHW + (W+1)(R-1) + jW + k] &= \sum_{l=0}^{b-1}\sum_{m=0}^{R-1}\sum_{h=0}^{R-1} \\ (\hat{w}[(i-l)HW + (W+1)(R-1) - mW - h]&\hat{x}[lHW + (j+m)W + (k+h)]) \\ &= \sum_{l=0}^{b-1}\sum_{m=0}^{R-1}\sum_{h=0}^{R-1} W[i,l,m,h]X[l,j+m,k+h] \\ &= Y[i,j,k] \end{aligned} \tag{20}$$

Besides, we extend the definition of $\hat{w}[(i-l)HW + \ldots] = \hat{w}[(b+i-l)HW + \ldots], \forall i < l$.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: /

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Section 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Proofs of all theoretical results are available in the Appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Our code and checkpoints are available on Git Hub.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Our code and checkpoints are available on Git Hub.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All training details are available in our code on Git Hub.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We conduct experiments on multiple models and datasets, which require significant computational resources. In addition, our code has been released, making it easy to reproduce the results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Information about our computing resources is available in the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: /

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: /

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: /

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: /

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: /

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: /

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: /

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.