PrivQuant: Communication-Efficient Private Inference with Quantized Network/Protocol Co-Optimization

Tianshi Xu[1,2], Shuzhang Zhong[2,1], Wenxuan Zeng[5,2], Runsheng Wang[1,3,4], Meng Li[2,1,4*]
*meng.li@pku.edu.cn
[1] School of Integrated Circuits & [2] Institute for Artificial Intelligence, Peking University, Beijing, China
[3] Institute of Electronic Design Automation, Peking University, wuxi, China
[4] Beijing Advanced Innovation Center for Integrated Circuits, Beijing, China
[5] School of Software and Microelectronics, Peking University, Beijing, China

ABSTRACT

Private deep neural network (DNN) inference based on secure two-party computation (2PC) enables secure privacy protection for both the server and the client. However, existing secure 2PC frameworks suffer from a high inference latency due to enormous communication. As the communication of both linear and non-linear DNN layers reduces with the bit widths of weight and activation, in this paper, we propose PrivQuant, a framework that jointly optimizes the 2PC-based quantized inference protocols and the network quantization algorithm, enabling communication-efficient private inference. PrivQuant proposes DNN architecture-aware optimizations for the 2PC protocols for communication-intensive quantized operators and conducts graph-level operator fusion for communication reduction. Moreover, PrivQuant also develops a communication-aware mixed precision quantization algorithm to improve the inference efficiency while maintaining high accuracy. The network/protocol co-optimization enables PrivQuant to outperform prior-art 2PC frameworks. With extensive experiments, we demonstrate PrivQuant reduces communication by 11×, 2.5 × and 2.8×, which results in 8.7×, 1.8 × and 2.4× latency reduction compared with SiRNN, COINN, and CoPriv, respectively.

1 INTRODUCTION

With deep learning being applied to increasingly sensitive data and tasks, privacy has emerged as one of the major concerns in the deployment of deep neural networks (DNNs). To enable DNN inference on private data, secure two-party computation (2PC) is proposed as a promising solution and has attracted increasing attention in recent years [1–4].

Secure 2PC helps solve the following dilemma: the server owns a private DNN model and the client owns private data. The server is willing to provide the model as a service but is reluctant to disclose it. Simultaneously, the client wants to apply the model to private data without revealing the data as well. Secure 2PC frameworks can fulfill both parties' requirements: the two parties can learn the inference results but nothing else beyond what can be derived from the results [5, 6].

This work was supported in part by the NSFC (62125401), the 111 Project (B18001), and Ant Group.

ICCAD '24, October 27–31, 2024, New York, NY, USA

ACM ISBN 979-8-4007-1077-3/24/10
https://doi.org/10.1145/3676536.3676661

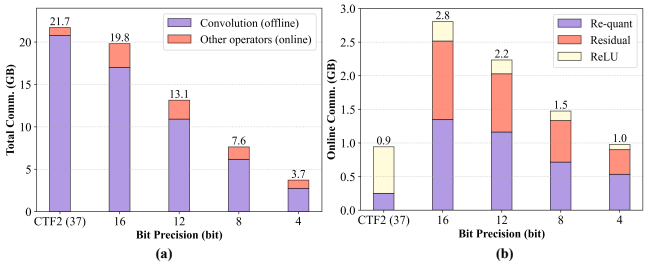

Figure 1: Profile the ResNet50 building block with representative 2PC protocols, i.e., CrypTFlow2 (first column) and SiRNN (other columns): the scaling and breakdown of (a) total communication and (b) online communication with different bit-widths of weight and activation.

However, the privacy protection offered by secure 2PC frameworks comes at the expense of high communication complexity, which stems from the extensive interaction required between the server and the client [6]. This leads to orders of magnitude latency gap compared to the conventional inference on plaintext. Recently, 2PC protocols for quantized inference are proposed together with the application of low bit-width networks in the 2PC framework [1–4]. As shown in Figure 1, in the prior-art work SiRNN [4], both total and online communication of the 2PC inference reduces significantly as the operands' bit-widths decrease, demonstrating promising efficiency improvements [2, 4].

Though promising, existing 2PC frameworks based on quantized DNNs still face the following limitations: 1) **complex protocols unaware of quantized DNN architectures**: 2PC-based quantized DNN inference requires complex protocols to align the bit-widths and scales for different operands, including truncation, extension, re-quantization, etc [4]. These 2PC protocols are designed without considering the DNN architectures, e.g., tensor shapes, bit-widths, etc., and suffer from significant communication overhead; 2) **DNN quantization algorithms unaware of 2PC protocols**: existing network quantization algorithms are designed or optimized for plaintext inference, ignoring the characteristics of 2PC-based inference. This may lead to sub-optimal quantization solutions with large accuracy degradation or high communication complexity. As a result, as shown in Figure 1, mixed bit-width inference protocol SiRNN requires almost the same total communication and more online communication compared to the uniformly quantized protocol CrypTFlow2 [3] even with less than half the bits.

To address the aforementioned limitations, we propose PrivQuant, which jointly optimizes the 2PC protocols for quantized DNN inference and the network quantization strategy. Compared with existing works, our contributions can be summarized as follows:

- We propose both operator-level and graph-level optimizations of 2PC protocols. At the operator level, DNN-aware

Table 1: Notations used in the paper.

Notations	Meanings
λ	Security parameter that measures the attack hardness
$\gg$	Shift right
l, s	Bit width, scale of an operand
l_w, l_x, l_{acc}	The bit width of weights, activations and accumulation.
l_{res}, l_{add}	The bit width of residual tensor and residual addition. Usually, $l_{add} = l_{res} + 1$ to avoid addition overflow.
s_w, s_x, s_{acc}	The scale of weights, activations and accumulation.
s_{res}, s_{add}	The scale of residual tensor and residual addition.
$x^{(l)}, \langle x \rangle^{(l)}$	An l-bit integer x and l-bit secret shares

protocol optimizations are proposed for communication-intensive quantized operators, including convolution and residual addition. At the graph level, operator fusion and sign propagation are proposed for further communication reduction.

- We propose a network optimization algorithm that leverages high bit-width residuals and communication-aware mixed bit-width quantization to enable accurate yet efficient 2PC-based quantized inference.
- We demonstrate communication reduction for individual operators and whole networks. PrivQuant reduces communication by $2 \sim 16\times$ compared to prior-art 2PC frameworks, including CrypTFlow2, SiRNN, COINN, and CoPriv, which leads to $1.3 \sim 12\times$ latency reduction.

2 PRELIMINARIES

2.1 Network Quantization for 2PC Inference

Quantization converts floating-point numbers into integers [7]. Specifically, a floating point number x_f can be approximated by an l_x-bit integer x_q and a scale s_x through quantization as x_q/s_x[1], where

$$x_q = \max(-2^{l_x-1}, \min(2^{l_x-1} - 1, \text{round}(s_x x_f))).$$

The multiplication of two floating point numbers x_f and w_f, denoted as y_f, can be approximately computed as $x_q w_q/(s_w s_x)$, which is a quantized number with $l_x + l_w$ bit and $s_w s_x$ scale. Then, y_f usually needs to be re-quantized to y_q with l_y bit and s_y scale as follows:

$$y_q = \max(-2^{b_y-1}, \min(2^{b_y-1} - 1, \text{round}(\frac{s_y}{s_w s_x} w_q x_q))).$$

For the addition of two quantized numbers x_q and y_q, directly computing x_q and y_q leads to incorrect results. Instead, the scales and the bit-widths of x_q and y_q need to be aligned first. Uniform quantization protocol CrypTFlow2 leverages the same bit-widths and scales for the tensors, e.g. 37-bit bit-width and 13-bit scale across all layers while mixed bit-width protocol SiRNN uses different quantization parameters for weight and activation, which introduces large communication overhead.

2.2 Notations

We now briefly introduce the security primitives used in the paper. We also summarize the notations in Table 1.

[1]We consider symmetric quantization without zero shift and force s_x to be power of 2 to reduce communication following [4].

Secret Share (SS). We use 2-out-of-2 secret sharing to keep the input data private throughout the whole inference. For an l-bit value $x \in \mathbb{Z}_{2^l}$, we denote its shares by $\langle x \rangle^{(l)} = (\langle x \rangle_s^{(l)}, \langle x \rangle_c^{(l)})$ such that $x = \langle x \rangle_s^{(l)} + \langle x \rangle_c^{(l)} \mod 2^l$ where the server holds $\langle x \rangle_s^{(l)}$ and the client holds $\langle x \rangle_c^{(l)}$.

Oblivious Transfer (OT). We use 1-out-of-2 OT, denoted by $\binom{2}{1} - \text{OT}_l$, where one party is the sender with 2 l-bit messages x_0, x_1 and the other party is the receiver with an index $j \in \{0, 1\}$. The receiver learns x_j as the output, and the sender learns nothing. The communication cost for a $\binom{2}{1} - \text{OT}_l$ is $O(\lambda + 2l)$ bits.

Underlying Protocols. PrivQuant relies on several underlying protocols from SiRNN [4], briefly introduced in Table 2.

2.3 Related works

Existing secure 2PC-based frameworks mainly leverage two classes of techniques: homomorphic encryption (HE) [9], which is computation intensive, and OT [10] which is communication intensive. In this paper, we focus on OT-based methods instead of HE-based methods as HE usually requires the client to have a high computing capability for encryption and decryption. SecureML [11] is the first OT-based framework for secure 2PC-based DNN inference. It suffers from high communication and takes around 1 hour to finish a simple two-layer network. To improve the 2PC inference efficiency, follow-up works can be roughly categorized into two classes: 1) protocol optimizations at the operator level, e.g., convolutions [12–14], matrix multiplications [15, 16], activation functions [3], and at network level, e.g., hybrid protocols [17, 18]; 2) 2PC-aware network optimization, such as using 2PC friendly activation functions [19] and 2PC-aware DNN architecture optimization [8, 20–26].

Previous works leveraging quantized networks to improve the efficiency of private inference may fall into either class. XONN [1] leverages binary weight and activation to reduce communication cost but suffers from large accuracy degradation. CrypTFlow2 [3] proposes a hybrid protocol that supports OT-based linear layers with uniform bit-widths. SiRNN [4] further proposes 2PC protocols for bit width extension and reduction to allow mixed bit-width inference. COINN [2] simultaneously optimizes both the quantized network as well as the protocols for linear and non-linear layers. However, COINN uses approximations extensively and also has both the least significant bit (LSB) error and the most significant error (MSB) 1-bit error during truncation. In Table 3, we compare PrivQuant with these works qualitatively, and as can be observed, PrivQuant leverages both protocol and network optimization to support efficient and faithful mixed bit-width inference.

2.4 Threat Model and Security of PrivQuant

PrivQuant works in a general private inference scenario, where a server holds a private DNN model and a client owns private data [13, 16]. PrivQuant enables the client to obtain the inference while keeping the model and the client's data private. Besides, the model parameters are privately held by the server in plaintext while the activations are in secret sharing form during the whole inference.

Consistent with previous works [2, 4, 6, 11], PrivQuant adopts an *honest-but-curious* security model [2] in which both parties follow the specification of the protocol but also try to learn more from the protocols than allowed. PrivQuant is built upon underlying OT

[2]Malicious security model is also an important research direction but is not the focus of this paper

Table 2: Base protocols used in PrivQuant.

Protocol	Description	Comm. Complexity
$\langle y\rangle^{(l_2)} = \Pi_{\text{Ext}}^{l_1,l_2}(\langle x\rangle^{(l_1)})$	Bit-width extension. Extending l_1-bit x to l_2-bit y such that $y^{(l_2)} = x^{(l_1)}$	$O(\lambda(l_1+1))$
$\langle y\rangle^{(l_1)} = \Pi_{\text{Trunc}}^{l_1,l_2}(\langle x\rangle^{(l_1)})$	Truncate (right shift) l_1-bit x by l_2-bit such that $y^{(l_1)} = x^{(l_1)} \gg l_2$	$O(\lambda(l_1+3))$
$\langle y\rangle^{(l_1-l_2)} = \Pi_{\text{TR}}^{l_1,l_2}(\langle x\rangle^{(l_1)})$	Truncate (right shift) l_1-bit x by l_2-bit and discard the high l_2-bit such that $y^{(l_1-l_2)} = x^{(l_1)} \gg l_2$	$O(\lambda(l_2+1))$
$\langle y\rangle^{l_2} = \Pi_{\text{Requant}}^{l_1,s_1,l_2,s_2}(\langle x\rangle^{l_1})$	Re-quantize x of l_1-bit and s_1-scale to y of l_2-bit and s_2-scale. It's a combination of $\Pi_{\text{Ext}}, \Pi_{\text{Trunc}}, \Pi_{\text{TR}}$ with detailed description in Algorithm 1.	in Algorithm 1

Table 3: Qualitative comparison with prior-art 2PC frameworks of quantized network.

Framework	Network Optimization	Protocol Optimization		Error free
		Operator Level	Graph Level	
XONN [1]	Binary Quant.	/	/	✓
CrypTFlow2 [3]	Uniform Quant.	ReLU Protocol	/	Accumulation Overflow
SIRNN [4]	Mixed Bit-width Quant.*	Mixed Bit-width Protocol	MSB Opt.	✓
COINN [2]	Layer-wise bit-width Quant.	Factorized Conv.	Protocol Conversion	Accumulation Overflow LSB and MSB error in Π_{Trunc}
CoPriv [8]	Winograd Conv.	Winograd Conv.	/	✓
Ours	Layer-wise Mixed bit-width Quant. High Bit-width Residual	DNN-aware Conv. Simplified Residual	MSB Opt. Protocol Fusion	✓

*Mixed Bit-width Quant. means the bit-width of output is dynamically determined by the input, the weight and the dimension of the layer.

primitives, which are proven to be secure in the *honest-but-curious* adversary model in [27] and [28], respectively. Utilizing quantized neural networks does not affect OT primitives in any way.

3 MOTIVATION

In Figure 1, we profile the communication cost of a ResNet block with both CrypTFlow2 and SiRNN protocol [3, 4]. We observe the total communication of SiRNN is dominated by convolution (offline) while the online communication bottleneck comes from the residual addition and re-quantization. Notably, the communication of all these operators decreases significantly with the reduction of weight and activation bit-width. Therefore, ideally, SiRNN should have achieved substantially higher efficiency compared to the uniform quantized protocol CrypTFlow2.

However, from Figure 1, we find that the communication of the 16-bit SiRNN protocol is comparable to or even higher than CrypTFlow2 with 37-bit. We further compare the detailed protocols for one convolution with residual connection in CrypTFlow2 and SiRNN in Figure 2. This identifies two critical issues with SiRNN:

- *Complex protocols unaware of quantized network architecture*: to enable mixed bit-width inference without introducing errors, SiRNN requires complex protocols to align the bit-widths and scales of operands, including bit-width extension, truncation, and re-quantization. These protocols are agnostic to the model architecture and introduce extremely high communication overhead, as shown in Table 2 for communication complexity.
- *Network quantization unaware of protocols*: SiRNN uniformly applies the same bit-width to weights and activations. Such a strategy ignores the cost of the protocol under different bit-width settings and results in sub-optimal communication efficiency or network accuracy.

Based on the observations above, we propose PrivQuant which features a network/protocol co-optimization. As shown in Figure 3, we propose network-aware protocol optimizations at both the operator level (Section 4.1) and the graph level (Section 4.2), directly

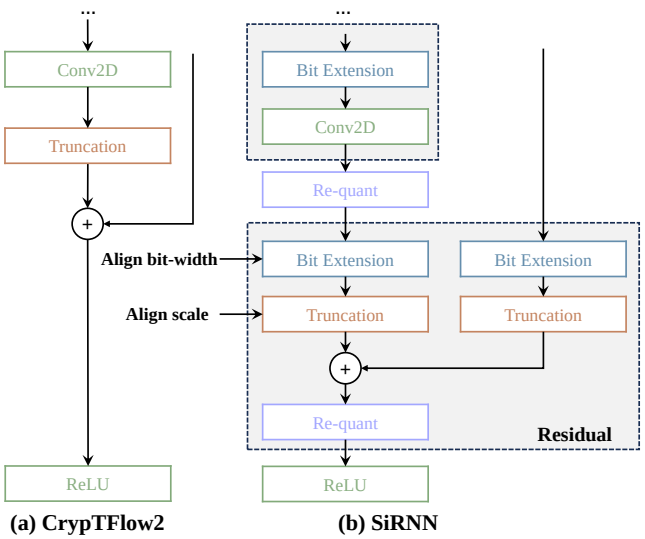

Figure 2: Detailed protocols for one convolution with residual connection in (a) CrypTFlow2 and (b) SiRNN. The bit extension, truncation, and re-quantization are required in SiRNN to align the bit-widths and scales of quantized operands.

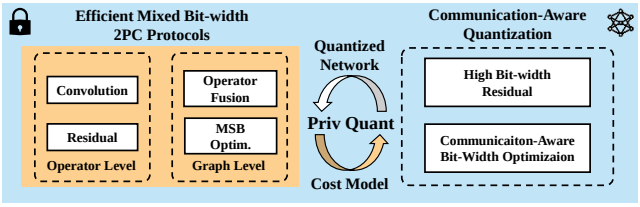

Figure 3: Overview of PrivQuant.

targeting the communication-intensive operations, i.e., convolution and residual addition. For network optimization, we leverage high bit-width residual (Section 5.1) and communication-aware mixed bit-width quantization (Section 5.2), enabling efficient-yet-accurate 2PC inference.

4 EFFICIENT MIXED BIT-WIDTH 2PC PROTOCOLS

We now describe the protocol optimization of PrivQuant at both the operator level and the graph level.

4.1 Operator Level Protocol Optimization

4.1.1 DNN Architecture-Aware Convolution Protocol. **Baseline Protocol in SiRNN.** In SiRNN, a convolution is first converted to

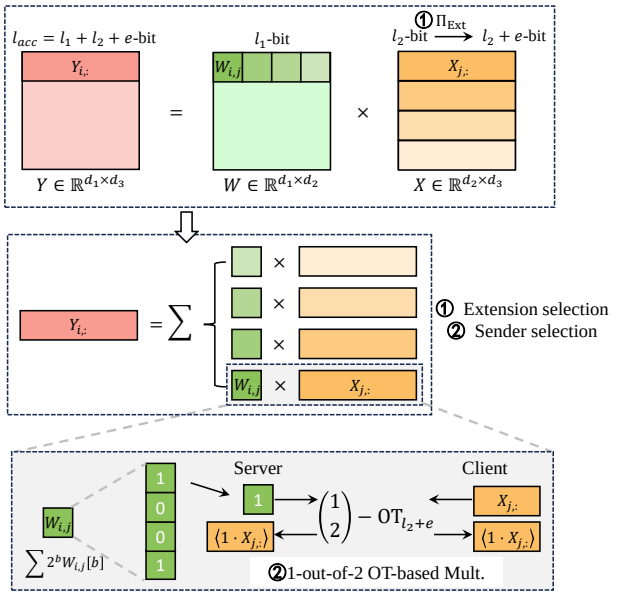

Figure 4: An illustration of OT-based matrix multiplication protocol which extends X and chooses the client to be the sender. We omit $\langle \cdot \rangle$ for simplicity.

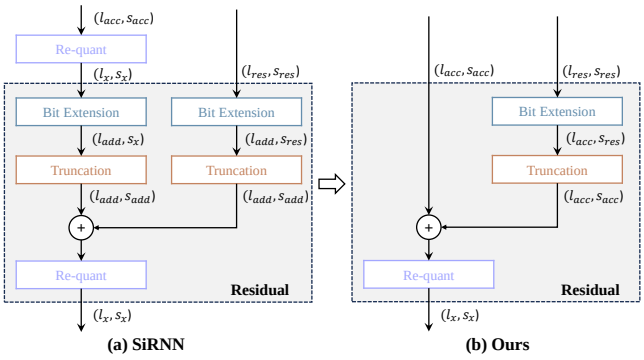

Figure 5: (a) The baseline protocol for the residual addition in SiRNN; and (b) our proposed simplified protocol. The $l\_, s\_$ means the bit-width and scale of the activations.

matrix multiplication with `im2col`. Hence, we focus on optimizing the matrix multiplication protocol where we want to compute $\langle Y \rangle = W\langle X \rangle = W\langle X \rangle_s + W\langle X \rangle_c$. The $W\langle X \rangle_s$ can be computed locally by the server. To compute $W\langle X \rangle_c$, both parties invoke an OT-based protocol as shown in Figure 4, where $W \in \mathbb{R}^{d_1 \times d_2}$ with l_1-bit, $\langle X \rangle_c \in \mathbb{R}^{d_2 \times d_3}$ with l_2-bit. The i-th row of Y, denoted as $Y_{i,:}$ is computed by $\sum_{i=1}^{d_2} W_{i,j} X_{j,:}$. To prevent numerical overflow during the multiplication and accumulation, the bit-width of $Y_{i,:}$ should be $l_1 + l_2 + e$ where $e = \log_2(d_2)$. Therefore, there exists a bit-width extension step before the multiplication where we can either extend W or X by e bits. To compute one $W_{i,j} X_{j,:}$, as shown in Figure 4, the server split $W_{i,j}$ into bits. For each bit $W_{i,j}[b], b \in [1, 2, \ldots, l_1]$, both parties invoke a $\binom{2}{1} - \text{OT}_{l_2+e}$ and obtain $W_{i,j}[b] X_{j,:}$. Finally, both parties can get Y by computing each $Y_{i,:}$:

$$Y_{i,:} = \sum_{i=1}^{d_2} \sum_{b=1}^{l_1} W_{i,j}[b] X_{j,:} \tag{1}$$

The total number of OTs invoked is $d_1 d_2 l_1$ and the communication of each OT is $O(\lambda + 2(l_2 + e)d_3)$.

Our Protocol. We first split the convolution protocol into three steps:

- Bit-width extension. We can either invoke $\Pi_{\text{Ext}}^{l_1, l_1+e}(W)$ or $\Pi_{\text{Ext}}^{l_2, l_2+e}(\langle X \rangle)$. When we extend W, the communication cost is 0 since W is in plaintext on the server side; when we extend $\langle X \rangle$, the communication cost is $O(d_2 d_3 \lambda (l_2 + 1))$.
- OT-based Multiplication. In this step, we find two execution ways with different communication costs. The first way is Equation 1 which is SiRNN's protocol in Figure 4. The second way is $Y_{:,j} = \sum_{i=1}^{d_2} \sum_{b=1}^{l_2+e} W_{:,i} X_{i,j}[b]$ where the client splits $X_{i,j}$ into bits and the server is the sender with input $W_{:,i}$. Suppose we extend X in the first step, the communication cost of the first way is $O(d_1 d_2 l_1 (\lambda + 2(l_2 + e)d_3))$ and the second way is $O(d_2 d_3 (l_2 + e)(\lambda + 2 l_1 d_1))$.
- Wrap & MUX. This step prevents the MSB error in the output. The communication cost is related to both bit-widths and dimensions of W and X. Since we do not optimize this step, we omit the detailed protocols and refer interested readers to SiRNN Section III .E [4].

Based on the analysis above, the communication varies when we choose to extend X or W and also when we choose a different party to be the OT sender, resulting in four possible communication costs. We analyze in detail the communication cost of the four choices in Table 4. Furthermore, we observe that the communication of different choices can be completely determined given the DNN architecture before the inference. Hence, in PrivQuant, we propose a DNN architecture-aware protocol for the matrix multiplication, which calculates the communication of all options for each convolution and selects the extension and sender adaptively to minimize the communication. On the contrary, SiRNN is agnostic to the DNN architecture and always extends X and selects the client as the sender. This strategy is sub-optimal because we find in experiments that it is not always the lowest communication cost choice.

4.1.2 Simplified Residual Protocol. As shown in Figure 1, residual addition takes up a large portion of online communication due to the complex alignment of both bit-widths and scales (Figure 2 (b)). The aligned operands are then added and quantized back to l_x-bit as shown in Figure 5 (a). The baseline protocol is quite expensive because both re-quantization and extension require multiple rounds of communication. Therefore, we propose a simplified residual protocol in Figure 5 (b) which aligns the bit-width and scale of the residual directly to the convolution output for addition. Through simplification, we get rid of all the operators for the convolution output's quantization while keeping the high bit-width residual addition since l_{acc} is usually quite large. As we will demonstrate in Section 6.2, this approach significantly reduces communication cost.

4.2 Graph Level Protocol Optimization

At the graph level, we propose both activation sign propagation and fused protocol for quantization to reduce communication. Figure 6 shows an example of a residual block.

4.2.1 Activation Sign Propagation. Previous work has proposed the most significant bit (MSB) optimization. Specifically, when the MSB of the operands is known, several protocols can be optimized

Table 4: Impact of expansion and sender on the communication of the matrix multiplication protocol. And the communication comparison of SiRNN and PrivQuant.

Method	Expansion	Sender	Communication (bits)		
			Extension	OT-based Mult.	Wrap & MUX
①	W	Server	0	$O(d_2d_3l_2(\lambda+2(l_1+e)d_1))$	$O(d_2d_3(\lambda+14)l_2+d_1d_2(\lambda+2(l_1+e)d_3))$
②	W	Client	0	$O(d_1d_2(l_1+e)(\lambda+2l_2d_3))$	$O(d_2d_3(\lambda+14)l_2+d_1d_2(\lambda+(l_1+e)d_3))$
③	X	Server	$O(d_2d_3\lambda(l_2+1))$	$O(d_2d_3(l_2+e)(\lambda+2l_1d_1))$	$O(d_2d_3(\lambda+14)(l_2+e)+d_1d_2(\lambda+l_1d_3))$
④	X	Client	$O(d_2d_3\lambda(l_2+1))$	$O(d_1d_2l_1(\lambda+2(l_1+e)d_3))$	$O(d_2d_3(\lambda+14)(l_2+e)+d_1d_2(\lambda+l_1d_3))$
SiRNN [4]: ④	X	Client	④		
PrivQuant : min(①,②,③,④)	Adaptive	Adaptive	min(①,②,③,④)		

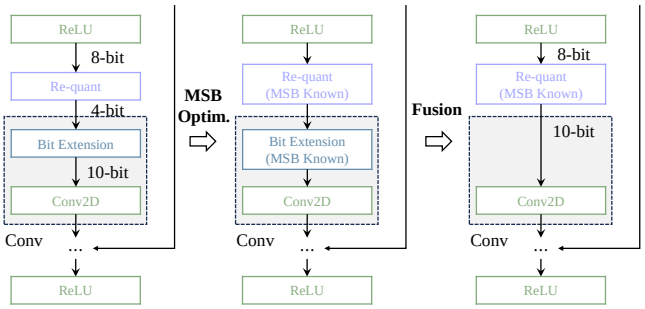

Figure 6: The graph level protocol optimization in an example residual block. (a) The baseline, (b) uses activation sign propagation, and (c) uses fused protocol for quantization.

including Π_{Ext}, Π_{Trunc}, etc., with detailed communication cost in Table 5. Since the output of ReLU is non-negative, we utilize this optimization thoroughly by searching the computation graph and making use of MSB optimization in every non-negative operand.

Table 5: Communication of several protocols with/without MSB optimization.

Protocol	Comm. w/o MSB Optimization (bits)	Comm. w/ MSB Optimization (bits)
$\Pi^{l_1,l_2}_{\text{Ext}}$	$O(\lambda(l_1+1))$	$O(2\lambda-l_1+l_2)$
$\Pi^{l_1,l_2}_{\text{Trunc}}$	$O(\lambda(l_1+3))$	$O(3\lambda+l_1+l_2)$
$\Pi^{l_1,l_2}_{\text{TR}}$	$O(\lambda(l_2+1))$	$O(\lambda+2)$

4.2.2 Fused Protocol for Quantization. $\Pi^{l_1,l_2}_{\text{Trunc}}$ is widely used in the quantized inference to avoid overflow. We observe opportunities to fuse neighboring truncation and extension protocols as well as re-quantization protocols at the graph level to reduce communication. First, we introduce the following propositions for the protocol fusion.

Proposition 4.1. *For a given* $\langle x\rangle^{(l_1)}$, $\Pi^{l_1,l_2}_{\text{Trunc}}(\langle x\rangle^{(l_1)})$ *can be decomposed into* $\Pi^{l_1,l_2}_{\text{TR}}$ *followed by* $\Pi^{l_1-l_2,l_1}_{\text{Ext}}$ *as*

$$\Pi^{l_1,l_2}_{\text{Trunc}}(\langle x\rangle^{(l_1)}) = \Pi^{l_1-l_2,l_1}_{\text{Ext}}(\Pi^{l_1,l_2}_{\text{TR}}(\langle x\rangle^{(l_1)}))$$

The decomposition reduce the communication from $\mathrm{O}(\lambda(l_1+3))$ *to* $\mathrm{O}(\lambda(l_1+2))$.

Proposition 4.2. *Two consecutive extension protocols can be fused into one as*

$$\Pi^{l_2,l_3}_{\text{Ext}}(\Pi^{l_1,l_2}_{\text{Ext}}(\langle x\rangle^{(l_1)})) = \Pi^{l_1,l_3}_{\text{Ext}}(\langle x\rangle^{(l_1)})$$

Extension fusion reduces communication from $\mathrm{O}(\lambda(l_1+l_2+2))$ *to* $\mathrm{O}(\lambda(l_1+1))$.

Proposition 4.3. *For a given* $\langle x\rangle^{(l_1)}$, *the consecutive truncation and extension protocol can be fused as*

$$\begin{aligned}\Pi^{l_1,l_3}_{\text{Ext}}(\Pi^{l_1,l_2}_{\text{Trunc}}(\cdot)) &= \Pi^{l_1,l_3}_{\text{Ext}}(\Pi^{l_1-l_2,l_1}_{\text{Ext}}(\Pi^{l_1,l_2}_{\text{TR}}(\cdot)))\\ &= \Pi^{l_1-l_2,l_3}_{\text{Ext}}(\Pi^{l_1,l_2}_{\text{TR}}(\cdot))\end{aligned}$$

Combining Proposition 4.1 and 4.2, this fusion reduces communication from $\mathrm{O}(\lambda(2l_1+4))$ *to* $\mathrm{O}(\lambda(l_1+2))$.

Proposition 4.3 enables us to fuse the re-quantization and extension protocols further. We first describe the proposed re-quantization protocol in Algorithm 1. The key idea of fusion is when Π_{Requant} ends up with Π_{Trunc} or Π_{Ext}, we can further fuse them.

Algorithm 1 Re-quantization, $\Pi^{l_1,s_1,l_2,s_2}_{\text{Requant}}(\langle x\rangle^{(l_1)})$

Input: P_0 & P_1 hold $\langle x\rangle^{(l_1)}$ with scale s_1
Output: P_0 & P_1 get $\langle y\rangle^{(l_2)}$ with scale s_2

if $l_1 \geq l_2$ **then**
  if $s_1 \leq s_2$ **then**
    $\forall b \in \{0,1\}, P_b$ sets $\langle t\rangle^{(l_1)} = \langle x\rangle^{(l_1)} \ll (s_2-s_1)$
    $\forall b \in \{0,1\}, P_b$ sets $\langle y\rangle^{(l_2)} = \langle t\rangle^{(l_1)}$
  else if $(l_1-l_2) \geq (s_1-s_2)$ **then**
    P_0 and P_1 invoke $\Pi^{l_1,s_1-s_2}_{\text{TR}}\left(\langle x\rangle^{(l_1)}\right)$ and learn $\langle t\rangle^{(l_1-s_1+s_2)}$
    $\forall b \in \{0,1\}, P_b$ sets $\langle y\rangle^{(l_2)} = \langle t\rangle^{(l_1-s_1+s_2)}$
  else
    P_0 and P_1 invoke $\Pi^{l_1,s_1-s_2}_{\text{Trunc}}\left(\langle x\rangle^{(l_1)}\right)$ and learn $\langle t\rangle^{(l_1)}$
    $\forall b \in \{0,1\}, P_b$ sets $\langle y\rangle^{(l_2)} = \langle t\rangle^{(l_1)}$
  end if
else
  if $s_1 > s_2$ **then**
    P_0 and P_1 invoke $\Pi^{l_1,s_1-s_2}_{\text{TR}}\left(\langle x\rangle^{(l_1)}\right)$ and learn $\langle t\rangle^{(l_1-s_1+s_2)}$
    P_0 and P_1 invoke $\Pi^{l_1-s_1+s_2,l_2}_{\text{Ext}}\left(\langle t\rangle^{(l_1-s_1+s_2)}\right)$ and learn $\langle y\rangle^{(l_2)}$
  else
    $\forall b \in \{0,1\}, P_b$ sets $\langle t\rangle^{(l_1)} = \langle x\rangle^{(l_1)} \ll (s_2-s_1)$
    P_0 and P_1 invoke $\Pi^{l_1,l_2}_{\text{Ext}}\left(\langle t\rangle^{(l_1)}\right)$ and learn $\langle y\rangle^{(l_2)}$
  end if
end if

Table 6: Accuracy comparison of different quantized networks on ResNet32. Here, residual has the same bit-width as activation, and scales are enforced to power-of-2.

Network	Weight (bits)	Activation (bits)	Acc. (%)
Baseline	FP	FP	68.68
Weight Quantized	2	FP	66.89
Activation Quantized	FP	2	45.30
Fully Quantized	2	2	43.18

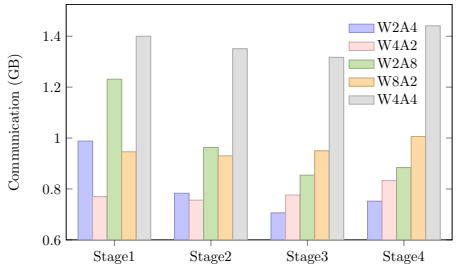

Figure 7: Communication of a single convolution from different stages of ResNet50 with different bit-widths, where W and A mean the bit-width of weight and activation.

Table 7: The impact of residual bit-width on inference accuracy and communication for ResNet32.

Activation (bits)	Weight (bits)	Residual (bits)	Comm. (GB)	Acc. (%)
3	4	3	0.910	66.29
3	4	6	0.972	67.56
3	4	8	0.973	68.10
3	4	16	1.045	68.11

5 COMMUNICATION-AWARE QUANTIZATION

In this section, we introduce our communication-aware network quantization strategy based on the optimized 2PC protocols proposed before.

To reduce communication and improve inference efficiency, we propose leveraging low bit-width quantization for convolutions. We follow [1] and directly quantize the network with ternary weight and activation in Table 6. However, we find it incurring a large accuracy degradation. To improve the accuracy while minimizing the communication of the 2PC-based inference, we consider the following two optimizations during quantization-aware training (QAT).

5.1 High Bit-width Residual

As previous works have revealed[29, 30], we also found through experiments that the quantized ternary activation is indeed the root cause of the accuracy loss.

To improve the network accuracy, we observe the widely used residual connections in a ResNet block can act as a remedy and we propose to use the high bit-width residual connections to improve the activation precision without impacting the operands' bit-widths for the convolution, which incurs negligible overhead as shown in Table 7.

Table 8: PrivQuant evaluation benchmarks.

Model	Layers	# Params (M)	MACs (G)	Dataset
MiniONN [9]	7 Conv, 1 FC, 2 AP, 7 ReLU	0.16	0.061	CIFAR-10
ResNet32	31 Conv, 1 FC, 1 AP, 31 ReLU	0.46	0.069	CIFAR-100
ResNet50	49 CONV, 1 FC, 1, MP, 1 AP, 49 ReLU	2.5	4.1	ImageNet

5.2 Communication-Aware bit-width Optimization

We also use a mixed bit-width quantization strategy to allocate the bit-widths based on communication cost. Existing works [30, 31] widely use the sum of $l_w \cdot l_x \cdot$ FLOPs for each layer as a proxy to estimate the inference cost. However, we observe the 2PC-based inference latency does not correlate well with $l_w \cdot l_x$. We profile the communication of our optimized protocols for different stages with different bit-widths in Figure 7. As we can see, the most efficient bit-width configurations for different layers are different. For example, stage 1 prefers W4A2 while stage 4 prefers W2A4. Hence, we propose a communication-aware mixed bit-width quantization algorithm based on HAWQ [32–34] and leverage our theoretic communication analysis to form a communication cost model to guide the bit-width optimization.

Let H_i denote the Hessian matrix of the i-th layer with a weight tensor W_i. HAWQ finds that layers with a larger trace of H_i are more sensitive to quantization. Hence, the perturbation of the i-th layer, denoted as Ω_i, due to the quantization error can be computed as:

$$\Omega_i = \overline{Tr}(H_i) \cdot ||Q(W_i) - W_i||_2^2,$$

where $\overline{Tr}(H_i)$ is the average Hessian trace, and $||Q(W_i) - W_i||_2$ is the L_2 norm of quantization perturbation. Given the communication bound and a network with L layers, we formulate the communication-aware bit-width optimization as an integer linear programming problem:

$$\text{Objective: } \min_{\{b_i\}_{i=1}^{L}} \sum_{i=1}^{L} \Omega_i^{b_i}$$

$$\text{Subject to: } \sum_{i=1}^{L} \text{Comm}_i^{b_i} \leq \text{Communication Limit}$$

Here, $\Omega_i^{b_i}$ is the i-th layer's sensitivity with bit-width b_i, $\text{Comm}_i^{b_i}$ is the associated communication cost in private inference. The objective is to minimize the perturbation of the whole model under the communication constraint. Here we fix the activation quantization to 4-bit on MiniONN and 6-bit on ResNet and only search for the quantization scheme of weight.

6 EXPERIMENTAL RESULTS

6.1 Experimental Setup

PrivQuant consists of two important parts, i.e., efficient protocols for quantized inference and communication-aware quantization. For network quantization, we use quantization-aware training (QAT) on three architectures and three datasets as shown in Table 8. We first load 8-bit quantization networks as the checkpoints. For MiniONN and ResNet32, we fine-tune the networks on CIFAR-10 and CIFAR-100 for 200 and 100 epochs, respectively, and we fine-tune ResNet50 on ImageNet for 30 epochs. Following [35] and [36], various widely used augmentation techniques are combined to improve the performance of quantized networks. In specific, for

Table 9: Comparing the communication (GB) of our convolution protocol with SOTA (the dimensions are represented by activation resolution, channels, and kernel size.). W2A4 means 2-bit weights and 4-bit activations, and the like.

Conv Dim.	W2A4		W2A6		W2A8	
	SiRNN	PrivQuant	SiRNN	PrivQuant	SiRNN	PrivQuant
(56, 64, 3)	1.26	0.99	1.88	1.10	2.63	1.26
(28, 128, 3)	1.21	0.78	1.80	0.87	2.53	0.96
(14, 256, 3)	1.23	0.71	1.82	0.78	2.55	0.84
(7, 512, 3)	1.30	0.75	1.90	0.82	2.63	0.84
(7, 512, 1)	0.11	0.069	0.17	0.084	0.24	0.081

Table 10: Communication comparison (MB) of residual addition protocols on a basic block of different layers in ResNet32 and ResNet50.

Block	SiRNN	PrivQuant
ResNet32 (1st layer, W2A6)	12.37 (2.6×)	4.86
ResNet32 (3rd layer, W4A6)	3.18 (2.6×)	1.24
ResNet50 (1st layer, W4A6)	605.81 (9.1×)	66.38
ResNet50 (4th layer, W3A6)	87.98 (5.9×)	14.98

CIFAR-10/100, random horizontal flip, random crop, and random erasing are used. For ImageNet, we use AutoAugment [37], CutMix [38], Mixup [39], etc.

For ciphertext execution, we implement protocols based on the Ezpc library in C++ [40–42]. We set $\lambda = 128$ and run our ciphertext evaluation on machines with 2.2 GHz Intel Xeon CPU and 256 GB RAM. For runtime measurements, we follow [43] which is based on a connection between a local PC and an Amazon AWS server. Specifically, we set the bandwidth to 200 Mbps and set the round-trip time to 13 ms.

Our baselines include prior-art frameworks for quantized networks including CrypTFlow2, SiRNN, and COINN. Among them, CrypTFlow2 only supports uniform bit-widths, hence, consistent with [3], we use 37-bit for weight and activation to maintain accuracy. SiRNN supports convolutions with non-uniform bit-widths and following [4], SiRNN uses 16-bit for weight and activation and has at least 32-bit accumulation bit-width. However, we also apply our quantization algorithm to SiRNN to fairly evaluate the effectiveness of our protocols. COINN uses layer-wise mixed bit-width quantization for both weight and activation. Following [2, 17], we evaluate and compare PrivQuant on MiniONN [9], ResNet32 and ResNet50 networks on CIFAR-10, CIFAR-100 and ImageNet datasets, the details are in Table 8.

6.2 Micro-Benchmark Evaluation

Convolution Protocol Evaluation. In Table 9, we compare the performance of the proposed convolution protocols with SiRNN under our low bit-width quantization strategy including W2A4, W2A6, and W2A8 (W2A4 means 2-bit weight and 4-bit activation). We select convolution layers with different dimensions from ResNet50. Due to the unavailability of open-source code for COINN's protocol, we are unable to evaluate the performance of each operation. As shown in Table 9, compared to SiRNN under the same quantization strategy, PrivQuant achieves 1.6 ∼ 3.0× communication reduction through DNN architecture-aware protocol optimization.

Residual Protocol Evaluation. In Table 10, we compare the performance of the simplified residual protocol with SiRNN. We focus on the comparison with SiRNN as other baselines usually ignore these protocols and suffer from computation errors. As shown in Table 10, PrivQuant achieve 2.6 ∼ 9.1× communication reduction on different network layers.

Table 11: End-to-end communication and latency comparison with prior-art methods.

Framework	MiniONN+CIFAR-10		ResNet32+CIFAR-100		ResNet50+ImageNet	
	Comm. (GB)	Latency (s)	Comm. (GB)	Latency (s)	Comm. (GB)	Latency (s)
MiniONN [9]	9.27 (22×)	544.2 (22×)	/	/	/	/
Chameleon [44]	2.65 (6×)	52.7 (2×)	/	/	/	/
LLAMA [45]	1.09 (2.6×)	105.98 (4.2×)	/	/	/	/
GAZELLE [12]	4.90 (12×)	139.4 (6×)	8.90 (10×)	238.7 (4×)	/	/
CrypTFlow2 [3]	6.83 (16×)	291.4 (12×)	8.60 (9×)	374.1 (6×)	377.5 (10×)	16530 (8×)
SiRNN [4]	4.89 (12×)	234.1 (9×)	6.55 (7×)	373.0 (6×)	484.8 (13×)	22153 (11×)
COINN [2]	1.00 (2.4×)	43.0 (1.7×)	1.90 (2×)	78.1 (1.3×)	122.0 (3.2×)	5161 (2.5×)
CoPriv* [8]	/	/	3.69 (3.9×)	166.52 (2.9×)	60.1 (1.6×)	3755 (1.8×)
PrivQuant (ours)	0.42	25.3	0.94	58.4	38.1	2088

* Note that CoPriv is evaluated on MobileNetV2.

6.3 End-to-End Inference Evaluation

Networks Accuracy and Communication Comparison. We now perform an end-to-end inference evaluation. We use communication-aware quantization to sample several quantized networks and perform end-to-end private inference. We draw the Pareto curve of accuracy and communication in Figure 8.

Result and analysis. From Figure 8, we make the following observations: (1) PrivQuant achieves SOTA Pareto front of accuracy and communication in all three benchmarks. More specifically, compared with COINN, PrivQuant achieves 2.4× communication reduction and still 1% higher accuracy in MiniONN. In ResNet32 and ResNet50, PrivQuant can achieve 2.2× and 3.4× communication reduction with the same accuracy. (2) PrivQuant has achieved an improvement of an order of magnitude over CrypTFlow2 and SiRNN. With higher accuracy, PrivQuant can achieve 16×, 12×, 10× communication reduction compared with CrypTFlow2 and 10×, 6.6×, 13× communication reduction compared with SiRNN on MiniONN, ResNet32 and ResNet50, respectively.

Compared with COINN. Our experiments reveal that COINN suffers significant accuracy degradation across all three datasets, for instance, a 3.5% drop in ResNet50. This issue primarily stems from COINN's inability to implement reliable quantization protocols such as extension and re-quantization, leading to errors in both the LSB and MSB. Under conditions of low bit-width quantization, even a 1-bit error can result in a substantial decrease in accuracy. In contrast, PrivQuant avoids such errors entirely and achieves considerably lower communication cost while maintaining accuracy.

Compared with other network optimization work. In Figure 8, we also compare PrivQuant with CoPriv [8] which utilizes winograd convolution to reduce communication. PrivQuant achieves 2.6% higher accuracy with the same communication on ImageNet, demonstrating the effectiveness of our quantization strategy and corresponding protocol optimizations.

Communication and Latency Comparison. To evaluate the communication and latency reduction clearly, we selected one point from each of the three Pareto graphs marked with red circles in Figure 8.

Results and analysis. As shown in Table 11, compared with previous works, with higher accuracy, PrivQuant reduces the communication by 2.4 ∼ 22× and the latency by 1.7 ∼ 22× on MiniONN. On ResNet32, PrivQuant reduces the communication by 2 ∼ 10× and the latency by 1.3 ∼ 6×. On ResNet50, PrivQuant reduces the communication by 1.6 ∼ 13× and the latency by 1.8 ∼ 11×.

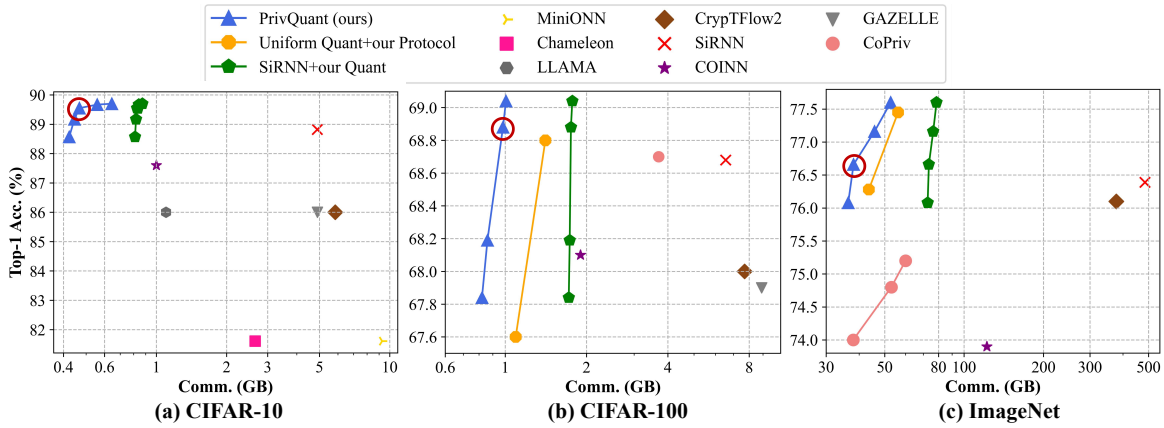

Figure 8: Comparison with prior-art methods on (a) CIFAR-10, (b) CIFAR-100 and (c) ImageNet.

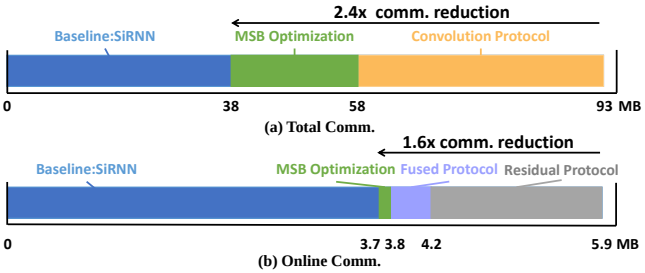

Figure 9: Communication reduction of the proposed protocol optimizations for (a) total comm. and (b) online comm.

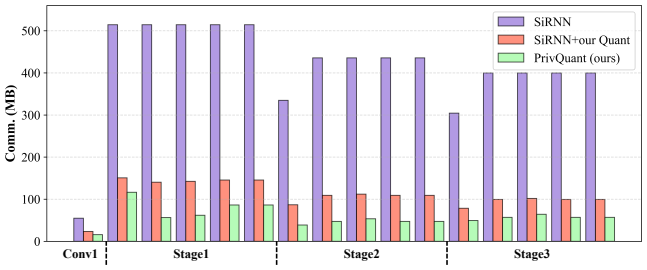

Figure 10: Block-wise comparison of communication on ResNet32/CIFAR-100.

6.4 Ablation Study

Effectiveness of the proposed quantization strategy To understand the importance of the proposed quantization strategy, we use SiRNN's protocol with our quantization strategy called "SiRNN+our Quant" in Figure 8. We can see that our quantization strategy can achieve 3.8×, 5.9 × and 6.6× communication reduction with higher accuracy compared with SiRNN baseline on three benchmarks, respectively.

Moreover, we evaluate the performance of uniform quantization to demonstrate the importance of communication-aware bit width optimization. Specifically, we use W4A4 and W5A5 for both ResNet32 and ResNet50. As shown in Figure 8 (Uniform Quant+our Protocol), communication-aware quantization is clearly superior to uniform quantization. For example, on ResNet32, it outperforms uniform quantization by 2.4% higher accuracy with the same 1.1GB of communication.

Effectiveness of the proposed efficient protocols To prove the effectiveness of our optimized protocols, we dive into a basic block in 3rd stage of ResNet32 with input shape $(8, 8, 64)$. Through our optimizations, we can reduce the total communication and the online communication by 2.4× and 1.6×, respectively, with the detailed breakdown shown in Figure 9. At the end-to-end inference, when comparing PrivQuant with "SiRNN+our Quant" in Figure 8, our optimized protocols can reduce the communication by 1.9 ∼ 2.4× with the same quantization settings. All of these emphasize the significance of our optimized protocols.

Table 12: Comm. comparison in LLAMA-7B self-attention layers (linear part) with different sequence lengths.

Sequence Length	SiRNN	PrivQuant
1	4.86 MB (5×)	0.98 MB
32	767.20 MB (6×)	122.12 MB
128	10769.02 MB (13×)	818.50MB

Blockwise Comparison We show the block-wise communication comparison between SiRNN, SiRNN+our Quant and PrivQuant on ResNet32 in Figure 10. From Figure 10, it is clear that our protocol/network co-optimization is effective, and different layers benefit from PrivQuant differently, which demonstrates the importance of both protocol optimization and network optimization.

7 CONCLUSION

To reduce the communication complexity and enable efficient secure 2PC-based inference, we propose PrivQuant to jointly optimize the secure 2PC-based inference protocols and the quantized networks. Our DNN architecture-aware protocol optimization achieves more than 2.4× communication reduction compared to prior-art protocols. Meanwhile, by network and protocol co-optimization, we can achieve in total 1.6 ∼ 22× communication reduction and 1.3 ∼ 22× inference latency reduction, which improves the practicality of the secure 2PC-based private inference by one step further.

8 FUTURE WORK

Scalability to LLMs. We find PrivQuant can be applied to large language models (LLMs). Recent work [46] has shown the feasibility of quantizing LLMs to low precision, e.g., 3-bit for weights. We conduct an experiment on an attention layer of LLAMA-7B and demonstrate 5 ∼ 13× communication reduction over SiRNN in Table 12. We will further research the scalability of PrivQuant to LLMs in the future.

REFERENCES

[1] M Sadegh Riazi, Mohammad Samragh, Hao Chen, Kim Laine, Kristin Lauter, and Farinaz Koushanfar. {XONN}:{XNOR-based} oblivious deep neural network inference. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1501–1518, 2019.

[2] Siam Umar Hussain, Mojan Javaheripi, Mohammad Samragh, and Farinaz Koushanfar. Coinn: Crypto/ml codesign for oblivious inference via neural networks. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 3266–3281, 2021.

[3] 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.

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

[5] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*, Jun 2017.

[6] 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.

[7] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

[8] Wenxuan Zeng, Meng Li, Haichuan Yang, Wen-jie Lu, Runsheng Wang, and Ru Huang. Copriv: Network/protocol co-optimization for communication-efficient private inference. *Advances in Neural Information Processing Systems*, 36:78906–78925, 2023.

[9] 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.

[10] Oded Goldreich. Secure multi-party computation. 1998.

[11] 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.

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

[13] 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.

[14] Tianshi Xu, Meng Li, Runsheng Wang, and Ru Huang. Falcon: Accelerating homomorphically encrypted convolutions for efficient private mobile network inference. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 1–9, 2023.

[15] Xiaoqian Jiang, Miran Kim, Kristin Lauter, and Yongsoo Song. Secure outsourced matrix computation and application to neural networks. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 1209–1222, 2018.

[16] 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.

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

[18] Fabian Boemer, Rosario Cammarota, Daniel Demmler, Thomas Schneider, and Hossein Yalame. Mp2ml: A mixed-protocol machine learning framework for private inference. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*, pages 1–10, 2020.

[19] 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.

[20] Zahra Ghodsi, Akshaj Kumar Veldanda, Brandon Reagen, and Siddharth Garg. Cryptonas: Private inference on a relu budget. *Advances in Neural Information Processing Systems*, 33:16961–16971, 2020.

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

[22] 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.

[23] Minsu Cho, Zahra Ghodsi, Brandon Reagen, Siddharth Garg, and Chinmay Hegde. Sphynx: Relu-efficient network design for private inference. *arXiv preprint arXiv:2106.11755*, 2021.

[24] 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.

[25] 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.

[26] Tianshi Xu, Meng Li, and Runsheng Wang. Hequant: Marrying homomorphic encryption and quantization for communication-efficient private inference, 2024.

[27] Yehuda Lindell and Benny Pinkas. A proof of security of yao's protocol for two-party computation. *Journal of Cryptology*, page 161–188, Apr 2009.

[28] MichaelO. Rabin. How to exchange secrets with oblivious transfer. *IACR Cryptology ePrint Archive,IACR Cryptology ePrint Archive*, Jan 1981.

[29] Bichen Wu, Yanghan Wang, Peizhao Zhang, Yuandong Tian, Péter Vajda, and Kurt Keutzer. Mixed precision quantization of convnets via differentiable neural architecture search. *ArXiv*, abs/1812.00090, 2018.

[30] Linjie Yang and Qing Jin. Fracbits: Mixed precision quantization via fractional bit-widths. *arXiv preprint arXiv:2007.02017*, 2020.

[31] Ziwei Wang, Han Xiao, Jiwen Lu, and Jie Zhou. Generalizable mixed-precision quantization via attribution rank preservation. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5271–5280, 2021.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 558–567, 2019.

[37] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

[38] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019.

[39] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[40] Nishanth Chandran, Divya Gupta, Aseem Rastogi, Rahul Sharma, and Shardul Tripathi. Ezpc: Programmable, efficient, and scalable secure two-party computation for machine learning. *Cryptology ePrint Archive*, 2017.

[41] Nishant Kumar, Mayank Rathee, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. Cryptflow: Secure tensorflow inference. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 336–353. IEEE, 2020.

[42] Deevashwer Rathee, Anwesh Bhattacharya, Rahul Sharma, Divya Gupta, Nishanth Chandran, and Aseem Rastogi. Secfloat: Accurate floating-point meets secure 2-party computation. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 576–595. IEEE, 2022.

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

[44] M Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M Songhori, Thomas Schneider, and Farinaz Koushanfar. Chameleon: A hybrid secure computation framework for machine learning applications. In *Proceedings of the 2018 on Asia conference on computer and communications security*, pages 707–721, 2018.

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

[46] Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on model compression for large language models. *arXiv preprint arXiv:2308.07633*, 2023.