

# EQO: Exploring Ultra-Efficient Private Inference with Winograd-Based Protocol and Quantization Co-Optimization

Wenxuan Zeng, Tianshi Xu, Meng Li\*, and Runsheng Wang

Peking University  
 {zwx.andy,tianshixu}@stu.pku.edu.cn,  
 {meng.li,r.wang}@pku.edu.cn

**Abstract.** Private convolutional neural network (CNN) inference based on secure two-party computation (2PC) suffers from high communication and latency overhead, especially from convolution layers. In this paper, we propose EQO, a quantized 2PC inference framework that jointly optimizes the CNNs and 2PC protocols. EQO features a novel 2PC protocol that combines Winograd transformation with quantization for efficient convolution computation. However, we observe naively combining quantization and Winograd convolution is sub-optimal: Winograd transformations introduce extensive local additions and weight outliers that increase the quantization bit widths and require frequent bit width conversions with non-negligible communication overhead. Therefore, at the protocol level, we propose a series of optimizations for the 2PC inference graph to minimize the communication. At the network level, We develop a sensitivity-based mixed-precision quantization algorithm to optimize network accuracy given communication constraints. We further propose a 2PC-friendly bit re-weighting algorithm to accommodate weight outliers without increasing bit widths. With extensive experiments, EQO demonstrates  $11.7\times$ ,  $3.6\times$ , and  $6.3\times$  communication reduction with 1.29%, 1.16%, and 1.29% higher accuracy compared to state-of-the-art frameworks SiRNN, COINN, and CoPriv, respectively.

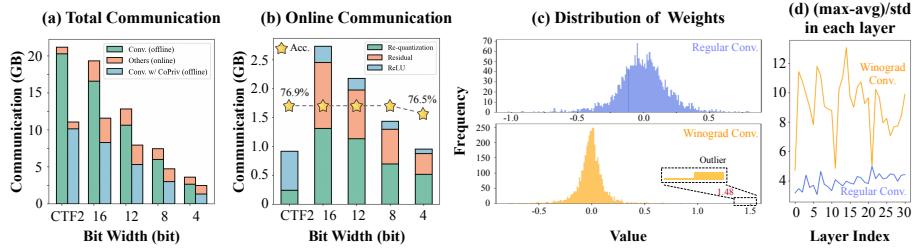
**Keywords:** 2PC-based Private Inference · Quantized Winograd Convolution Protocol · Quantization Bit Re-weighting · Sensitivity-based Mixed-precision Quantization

## 1 Introduction

Deep learning has recently demonstrated superior performance in various privacy-sensitive applications such as personal assistants [25], financial recommendation system [11, 31], medical diagnosis [39, 50], etc. Privacy has thus emerged as one of the major concerns when deploying deep neural networks (DNNs). Secure two-party computation (2PC) is proposed to provide cryptographically strong

---

\* Corresponding author.



**Fig. 1:** (a) Total communication and (b) online communication breakdown on the ResNet-50 building block profiled with CrypTFlow2 (CTF2) [48] (uniform 37-bit) in the first column and SiRNN [47] (supporting mixed precision) in the next four columns; (c) weight distribution in regular and Winograd convolution, (d) the ratio of (max-average) and standard deviation indicates weight outliers consistently exist after Winograd transformation across different layers.

data privacy protection and has attracted more and more attention in recent years [7, 35, 40, 42, 46–48, 53].

2PC inference frameworks target protecting the privacy of both model parameters held by the server and input data held by the client. By jointly executing a series of 2PC protocols, the client can learn the final inference results but nothing else on the model can be derived from the results. Meanwhile, the server knows nothing about the client’s input [20, 35, 38, 40–42, 47, 48, 53].

However, 2PC frameworks achieve high privacy protection at the cost of orders of magnitude latency overhead. Due to the massive interaction between the server and client, 2PC frameworks suffer from high communication cost. As shown in Figure 1(a) and (b), the total inference communication of a convolutional neural network (CNN) is dominated by its convolution layers while the online communication is generated by non-linear functions, e.g., ReLU. To improve communication efficiency, a series of works have proposed network and protocol optimizations [19, 47, 49, 53, 62]. CoPriv [62] proposes a Winograd-based convolution protocol to reduce the communication-expensive multiplications at the cost of local additions. Although it achieves 2 $\times$  communication reduction, it still requires more than 2GB of communication for a single ResNet-18 block. Recent works also leverage mixed-precision quantization for communication reduction [19, 47, 49, 53]. Although total communication reduces consistently with the inference bit widths, as shown in Figure 1(b), existing mixed-precision protocols suffer from much higher online communication even for 4-bit quantized networks due to protocols like re-quantization, residual, etc.

As the communication of 2PC frameworks scales with both the inference bit widths and the number of multiplications, we propose to combine the Winograd optimization with low-precision quantization. However, we observe a naive combination leads to limited improvement. On one hand, although the local additions in the Winograd transformations do not introduce communication directly, they require higher inference bit width and complicate the bit width conversion protocols. On the other hand, the Winograd transformations also reshape the model weight distribution and introduce more outliers that make quantization

**Table 1:** Underlying protocols and the corresponding descriptions used in this paper.  $\lambda$  denotes the security parameter.

Underlying Protocol	Description	Communication Complexity
$\langle z \rangle^{(l_2)} = \Pi_{\text{Ext}}^{l_1, l_2}(\langle x \rangle^{(l_1)})$	Extend $l_1$ -bit $x$ to $l_2$ -bit $z$ such that $z^{(l_2)} = x^{(l_1)}$	$O(\lambda(l_1 + 1) + 13l_1 + l_2)$
$\langle z \rangle^{(l_1)} = \Pi_{\text{Trunc}}^{l_1, l_2}(\langle x \rangle^{(l_1)})$	Truncate (right shift) $l_1$ -bit $x$ by $l_2$ -bit such that $z^{(l_1)} = x^{(l_1)} \gg l_2$	$O(\lambda(l_1 + 3) + 15l_1 + l_2 + 20)$
$\langle z \rangle^{(l_1-l_2)} = \Pi_{\text{TR}}^{l_1, l_2}(\langle x \rangle^{(l_1)})$	Truncate $l_1$ -bit $x$ by $l_2$ -bit and discard the high $l_2$ -bit such that $z^{(l_1-l_2)} = x^{(l_1)} \gg l_2$	$O(\lambda(l_2 + 1) + 13l_2 + l_1)$
$\langle z \rangle^{(l_2)} = \Pi_{\text{Requant}}^{l_1, s_1, l_2, s_2}(\langle x \rangle^{(l_1)})$	Re-quantize $x$ with $l_1$ -bit and $s_1$ scale to $z$ with $l_2$ -bit and $s_2$ scale	Combination of $\Pi_{\text{Ext}}, \Pi_{\text{Trunc}}, \Pi_{\text{TR}}$

harder as shown in Figure 1(c). As a result, a naive combination only reduces  $\sim 20\%$  total communication with even higher online communication compared to not using Winograd convolution as shown in Figure 1(a).

In this paper, we propose a communication-efficient 2PC framework named EQO. EQO carefully combines Winograd transformation with quantization and features a series of protocol and network optimizations to address the aforementioned challenges. Our contributions can be summarized below:

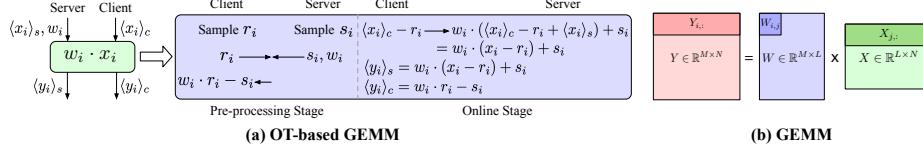
- We observe the communication of 2PC inference scales with both the bit widths and the number of multiplications. Hence, we propose EQO, which combines Winograd convolution and mixed-precision quantization for efficient 2PC inference for the first time. A series of graph-level optimizations are further proposed to reduce the online communication cost.
- We propose a communication-aware mixed-precision quantization algorithm, and further develop a 2PC-friendly bit re-weighting algorithm to handle the outliers introduced by the Winograd convolution.
- Extensive experiments demonstrate that EQO achieves  $11.7\times$ ,  $3.6\times$ , and  $6.3\times$  communication reduction with  $1.29\%$ ,  $1.16\%$ , and  $1.29\%$  higher accuracy compared to state-of-the-art frameworks SiRNN, COINN, and CoPriv, respectively.

## 2 Preliminaries

Following [6, 20, 29, 37, 40, 42, 47, 48, 53], EQO adopts a semi-honest attack model where both the server and client follow the protocol but also try to learn more from the information than allowed. We provide a detailed description of the threat model in Appendix A. For the convenience of readers, we summarize the underlying protocols used in this paper in Table 1 and notations in Appendix B. Due to the space constraint, we put more preliminaries including Winograd convolution in Appendix C.

### 2.1 Oblivious Transfer (OT)-based Linear Layers

Figure 2(a) shows the flow of 2PC-based inference. With arithmetic secret sharing (ArSS), each intermediate activation tensor, e.g.,  $x_i$ , is additively shared



**Fig. 2:** (a) Flow of OT-based linear layer, e.g., GEMM, including a pre-processing stage to generate input-independent helper data and an online stage to process client’s input. (b) An illustration of a GEMM  $Y = WX$ .

where the server holds  $\langle x_i \rangle^S$  and the client holds  $\langle x_i \rangle^C$  such that  $x_i = \langle x_i \rangle^S + \langle x_i \rangle^C \bmod P$  [42]. To generate the result  $y_i$  of a linear layer, e.g., general matrix multiplication (GEMM), a series of 2PC protocols is executed in the pre-processing stage and online stage [40]. In the pre-processing stage, the client and server first sample random  $r_i$  and  $s_i$ , respectively. Then,  $\langle y_i \rangle_c = w_i \cdot r_i - s_i$  can be computed with a single OT if  $r_i \in \{0, 1\}$ . With the vector optimization [19], one OT can be extended to compute  $w_i \cdot r_i - s_i$ , where  $r_i$  and  $s_i$  are both vectors. When  $w_i$  has  $l_w$  bits, we can repeat the OT protocol  $l_w$  times by computing  $w_i^{(b)} \cdot r_i - s_i$  each time, where  $w_i^{(b)}$  denotes  $b$ -th bit of  $w_i$ . The final results can then be acquired by shifting and adding the partial results together. Compared with the pre-processing stage, the online stage only requires very little communication to obtain  $\langle y_i \rangle_s = w_i \cdot (x_i - r_i) + s_i$ .

## 2.2 Related Works

Existing 2PC-based private inference frameworks can be divided into two categories, i.e., OT-based and homomorphic encryption (HE)-based. The HE-based frameworks [15, 18, 43, 56] leverages HE to compute linear layers and achieve lower communication compared to OT-based frameworks at the cost of more computation overhead for both the server and client. Hence, HE-based and OT-based protocols have different applicable scenarios [19, 62]. For example, for resource-constrained clients, HE-based protocol may not be applicable since it involves repetitive encoding, encryption, and decryption on the client side [14, 27]. Hence, we focus on optimizing OT-based frameworks to improve the communication efficiency in this work.

In recent years, there has been an increasing amount of literature on efficient OT-based private inference, including protocol optimization [7, 26, 41, 42, 46–48], network optimization [6, 20, 29, 32, 37, 61], and network/protocol joint optimization [19, 40, 62]. To reduce the communication overhead, quantization has been used for private inference [1, 16, 19, 47, 49, 51, 53]. In Table 2, we compare EQO with previous works qualitatively. As can be observed, EQO jointly optimizes both protocol and network and simultaneously reduces the number of multiplication and communication per multiplication. We leave the detailed review of existing works to Appendix D.

**Table 2:** Comparison with existing works in terms of quantized optimization. “✓” denotes the communication can be optimized. C/M means communication per multiplication (MUL).

Framework	Protocol Optimization		Network Optimization	Linear		Non-linear
	Operator Level	Graph Level		#MUL	C/M	
[6, 20, 21, 29]	-	-	ReLU-aware Pruning	-	-	✓
SiRNN [47]	16-bit Quant.	MSB Opt.	-	-	✓	✓
XONN [49]	Binary Quant.	-	End-to-end Binary Quant. (Large Acc. Degradation)	-	✓	✓
COINN [19]	Factorized GEMM	Protocol Conversion	Mixed-precision Quant.	✓	✓	✓
ABNN2 [53]	-	-	Uniform Quant.	-	✓	✓
QUOTIENT [1]	-	-	Gradient Quant. Adaptive Gradient.	-	✓	✓
EQO (ours)	Mixed-precision Quant. Winograd Conv.	Simplified Residual Protocol Fusion MSB Opt.	Comm.-aware Quant. Mixed-precision Quant. High-precision Residual Bit Re-weighting	✓	✓	✓

### 3 Motivations and Overview

In this section, we analyze the communication complexity of OT-based 2PC inference. We also explain the challenges when combining Winograd transformation with quantization, which motivates EQO.

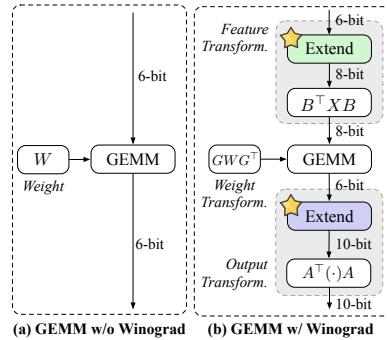
*Observation 1: the total communication of OT-based 2PC is determined by both the bit widths and the number of multiplications in linear layers.* Consider an example of  $Y = WX$ , where  $W \in \mathbb{R}^{M \times L}$ ,  $X \in \mathbb{R}^{L \times N}$  and  $Y \in \mathbb{R}^{M \times N}$  in Figure 2(b). With one round of OT, we can compute  $W_{i,j}^{(b)} \cdot X_{j,:}$  for the  $b$ -th bit of  $W_{i,j}$  and  $j$ -th row of  $X$ . Then, the  $i$ -th row of  $Y$ , denoted as  $Y_{i,:}$ , can be computed as

$$Y_{i,:} = \sum_{j=0}^{L-1} \sum_{b=0}^{l_w-1} 2^b \cdot W_{i,j}^{(b)} \cdot X_{j,:},$$

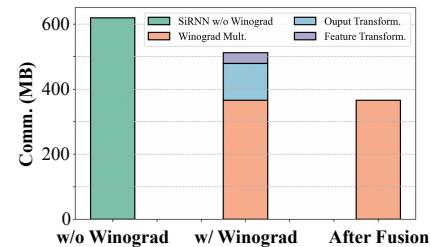
where  $l_w$  denotes the bit width of  $W$ . Hence, to compute  $Y_{i,:}$ , in total  $O(l_w L)$  OTs are invoked. In each OT, the communication scales proportionally with the vector length of  $X_{j,:}$ , i.e.,  $O(N l_x)$ , where  $l_x$  denotes the bit width of  $X$ . The total communication of the GEMM thus becomes  $O(MLN l_w l_x)$ . Thus, we observe the total communication of a GEMM scales with both the operands’ bit widths, i.e.,  $l_x$  and  $l_w$ , and the number of multiplications, i.e.,  $MLN$ , both of which impact the round of OTs and the communication per OT. Convolutions follow a similar relation as GEMM. Hence, combining Winograd transformations and quantization is a promising solution to improve the communication efficiency of convolution layers. For non-linear layers, e.g., ReLU, the communication cost also scales proportionally with activation bit widths [47].

*Observation 2: a naive combination of Winograd transformations and quantization provides limited communication reduction.* Although Winograd transformation reduces the number of multiplications, it is not friendly to quantization as it

**Fig. 3:** Insertion of bit extension (★) with quantized inference. There is a re-quantization in the GEMM protocol.

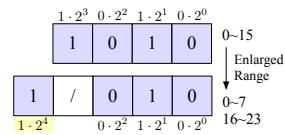


**Fig. 4:** Communication breakdown of Winograd convolution on ResNet-32. After graph-level protocol fusion introduced in Section 4.2, the communication of both feature and output transformation are reduced.

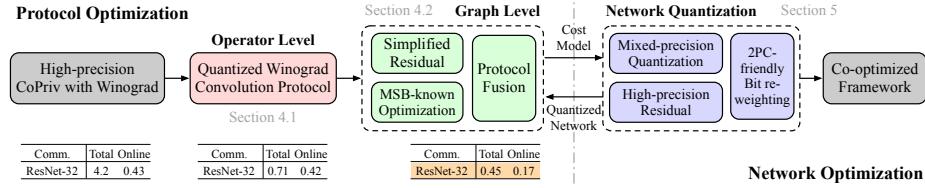


introduces many more local additions during the feature and output transformation. Hence, to guarantee the computation correctness with avoiding overflow, extra bit width conversion protocols need to be performed as shown in Figure 3. Take ResNet-32 with 2-bit weight and 6-bit activation (abbreviated as W2A6) as an example in Figure 4, naively combining Winograd transformation and quantization achieves only  $\sim 20\%$  communication reduction compared to not using Winograd convolution in SiRNN, which is far less compared to [62]. Hence, protocol optimization is important to reduce the overhead induced by the bit width conversions.

*Observation 3: Winograd transformations introduce quantization outliers and make low-precision weight quantization challenging.* We show the weight distribution with and without Winograd transformations in Figure 1(c). The weight transformation involves multiplying or dividing large plaintext integers [30] and tends to generate large weight outliers, which makes the low-precision quantization challenging. Instead of simply increasing the quantization bit width, we observe it is possible to accommodate the weight outliers by bit re-weighting. Recall for OT-based linear layers, each weight is first written as  $\sum_{b=0}^{l_w-1} w^{(b)} \cdot 2^b$  (we ignore the sign bit for convenience) and then, each bit  $w^{(b)}$  is multiplied with the corresponding activations with a single OT. The final results are acquired by combining the partial results by shift and addition. This provides us with unique opportunities to re-weight each bit by adjusting  $2^b$  to increase the representation range without causing extra communication overhead. An example is shown in Figure 5. As can be observed, through bit re-weighting, we can increase the representation range by  $2\times$  with the same quantization bit width (note the total number of possible represented values remain the same).



**Fig. 5:** Example of bit re-weighting with adjusted representation range.



**Fig. 6:** Overview of co-optimization framework EQO and the communication cost (GB) after each optimization step. Example is evaluated on ResNet-32 with W2A6.

*Overview of EQO* Based on these observations, we propose EQO, a network/protocol co-optimization framework for communication-efficient 2PC inference. We show the overview of in Figure 6. We first optimize the 2PC protocol for convolutions combining quantization and Winograd transformation (Section 4.1). We then propose a series of graph-level optimizations, including protocol fusion to reduce the communication for Winograd transformations (Section 4.2), simplified residual protocol to remove redundant bit width conversion protocols (Section 4.2), and graph-level activation sign propagation and protocol optimization given known most significant bits (MSBs) (Section 4.2). In Figure 6, although Winograd is quantized to low precision, there is no benefit for online communication due to the extra bit width conversions. As a result, the graph-level optimizations enable to reduce the online communication by  $2.5\times$  in the example. At the network level, we further propose communication-aware mixed-precision quantization and bit re-weighting algorithms to reduce the inference bit widths as well as communication (Section 5). EQO reduces the total communication by  $9\times$  in the example.

## 4 EQO Protocol Optimization

#### 4.1 Quantized Winograd Convolution Protocol

As explained in Section 3, when combining the Winograd transformation and quantization, extra bit width conversions, i.e., extensions are needed to guarantee compute correctness due to local additions in the feature and output transformations. Two natural questions hence arise: 1) where to insert the bit width extension protocols? 2) how many bits to extend?

For the first question, there are different ways to insert the bit extension. Take the output transformation as an example. One method is to extend the activations right before computing the output transformation, which incurs online communication. The second method is to insert the bit width extensions before the GEMM protocol. While this enables to merge the bit extension protocols

$$X \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -1 & -1 \\ -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \quad X \cdot \begin{bmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & -4 & 4 & 2 & 4 \\ -5 & -4 & -4 & -1 & 0 \\ 0 & 1 & -1 & 2 & -5 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

**Fig. 7:** GEMM of  $X$  and transformation matrix  $B$ . The columns colored in grey mark the possible maximum number of additions.

with the offline weight transformation, we find it will drastically increase the GEMM communication and thus, choose the first method in EQO.

For the second question, we have the following lemma that bounds the outputs after transformation.

**Lemma 1.** *Consider  $Y = XB$ . For each element  $Y_{i,j}$ , its magnitude can always be bounded by*

$$|Y_{i,j}| = \left| \sum_k X_{i,k} B_{k,j} \right| \leq \sum_k |X_{i,k}| |B_{k,j}| \leq 2^{l_x} \sum_k |B_{k,j}| = 2^{l_x + \log_2 \|B_{:,j}\|_1},$$

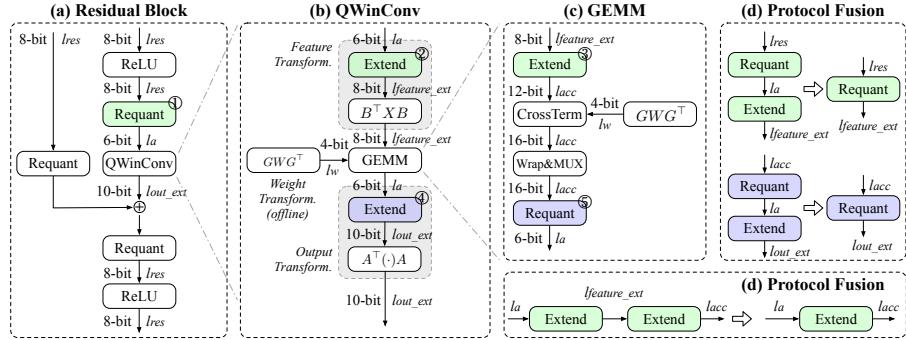
where  $\|\cdot\|_1$  is the  $\ell_1$ -norm.

Therefore, each output element  $Y_{i,j}$  requires at most  $l_x + \log_2 \|B_{:,j}\|_1$  bits. Since we use per-tensor activation quantization, to guarantee the computation correctness, we need  $\max_j l_x + \log_2 \|B_{:,j}\|_1$  bits to represent the output. Similarly, to compute  $B^\top X$ ,  $\max_j \log_2 \|B_{:,j}\|_1$  is needed, adding up to  $2 \times \max_j \log_2 \|B_{:,j}\|_1$  bits to extend. In Figure 7, we show the transformation matrix  $B$  for Winograd convolution with the output tile size of 2 and 4. As can be calculated, 2-bit and 8-bit extensions are needed, respectively. As 8-bit extension drastically increases the communication of the following GEMM, we choose 2 as the output tile size. The bit extension for the output transformation can be calculated similarly.

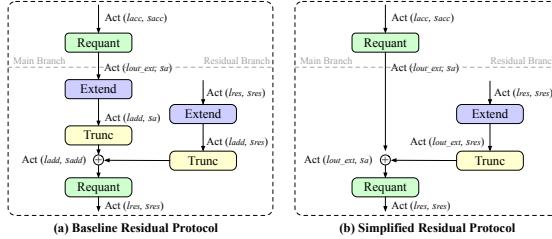
Based on the above analysis, we propose the quantized Winograd convolution protocol, dubbed QWinConv, in Figure 8. Consider an example of 4-bit weights and 6-bit activations. In Figure 8(a), before QWinConv, following SiRNN [47], activations are first re-quantized to 6-bit to reduce the communication of QWinConv (block ①). We always keep the residual in higher bit width, i.e., 8-bit, for better accuracy [55, 59]. After QWinConv, the residual and QWinConv output are added together with the residual protocol. The design of QWinConv is shown in Figure 8(b), which involves four steps: feature transformation, weight transformation, Winograd-domain GEMM, and output transformation. The GEMM protocol in Figure 8(c) follows SiRNN [47] and the extension (block ③) ensures accumulation correctness. Weight transformation and its quantization can be conducted offline, while feature and output transformation and quantization must be executed online. We insert bit extensions right before the feature and output transformation as marked with block ② and ④, respectively.

## 4.2 Graph-level Protocol Optimization

*Graph-level Protocol Fusion* As explained in Section 3, extra bit width conversion protocols in QWinConv (② and ④ in Figure 8) increase the online communication and diminish the benefits of combining quantization with Winograd transformation, especially for low-precision quantized networks, e.g., 4-bit. We observe there are neighboring bit width conversion protocols that can be fused to reduce the overhead. Specifically, we find two patterns that appear frequently: 1) consecutive bit width conversions, e.g., ① and ②, ④ and ⑤; 2) bit width conversions that are only separated by local operations, e.g., ② and ③. As the cost of



**Fig. 8:** Overall framework of quantized Winograd convolution protocol EQO in (a) an example residual block; (b) the design of QWinConv; (c) GEMM protocol in SiRNN [47]; (d) graph-level protocol fusion.

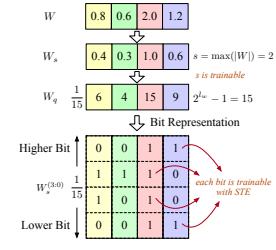


**Fig. 9:** Comparison between (a) baseline residual protocol [47] and (b) our simplified residual protocol.  $l, s$  mean the bit width and scale of the operand, respectively.

bit width conversion protocols is only determined to the bit width of inputs [47], such protocol fusion enables complete removal of the cost of ②, ③ and ⑤, which completely removes the cost of feature and output transformations. We refer interested readers to a formal description in Appendix I.

*Simplified Residual Protocol* As shown in Figure 1(b), residual protocol consumes around 50% of the online communication due to the alignment of both bit widths and scales<sup>1</sup> as shown in Figure 9(a). Therefore, we propose a new simplified residual protocol to reduce the communication as shown in Figure 9(b). Specifically, we directly align the bit width and scale of residual to the output of QWinConv for addition. In this way, we get rid of the redundant bit width conversion protocol in the main branch, reducing the communication from  $O(\lambda(l_{out\_ext} + l_{res} + 2l_{add} + 8))$  to  $O(\lambda(l_{out\_ext} + l_{res} + 4))$ .

<sup>1</sup> We explain the details of bit width and scales in quantization in Appendix C.3.



**Fig. 10:** Conversion of bit representation with bit width  $l_w = 4$ . During training, each bit in  $W_s^{(3:0)}$  is trainable.

*MSB-known Optimization* As pointed out by [15, 18, 47], 2PC protocols can be designed in a more efficient way when the MSB of the secret shares is known. Since the activation after ReLU is always non-negative, protocols including truncation and extension can be further optimized. In EQO, we locate the ReLU function and then, propagate the sign of the activations to all the downstream protocols. In Figure 8, for example, the input of re-quantization and extensions in green (block ①, ② and ③) must be non-negative, so they can be optimized. In contrast, re-quantization and extension in blue (block ④ and ⑤) can not be optimized since the GEMM outputs can be either positive or negative.

*Security and Correctness Analysis* EQO is built on top of SiRNN [47] with new quantized Winograd convolution protocol and graph-level protocol optimizations. The security of the quantized Winograd convolution protocol directly follows from the security guarantee from OT. Observe that all the communication between the client and the server is performed through OT which ensures the privacy of both the selection bits and messages. The graph-level protocol optimizations, including protocol fusion, residual simplification, and MSB-known optimization leverage information known to both parties, e.g., the network architecture, and thus, do not reveal extra information. The correctness of the quantized Winograd convolution protocol is guaranteed by the theorem of Winograd transformation [30] and bit width extensions to avoid overflow.

## 5 EQO Network Quantization Optimization

In this section, we propose Winograd-domain quantization algorithm that is compatible with OT-based 2PC inference. The overall training procedure is shown in Algorithm 1. We first assign different bit widths to each layer based on the quantization sensitivity, and then propose 2PC-friendly bit re-weighting to improve the quantization performance.

### 5.1 Communication-aware Sensitivity-based Quantization

Different layers in a CNN have different sensitivity to the quantization noise [9, 10, 60]. To enable accurate and efficient private inference of EQO, we propose a communication-aware mixed-precision algorithm to search for the optimal bit width assignment for each layer based on sensitivity. Sensitivity can be approximately measured by the average trace of the Hessian matrix. Following [9], let  $\Omega_i$  denote the output perturbation induced by quantizing the  $i$ -th layer. Then, we have

$$\Omega_i = \overline{Tr}(H_i) \cdot \|\text{Quant}(GW_iG^\top) - GW_iG^\top\|_2^2,$$

where  $H_i$  and  $\overline{Tr}(H_i)$  denote the Hessian matrix of the  $i$ -th layer and its average trace,  $GW_iG^\top$  denotes the Winograd-domain weight,  $\text{Quant}(\cdot)$  denotes quantization, and  $\|\cdot\|_2$  denotes the  $\ell_2$ -norm of quantization perturbation. Given the communication bound  $\zeta$  and the set of admissible bit widths  $\mathcal{B}$ , inspired

**Algorithm 1:** Training Procedure of EQO Quantization

---

**Input :** Pre-trained floating-point weight  $w_f$ , communication bound  $\zeta$ , finetuning epochs  $E$

**Output:** Set of bit weights  $\mathbb{B}$  and quantized weight  $w$

```

1  $l_w = \text{HessianAlgorithm}(w_f, \zeta)$ ; /* Bit width assignment for each layer
   based on Hessian matrix */
2 for  $i \in [0, \dots, L - 1]$  do
3    $\mathbb{B}^{(i)} \leftarrow \{2^{l_w^{(i)}-1}, 2^{l_w^{(i)}-2}, \dots, 2^1, 2^0\}$ ;
4   if has_outlier( $w^{(i)}$ ) then
5      $\mathbb{B}^{(i)} \leftarrow \{2^{l_w^{(i)}}, 2^{l_w^{(i)}-2}, \dots, 2^1, 2^0\}$ ; /* Bit re-weighting */
6 for  $e \in [0, \dots, E - 1]$  do
7   Forward propagation based on Equation 1; /* Re-weighted
      quantization */
8   Compute loss  $\mathcal{L}_{CE}$ ;
9   Back propagation based on Equation 2; /* Training with STE */
10  Update  $w^{(b)}$  with SGD optimizer;
11 return  $w$  and  $\mathbb{B}$ 

```

---

by [60], we formulate the communication-aware bit width assignment problem as an integer linear programming problem (ILP):

$$\min \sum_{i \in [1, L]} \sum_{j \in \mathcal{B}} T_{i,j} \cdot \Omega_{i,j}, \quad \text{s.t. } \sum_{i \in [1, L]} \sum_{j \in \mathcal{B}} T_{i,j} \cdot C_{i,j} \leq \zeta, \sum_{j \in \mathcal{B}} T_{i,j} = 1, \forall i \in [1, L],$$

where  $T_{i,j} \in \{0, 1\}$ ,  $C_{i,j}$ , and  $\Omega_{i,j}$  are the indicator, communication cost, and perturbation when quantizing the  $i$ -th layer to  $j$ -bit, respectively. The objective is to minimize the perturbation in the network output under the given communication constraint.

## 5.2 2PC-friendly Bit Re-weighting Algorithm

*Bit Re-weighting* Based on the above Hessian-based bit width assignment, we obtain the bit width of each layer. However, as mentioned in Section 3, outliers introduced by Winograd transformations make quantization challenging. Recall for OT-based linear layers, each weight is first written as  $\sum_{b=0}^{l_w-1} w^{(b)} \cdot 2^b$  and then, each bit  $w^{(b)}$  is multiplied with the corresponding activations with a single OT. This provides us with opportunities to re-weight each bit by adjusting  $2^b$  to increase the representation range without causing extra communication overhead. We define  $2^b$  as the importance for  $b$ -th bit and define  $\mathbb{B} = \{2^{l_w-1}, 2^{l_w-2}, \dots, 2^1, 2^0\}$ . The OT-based computation can be re-written as  $\sum_{b=0}^{l_w-1} w^{(b)} \cdot \mathbb{B}[b]$ . We first determine whether there is an outlier situation in each layer based on the ratio between the maximum and the standard deviation of the weights. Then, for the layers with large outliers, we re-weight the bit by adjusting  $2^b$  to  $2^{b'} (b' > b)$  to increase the representation range flexibly. Specifically,

we increase MSB by adjusting  $\mathbb{B}$  to  $\{2^{l_w}, 2^{l_w-2} \dots, 2^1, 2^0\}$  to accommodate the outliers flexibly.

*Finetuning* After the bit re-weighting, we further conduct quantization-aware finetuning to improve the accuracy of the quantized networks. Since the bit importance is adjusted, we find it is more convenient to leverage the bit-level quantization strategy [58] to finetune our networks. Bit-level quantization with bit representation is demonstrated in Figure 10. More specifically, the quantized weight after bit re-weighting can be formulated as

$$\textbf{Forward: } w_q = s \cdot \text{Round}\left(\sum_{b=0}^{l_w-1} w^{(b)} \cdot \mathbb{B}[b]\right)/(2^{l_w} - 1), \quad (1)$$

$$\textbf{Backward: } \frac{\partial \mathcal{L}}{\partial w^{(b)}} = \frac{2^b}{2^{l_w} - 1} \frac{\partial \mathcal{L}}{\partial w_q} = \frac{2^b}{2^{l_w} - 1} \frac{\partial \mathcal{L}_{CE}(\mathcal{M}_{w_q}(x), y)}{\partial w_q}, \quad (2)$$

where  $(x, y)$  denotes input-label pair,  $s$  denotes the scaling factor,  $\mathcal{L}_{CE}$  denotes cross-entropy loss for classification task,  $\mathcal{M}$  denotes the model architecture. During finetuning,  $w^{(b)}$  and  $s$  are trainable via STE [4].

## 6 Experiments

*Experimental Setups* EQO is implemented on the top of SiRNN [47] in EzPC<sup>2</sup> library. Following [18, 48], we use LAN modes for communication. For network quantization, we use quantization-aware training (QAT) for the Winograd-based networks. We conduct experiments on different datasets and networks, i.e., MiniONN [35] on CIFAR-10, ResNet-32 [17] on CIFAR-100, ResNet-18 [17] on Tiny-ImageNet and ImageNet [8]. For baselines, DeepReDuce [21], SNL [6], SAFENet [37], and SENet [29] are ReLU-optimized methods evaluated under SiRNN [47]. Please refer to Appendix E for details, including the network and protocol used in each baseline.

### 6.1 Micro-benchmark on Efficient Protocols

*Convolution Protocol* To verify the effectiveness of our proposed convolution protocol, we benchmark the convolution communication in Table 3. We compare EQO with SiRNN [47] and CoPriv [62] given different layer dimensions. As can be observed, with W2A4, EQO achieves 15.8~24.8× and 9.6~14.7× communication reduction, compared to SiRNN and CoPriv, respectively.

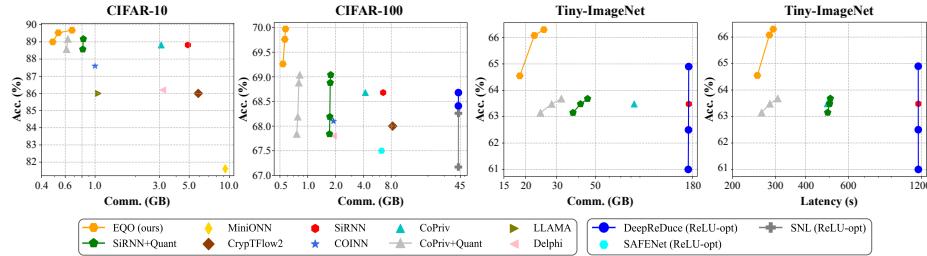
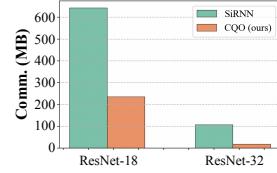
*Residual Protocol* As shown in Figure 11, we compare our simplified residual protocol with SiRNN. EQO achieves 2.7× and 5.6× communication reduction on ResNet-18 (ImageNet) and ResNet-32 (CIFAR-100), respectively.

---

<sup>2</sup> <https://github.com/mpc-msri/EzPC>

**Table 3:** Micro-benchmark (MB) of convolution protocol.

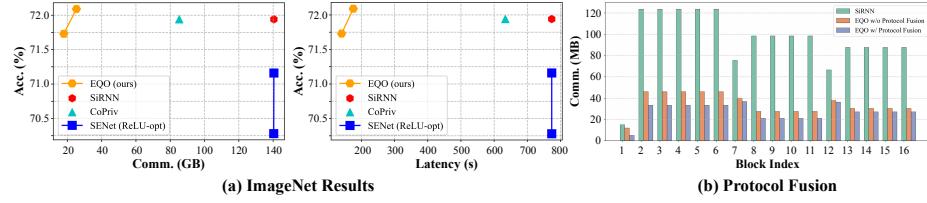
Dimension				SiRNN	CoPriv	EQO (W2A4)	EQO (W2A6)
H	W	C	K				
32	32	16	32	414.1	247.3	25.75	30.88
16	16	32	64	380.1	247.3	17.77	20.33
56	56	64	64	9336	5554	376.9	440.6
28	28	128	128	8598	5491	381.9	438.9

**Fig. 11:** Micro-benchmark of residual protocol.**Fig. 12:** Comparison with prior-art methods on three datasets.

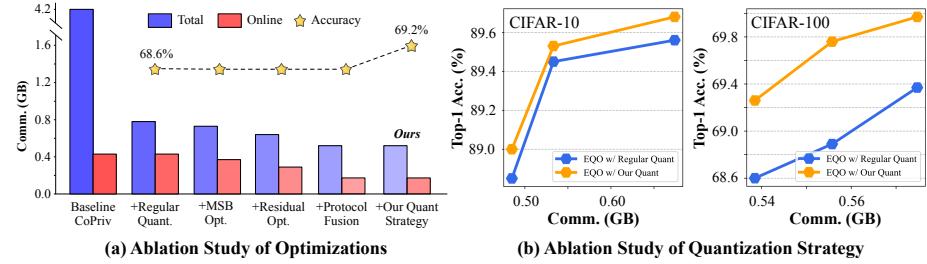
## 6.2 Benchmark on End-to-end Inference

*CIFAR-10 and CIFAR-100 Evaluation* From Figure 12(a) and (b), we make the following observations: 1) EQO achieves state-of-the-art (SOTA) Pareto front of the accuracy and communication. Specifically, on CIFAR-10, EQO outperforms SiRNN with 0.71% higher accuracy and 9.41× communication reduction. On CIFAR-100, EQO achieves 1.29% higher accuracy and 11.7×/6.33× communication reduction compared with SiRNN/CoPriv; 2) compared with COINN, EQO achieves 1.4% and 1.16% higher accuracy as well as 2.1× and 3.6× communication reduction on CIFAR-10 and CIFAR-100, respectively; 3) we also compare EQO with ReLU-optimized methods. The result shows these methods cannot effectively reduce total communication, and EQO achieves more than 80× and 15× communication reduction with even higher accuracy, compared with DeepReDuce/SNL and SAFENet, respectively;

*Tiny-ImageNet and ImageNet Evaluation* We compare EQO with SiRNN, CoPriv, and ReLU-optimized method DeepReDuce on Tiny-ImageNet. As shown in Figure 12(c), we observe that 1) EQO achieves 9.26× communication reduction with 1.07% higher accuracy compared with SiRNN; 2) compared with SiRNN equipped with mixed-precision quantization, EQO achieves 2.44× communication reduction and 0.87% higher accuracy; 3) compared with CoPriv, EQO achieves 4.5× communication reduction with 1.07% higher accuracy. We also evaluate the accuracy and efficiency of EQO on ImageNet. As shown in Figure 13(a), EQO achieves 4.88× and 2.96× communication reduction with 0.15% higher accuracy compared with SiRNN and CoPriv, respectively.



**Fig. 13:** (a) Evaluation on ImageNet. (b) Ablation study and block-wise visualization of graph-level protocol fusion of Winograd transformation.



**Fig. 14:** Ablation studies of EQO.

### 6.3 Ablation Study

*Effectiveness of Quantization Strategy* As shown in Figure 14(b), our proposed quantization algorithm achieves 0.1~0.2% higher accuracy and more than 0.6% higher accuracy on CIFAR-10 and CIFAR-100 without sacrificing efficiency, respectively, demonstrating the efficacy of EQO.

*Block-wise Visualization of Protocol Fusion* To demonstrate the effectiveness of protocol fusion of Winograd transformation, we take W2A6 as an example to conduct the ablation experiment on ResNet-32 on CIFAR-100. As shown in Figure 13(b), protocol fusion further saves 30% communication compared to without fusion. We also find the communication portion of bit width conversions for Winograd transformation in the early layer is larger than in the later layers.

*Effectiveness of Different Optimizations* To understand how different optimizations help improve communication efficiency, we add the protocol optimizations step by step on ResNet-32, and present the results in Figure 14(a). As observed from the results, we find that 1) low-precision Winograd quantization benefits the total communication efficiency most. However, there is no benefit to online communication due to the extra bit width conversions for Winograd transformation even though it is low precision; 2) simplified residual protocol, protocol fusion, and MSB-known optimization consistently reduce the online and total communication; 3) although naively combining Winograd convolution with quantized private inference enlarges the online communication, our optimizations finally achieve 8.1 $\times$  and 2.5 $\times$  total and online communication reduction compared to

CoPriv; 4) EQO with bit re-weighting achieves higher accuracy without sacrificing efficiency. The findings indicate that all of our optimizations are indispensable for 2PC-based inference efficiency.

## 7 Conclusion

In this work, we propose EQO , a communication-efficient 2PC-based framework with Winograd and mixed-precision quantization. We observe naively combining quantization and Winograd convolution is sub-optimal. Hence, at the protocol level, we propose a series of optimizations for the 2PC inference graph to minimize the communication. At the network level, we develop a sensitivity-based mixed-precision quantization algorithm and a 2PC-friendly bit re-weighting algorithm to accommodate weight outliers without increasing bit widths. With extensive experiments, EQO consistently reduces the communication without compromising the accuracy compared with the prior-art 2PC frameworks and network optimization methods.

## References

1. Agrawal, N., Shahin Shamsabadi, A., Kusner, M.J., Gascón, A.: Quotient: two-party secure neural network training and prediction. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 1231–1247 (2019) [4](#), [5](#)
2. Alam, S.A., Anderson, A., Barabasz, B., Gregg, D.: Winograd convolution for deep neural networks: Efficient point selection. ACM Transactions on Embedded Computing Systems **21**(6), 1–28 (2022) [21](#)
3. Andri, R., Bussolino, B., Cipolletta, A., Cavigelli, L., Wang, Z.: Going further with winograd convolutions: Tap-wise quantization for efficient inference on 4x4 tiles. In: 2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO). pp. 582–598. IEEE (2022) [22](#)
4. Bengio, Y., Léonard, N., Courville, A.: Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432 (2013) [12](#)
5. Chikin, V., Kryzhanovskiy, V.: Channel balancing for accurate quantization of winograd convolutions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12507–12516 (2022) [22](#)
6. Cho, M., Joshi, A., Reagen, B., Garg, S., Hegde, C.: Selective network linearization for efficient private inference. In: International Conference on Machine Learning. pp. 3947–3961. PMLR (2022) [3](#), [4](#), [5](#), [12](#), [20](#), [22](#), [24](#)
7. Demmler, D., Schneider, T., Zohner, M.: Aby-a framework for efficient mixed-protocol secure two-party computation. In: NDSS (2015) [2](#), [4](#), [22](#)
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009) [12](#)
9. Dong, Z., Yao, Z., Arfeen, D., Gholami, A., Mahoney, M.W., Keutzer, K.: Hawq-v2: Hessian aware trace-weighted quantization of neural networks. Advances in neural information processing systems **33**, 18518–18529 (2020) [10](#)

10. Dong, Z., Yao, Z., Gholami, A., Mahoney, M.W., Keutzer, K.: Hawq: Hessian aware quantization of neural networks with mixed-precision. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 293–302 (2019) [10](#)
11. Fan, W., Zhao, Z., Li, J., Liu, Y., Mei, X., Wang, Y., Tang, J., Li, Q.: Recommender systems in the era of large language models (llms). arXiv preprint arXiv:2307.02046 (2023) [1](#)
12. Fernandez-Marques, J., Whatmough, P., Mundy, A., Mattina, M.: Searching for winograd-aware quantized networks. Proceedings of Machine Learning and Systems **2**, 14–29 (2020) [22](#), [23](#)
13. Ganesan, V., Bhattacharya, A., Kumar, P., Gupta, D., Sharma, R., Chandran, N.: Efficient ml models for practical secure inference. arXiv preprint arXiv:2209.00411 (2022) [22](#)
14. Gupta, S., Cammarota, R., Rosing, T.Š.: Memfhe: End-to-end computing with fully homomorphic encryption in memory. ACM Transactions on Embedded Computing Systems (2022) [4](#)
15. Hao, M., Li, H., Chen, H., Xing, P., Xu, G., Zhang, T.: Iron: Private inference on transformers. Advances in Neural Information Processing Systems **35**, 15718–15731 (2022) [4](#), [10](#), [22](#)
16. Hasler, S., Krips, T., Küsters, R., Reisert, P., Rivinius, M.: Overdrive lowgear 2.0: Reduced-bandwidth mpc without sacrifice. Cryptology ePrint Archive (2023) [4](#)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016) [12](#)
18. Huang, Z., Lu, W.j., Hong, C., Ding, J.: Cheetah: Lean and fast secure {two-party} deep neural network inference. In: 31st USENIX Security Symposium (USENIX Security 22). pp. 809–826 (2022) [4](#), [10](#), [12](#), [23](#)
19. Hussain, S.U., Javaheripi, M., Samragh, M., Koushanfar, F.: Coinn: Crypto/ml codesign for oblivious inference via neural networks. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. pp. 3266–3281 (2021) [2](#), [4](#), [5](#), [22](#), [24](#)
20. Jha, N.K., Ghodsi, Z., Garg, S., Reagen, B.: Deepreduce: Relu reduction for fast private inference. In: International Conference on Machine Learning. pp. 4839–4849. PMLR (2021) [2](#), [3](#), [4](#), [5](#), [20](#), [22](#), [24](#), [25](#)
21. Jha, N.K., Reagen, B.: Deepreshape: Redesigning neural networks for efficient private inference. arXiv preprint arXiv:2304.10593 (2023) [5](#), [12](#)
22. Jia, L., Liang, Y., Li, X., Lu, L., Yan, S.: Enabling efficient fast convolution algorithms on gpus via megakernels. IEEE Transactions on Computers **69**(7), 986–997 (2020) [22](#)
23. Jordà, M., Valero-Lara, P., Pena, A.J.: cuconv: Cuda implementation of convolution for cnn inference. Cluster Computing **25**(2), 1459–1473 (2022) [22](#)
24. Kala, S., Jose, B.R., Mathew, J., Nalesh, S.: High-performance cnn accelerator on fpga using unified winograd-gemm architecture. IEEE Transactions on Very Large Scale Integration (VLSI) Systems **27**(12), 2816–2828 (2019) [22](#)
25. Kepuska, V., Bohouta, G.: Next-generation of virtual personal assistants (microsoft cortana, apple siri, amazon alexa and google home). In: 2018 IEEE 8th annual computing and communication workshop and conference (CCWC). pp. 99–103. IEEE (2018) [1](#)
26. Knott, B., Venkataraman, S., Hannun, A., Sengupta, S., Ibrahim, M., van der Maaten, L.: Crypten: Secure multi-party computation meets machine learning. Advances in Neural Information Processing Systems **34**, 4961–4973 (2021) [4](#), [22](#)

27. Krieger, F., Hirner, F., Mert, A.C., Roy, S.S.: Aloha-he: A low-area hardware accelerator for client-side operations in homomorphic encryption. Cryptology ePrint Archive (2023) [4](#)
28. Krishnamoorthi, R.: Quantizing deep convolutional networks for efficient inference: A whitepaper. arXiv preprint arXiv:1806.08342 (2018) [21](#), [23](#)
29. Kundu, S., Lu, S., Zhang, Y., Liu, J., Beerel, P.A.: Learning to linearize deep neural networks for secure and efficient private inference. arXiv preprint arXiv:2301.09254 (2023) [3](#), [4](#), [5](#), [12](#), [20](#), [22](#), [24](#)
30. Lavin, A., Gray, S.: Fast algorithms for convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4013–4021 (2016) [6](#), [10](#), [21](#), [22](#)
31. Lee, S., Kim, D.: Deep learning based recommender system using cross convolutional filters. Information Sciences **592**, 112–122 (2022) [1](#)
32. Li, D., Shao, R., Wang, H., Guo, H., Xing, E.P., Zhang, H.: Mpcformer: fast, performant and private transformer inference with mpc. arXiv preprint arXiv:2211.01452 (2022) [4](#), [22](#)
33. Li, G., Jia, Z., Feng, X., Wang, Y.: Lowino: Towards efficient low-precision winograd convolutions on modern cpus. In: Proceedings of the 50th International Conference on Parallel Processing. pp. 1–11 (2021) [23](#)
34. Li, G., Liu, L., Wang, X., Ma, X., Feng, X.: Lance: efficient low-precision quantized winograd convolution for neural networks based on graphics processing units. In: ICASSP 2020–2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 3842–3846. IEEE (2020) [23](#)
35. Liu, J., Juuti, M., Lu, Y., Asokan, N.: Oblivious neural network predictions via minionn transformations. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security. pp. 619–631 (2017) [2](#), [12](#), [24](#)
36. Liu, X., Pool, J., Han, S., Dally, W.J.: Efficient sparse-winograd convolutional neural networks. arXiv preprint arXiv:1802.06367 (2018) [22](#)
37. Lou, Q., Shen, Y., Jin, H., Jiang, L.: Safenet: A secure, accurate and fast neural network inference. In: International Conference on Learning Representations (2020) [3](#), [4](#), [12](#), [22](#), [24](#)
38. jie Lu, W., Huang, Z., Zhang, Q., Wang, Y., Hong, C.: Squirrel: A scalable secure Two-Party computation framework for training gradient boosting decision tree. In: 32nd USENIX Security Symposium (USENIX Security 23). pp. 6435–6451. Anaheim, CA (2023) [2](#)
39. Marques, G., Agarwal, D., De la Torre Díez, I.: Automated medical diagnosis of covid-19 through efficientnet convolutional neural network. Applied soft computing **96**, 106691 (2020) [1](#)
40. Mishra, P., Lehmkuhl, R., Srinivasan, A., Zheng, W., Popa, R.A.: Delphi: A cryptographic inference system for neural networks. In: Proceedings of the 2020 Workshop on Privacy-Preserving Machine Learning in Practice. pp. 27–30 (2020) [2](#), [3](#), [4](#), [20](#), [22](#), [25](#)
41. Mohassel, P., Rindal, P.: Aby3: A mixed protocol framework for machine learning. In: Proceedings of the 2018 ACM SIGSAC conference on computer and communications security. pp. 35–52 (2018) [2](#), [4](#), [22](#)
42. Mohassel, P., Zhang, Y.: Secureml: A system for scalable privacy-preserving machine learning. In: 2017 IEEE symposium on security and privacy (SP). pp. 19–38. IEEE (2017) [2](#), [3](#), [4](#), [20](#), [22](#)
43. Pang, Q., Zhu, J., Möllering, H., Zheng, W., Schneider, T.: Bolt: Privacy-preserving, accurate and efficient inference for transformers. Cryptology ePrint Archive (2023) [4](#)

44. Peng, H., Ran, R., Luo, Y., Zhao, J., Huang, S., Thorat, K., Geng, T., Wang, C., Xu, X., Wen, W., et al.: Lingcn: Structural linearized graph convolutional network for homomorphically encrypted inference. arXiv preprint arXiv:2309.14331 (2023) [22](#)
45. Ran, R., Wang, W., Gang, Q., Yin, J., Xu, N., Wen, W.: Cryptogcn: Fast and scalable homomorphically encrypted graph convolutional network inference. Advances in Neural Information Processing Systems **35**, 37676–37689 (2022) [22](#)
46. Rathee, D., Bhattacharya, A., Sharma, R., Gupta, D., Chandran, N., Rastogi, A.: Secfloat: Accurate floating-point meets secure 2-party computation. In: 2022 IEEE Symposium on Security and Privacy (SP). pp. 576–595. IEEE (2022) [2, 4, 22](#)
47. Rathee, D., Rathee, M., Goli, R.K.K., Gupta, D., Sharma, R., Chandran, N., Rastogi, A.: Sirnn: A math library for secure rnn inference. In: 2021 IEEE Symposium on Security and Privacy (SP). pp. 1003–1020. IEEE (2021) [2, 3, 4, 5, 8, 9, 10, 12, 20, 22, 24](#)
48. Rathee, D., Rathee, M., Kumar, N., Chandran, N., Gupta, D., Rastogi, A., Sharma, R.: Cryptflow2: Practical 2-party secure inference. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. pp. 325–342 (2020) [2, 3, 4, 12, 20, 22, 23, 24](#)
49. Riazi, M.S., Samragh, M., Chen, H., Laine, K., Lauter, K., Koushanfar, F.: {XONN}:{XNOR-based} oblivious deep neural network inference. In: 28th USENIX Security Symposium (USENIX Security 19). pp. 1501–1518 (2019) [2, 4, 5, 22](#)
50. Richens, J.G., Lee, C.M., Johri, S.: Improving the accuracy of medical diagnosis with causal machine learning. Nature communications **11**(1), 3923 (2020) [1](#)
51. Samragh, M., Hussain, S., Zhang, X., Huang, K., Koushanfar, F.: On the application of binary neural networks in oblivious inference. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4630–4639 (2021) [4, 22](#)
52. Sanderson, C., Curtin, R.: Armadillo: a template-based c++ library for linear algebra. Journal of Open Source Software **1**(2), 26 (2016) [23](#)
53. Shen, L., Dong, Y., Fang, B., Shi, J., Wang, X., Pan, S., Shi, R.: Abnn2: Secure two-party arbitrary-bitwidth quantized neural network predictions. In: Proceedings of the 59th ACM/IEEE Design Automation Conference. pp. 361–366 (2022) [2, 3, 4, 5, 22](#)
54. Wang, X., Wang, C., Cao, J., Gong, L., Zhou, X.: Winonn: Optimizing fpga-based convolutional neural network accelerators using sparse winograd algorithm. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **39**(11), 4290–4302 (2020) [22](#)
55. Wu, B., Wang, Y., Zhang, P., Tian, Y., Vajda, P., Keutzer, K.: Mixed precision quantization of convnets via differentiable neural architecture search. arXiv preprint arXiv:1812.00090 (2018) [8, 23](#)
56. Xu, T., Li, M., Wang, R., Huang, R.: Falcon: Accelerating homomorphically encrypted convolutions for efficient private mobile network inference. In: 2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD). pp. 1–9. IEEE (2023) [4](#)
57. Yan, D., Wang, W., Chu, X.: Optimizing batched winograd convolution on gpus. In: Proceedings of the 25th ACM SIGPLAN symposium on principles and practice of parallel programming. pp. 32–44 (2020) [22](#)
58. Yang, H., Duan, L., Chen, Y., Li, H.: Bsq: Exploring bit-level sparsity for mixed-precision neural network quantization. arXiv preprint arXiv:2102.10462 (2021) [12](#)

59. Yang, L., Jin, Q.: Fracbits: Mixed precision quantization via fractional bit-widths. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 10612–10620 (2021) [8](#), [23](#)
60. Yao, Z., Dong, Z., Zheng, Z., Gholami, A., Yu, J., Tan, E., Wang, L., Huang, Q., Wang, Y., Mahoney, M., et al.: Hawq-v3: Dyadic neural network quantization. In: International Conference on Machine Learning. pp. 11875–11886. PMLR (2021) [10](#), [11](#)
61. Zeng, W., Li, M., Xiong, W., Tong, T., Lu, W.j., Tan, J., Wang, R., Huang, R.: Mpcvit: Searching for accurate and efficient mpc-friendly vision transformer with heterogeneous attention. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5052–5063 (2023) [4](#), [22](#)
62. Zeng, W., Li, M., Yang, H., jie Lu, W., Wang, R., Huang, R.: Copriv: Network/protocol co-optimization for communication-efficient private inference. In: Thirty-seventh Conference on Neural Information Processing Systems (2023), <https://openreview.net/forum?id=qVeDwgYsho> [2](#), [4](#), [6](#), [12](#), [21](#), [22](#), [24](#)

**Table 4:** Notations and the corresponding descriptions used in this paper.

Notation	Description
$\lambda$	Security parameter
$\gg$	Shift right
$l, s$	Bit width, scale
$l_w, l_a, l_{acc}, l_{res}, l_{add}$	Bit width of weight, activation, accumulation, residual, and addition
$l_{feature\_ext}, l_{out\_ext}$	Bit width of Winograd feature and output transformation
$x^{(l)}, \langle x \rangle^{(l)}$	An $l$ -bit integer $x$ and $l$ -bit secret shares
$H, W, C, K$	Height and width of output feature, and number of input and output channel
$A, B, G$	Winograd transformation matrices
$m, r, T$	Size of output tile and convolution weight, and number of tiles
WmAn	$m$ -bit weight and $n$ -bit activation

## A Threat Model

EQO is a 2PC-based private inference framework that involves a client and a server. The server holds the private model weights and the client holds the private input data. Following [6, 20, 29, 40, 42, 47, 48], we assume the DNN architecture (including the number of layers and the operator type, shape, and bit widths) are known by two parties. At the end of the protocol execution, the client learns the inference result and the two parties know nothing else about each other’s input. Following prior work, we assume the server and client are semi-honest adversaries. Specifically, both parties follow the protocol specifications but also attempt to learn more from the information than allowed. We assume no trusted third party exists so the helper data needs to be generated by the client and server [40, 47, 48].

## B Notations and Underlying Protocols

Notations used in this paper is shown in Table 4. We also show the detailed communication complexity with and without MSB-known optimization of the underlying protocols in Table 5.

## C Supplementary Preliminaries

### C.1 Arithmetic Secret Sharing

Arithmetic secret sharing (ArSS) is a fundamental cryptographic scheme used in our framework. Specifically, an  $l$ -bit value  $x$  is additively shared in the integer ring  $\mathbb{Z}_{2^l}$  as the sum of two values, e.g.,  $\langle x \rangle_s^{(l)}$  held by the server and  $\langle x \rangle_c^{(l)}$  held by the client.  $x$  can be reconstructed as  $\langle x \rangle_s^{(l)} + \langle x \rangle_c^{(l)} \bmod 2^l$ . With ArSS, additions can be conducted locally by the server and client while multiplication requires helper data, which are independent of the secret shares and are generated through communication during the pre-processing stage [40, 47, 48].

## C.2 Winograd Convolution

Winograd convolution [30] is an efficient convolution algorithm that minimizes the number of multiplications when computing a convolution. The 2D Winograd transformation  $F(m \times m, r \times r)$ , where the sizes of the output tile, weight, and input tile are  $m \times m$ ,  $r \times r$ , and  $n \times n$ , respectively, with  $n = m + r - 1$ , can be formulated as follows:

$$Y = W \circledast X = A^\top [(G W G^\top) \odot (B^\top X B)] A,$$

where  $\circledast$  denotes a regular convolution and  $\odot$  denotes element-wise matrix multiplication (EWMM).  $A$ ,  $B$ , and  $G$  are transformation matrices that are independent of the weight  $W$  and activation  $X$  and can be computed based on  $m$  and  $r$  [2, 30]. When computing the Winograd convolution,  $W$  and  $X$  are first transformed to the Winograd domain before computing EWMM. The number of multiplications can be reduced from  $m^2r^2$  to  $(m + r - 1)^2$  at the cost of many more additions introduced by the transformations of  $B^\top X B$  and  $A^\top [\cdot] A$ . [62] further formulates the EWMM into the form of general matrix multiplication for better communication efficiency. As a consequence, the communication complexity is reduced from  $O(\lambda C K T (m + r - 1)^2)$  to  $O((m + r - 1)^2 C T (\lambda + K))$ , where  $C$ ,  $K$ , and  $T$  denote the number of input channels, output channels, and the number of tiles, respectively.

## C.3 Network Quantization

Quantization converts a floating-point number into integers for a better training and inference efficiency [28]. 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  $x_q/s_x$ , 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, e.g., residual connection of two quantized numbers  $x_q$  and  $y_q$ , directly adding them together leads to incorrect results. Instead, the scales and the bit widths of  $x_q$  and  $y_q$  need to be aligned first.

## D Related Works

**Efficient Private Inference** In recent years, there has been an increasing amount of literature on efficient private inference, including protocol optimiza-

**Table 5:** Communication complexity of underlying protocols.

Protocol	Communication w/o MSB Optimization	Communication w/ MSB Optimization
$\Pi_{\text{Ext}}^{l_1, l_2}(\langle x \rangle^{(l_1)})$	$O(\lambda(l_1 + 1) + 13l_1 + l_2)$	$O(2\lambda - l_1 + l_2 + 2)$
$\Pi_{\text{Trunc}}^{l_1, l_2}(\langle x \rangle^{(l_1)})$	$O(\lambda(l_1 + 3) + 15l_1 + l_2 + 20)$	$O(3\lambda + l_1 + l_2 + 20)$
$\Pi_{\text{TR}}^{l_1, l_2}(\langle x \rangle^{(l_1)})$	$O(\lambda(l_2 + 1) + 13l_2 + l_1)$	$O(\lambda + 2)$

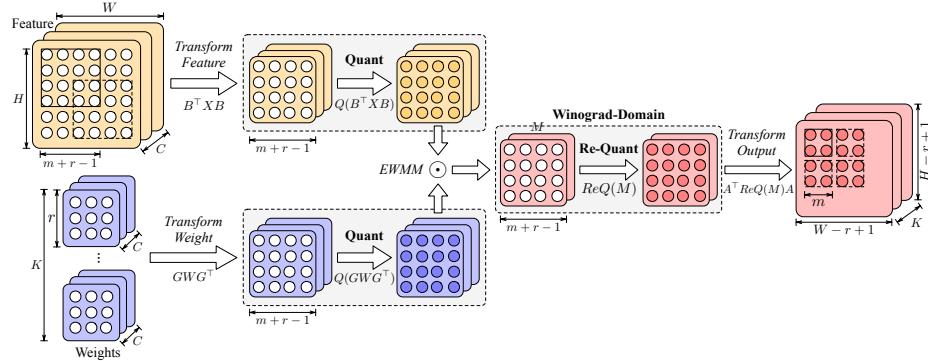
tion [7, 15, 26, 41, 42, 46–48], network optimization [6, 20, 29, 32, 37, 61] and co-optimization [19, 40, 44, 45, 62]. These works are designed for convolutional neural networks (CNNs), Transformer-based models and graph neural networks (GNNs). In this paper, we mainly focus on efficient quantized private inference. XONN [49] combines binary neural networks with Yao’s garbled circuits (GC) to replace the costly multiplications with XNOR operations. As an extension of XONN, [51] proposes a hybrid approach where the 2PC protocol is customized to each layer. CrypTFlow2 [48] supports efficient private inference with uniform bit widths of both weight and activation. SiRNN [47] proposes a series of protocols to support bit extension and truncation, which enable mixed-precision private inference for recurrent neural networks (RNNs). ABNN2 [53] utilizes the advantages of quantized network and realizes arbitrary-bitwidth quantized private inference by proposing efficient matrix multiplication protocol based on 1-out-of-N OT extension. COINN [19] proposes low-bit quantization for efficient ciphertext computation, and has 1-bit truncation error.

**Winograd Algorithm** Many research efforts have been made to Winograd algorithm in hardware domain [22–24, 54, 57]. [30] first applies Winograd algorithm to CNNs, and shows 2.25~4× efficiency improvement. [12] applies Winograd convolution to 8-bit quantized networks, and adds transformation matrices to the set of learnable parameters. [12] also proposes a Winograd-aware neural architecture search (NAS) algorithm to select among different tile size, achieving a better accuracy-latency trade-off. [36] exploits the sparsity of Winograd-based CNNs, and propose two modifications to avoid the loss of sparsity. For Winograd quantization, [5] proposes to balance the ranges of input channels on the forward pass by scaling the input tensor using balancing coefficients. To alleviate the numerical issues of using larger tiles, [3] proposes a tap-wise quantization method. [13] integrates Winograd convolution in CrypTFlow2 [48], but only uses 60-bit fixed-point number, leading to significant communication overhead. Therefore, quantized Winograd-based networks for privacy inference have not been well studied.

## E Details of Experimental Setups

**Private Inference** EQO is implemented based on the framework of SiRNN [47] in the EzPC<sup>3</sup> library for private network inference. Specifically, our quantized Winograd convolution is implemented in C++ with Eigen and Armadillo matrix

<sup>3</sup> <https://github.com/mpc-msri/EzPC>



**Fig. 15:** Overview of quantization procedure of Winograd convolution.

calculation library [52]. Following [18, 48], we use WAN and LAN modes for 2PC communication. Specifically, the bandwidth between is 377 MBps and 40 MBps, and the echo latency is 0.3ms and 80ms in LAN and WAN mode, respectively. In our experiments, we evaluate the inference efficiency on the Intel Xeon Gold 5220R CPU @ 2.20GHz.

**Network Quantization** To improve the performance of quantized networks, we adopt quantization-aware training with PyTorch framework for the Winograd-based networks. During training, we fix the transformation matrices  $A, B$  while  $G$  is set to be trainable since  $GWG^\top$  is computed and quantized before inference (processed offline). For each network, we fix the bit width of the first and last layer to 8-bit. For MiniONN, we fix the bit width of activation to 4-bit while for ResNets, we fix it to 6-bit. As we have introduced, we always use high precision, i.e., 8 bits, for the residual for better accuracy [55, 59]. The benefit of high-precision residual is evaluated in Appendix F. We illustrate the quantization procedure for Winograd convolution with  $F(2 \times 2, 3 \times 3)$  transformation as shown in Figure 15. Following [12, 28, 33, 34], we use fake quantization to both the activation and weights and then re-quantize the activation back after GEMM.

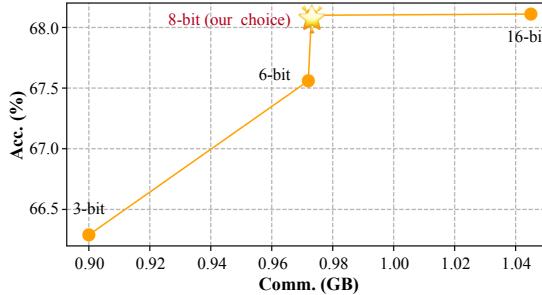
**Baselines** We describe the detailed information of each baseline we use in Table 6, including their networks, datasets, and protocols.

## F Benefit of High-precision Residual

From Figure 16, we fix the bit width of activation and weight to 3-bit and 4-bit, we observe that the accuracy of ResNet-32 significantly improves by 1.8% and the communication only increases slightly when we increase the residual bit width from 3-bit to 8-bit. And the benefit of 16-bit residual is not significant. Hence, we propose to use 8-bit high-precision residual to train our quantized networks for a better performance.

**Table 6:** Networks, datasets, and protocols used in baselines.

Method	Network (Dataset)	Protocol
	Protocol Frameworks	
CrypTFlow2 [48]	MiniONN (CIFAR-10), ResNet-32 (CIFAR-100), ResNet-18 (Tiny-ImageNet, ImageNet)	OT
SiRNN [47]	MiniONN (CIFAR-10), ResNet-32 (CIFAR-100), ResNet-18 (Tiny-ImageNet, ImageNet)	OT
CoPriv [62]	MiniONN (CIFAR-10), ResNet-32 (CIFAR-100), ResNet-18 (Tiny-ImageNet, ImageNet)	OT
COINN [19]	MiniONN (CIFAR-10), ResNet-32 (CIFAR-100)	OT
MiniONN [35]	MiniONN (CIFAR-10)	OT
ReLU-optimized Methods		
DeepReDuce [20]	ResNet-18 (CIFAR-100, Tiny-ImageNet)	OT
SNL [6]	ResNet-18 (CIFAR-100)	OT
SAFENet [37]	ResNet-32 (CIFAR-100)	OT
SENet [29]	ResNet-18 (ImageNet)	OT

**Fig. 16:** The impact of residual bit width on accuracy and communication for ResNet-32 on CIFAR-100.

## G Online and Total Communication Comparison with ReLU-optimized Methods

In Table 7, we separately compare the online and total communication with ReLU-optimized methods. The result shows that EQO focuses reducing the dominant convolution communication, such that achieves significantly lower total communication with even higher accuracy. In terms of online communication, although EQO does not focus on directly removing online components, e.g., ReLUs, EQO still achieves low online communication with our proposed graph-level optimizations.

**Table 7:** Comparison of online and total communication on the CIFAR-100 dataset.

Method	Online (GB)	Total (GB)	Acc. (%)
DELPHI [40]	0.19	1.90	67.8
DeepReDuce [20]	0.15	42.6	68.4
EQO	0.17	0.47	69.2

## H Analysis of the Extension Communication Complexity of Winograd Transformation

As mentioned in Section B, the communication complexity per element mainly scales with the initial bit width  $l_1$ . Also, for a given matrix with the dimension of  $d_1 \times d_1 \times d_3$ , the number of extensions needed would be  $d_1 \times d_1 \times d_3$ . Therefore, the total communication complexity of extension for a matrix becomes  $O(d_1 d_2 d_3 (\lambda(l_1 + 1) + 13l_1 + l_2))$ . In Figure 8, the extension communication of feature extension (block ②) is  $O(HWC(\lambda(l_1 + 1) + 13l_1 + l_2))$  and the extension communication of output extension (block ④) is  $O(KT(m + r - 1)^2(\lambda(l_1 + 1) + 13l_1 + l_2))$ , which is more expensive for communication.

## I Formal Description of Protocol Fusion

**Proposition 1.** For a given  $\langle x \rangle^{(l_1)}$ ,  $\Pi_{\text{Trunc}}^{l_1, l_2}(\langle x \rangle^{(l_1)})$  can be decomposed into  $\Pi_{\text{TR}}^{l_1, l_2}$  followed by  $\Pi_{\text{Ext}}^{l_1 - l_2, l_1}$  as

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

The decomposition does not change communication.

**Proposition 2.** If a given  $\langle x \rangle^{(l_1)}$  is extended to  $l_2$ -bit first and then extended to  $l_3$ -bit. Then, the two neighboring extensions can be fused together as

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

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

**Proposition 3.** For a given  $\langle x \rangle^{(l_1)}$ , when re-quantization ends up with truncation, and is followed by an extension, the protocol can be first decomposed and then fused as

$$\Pi_{\text{Ext}}^{l_1, l_3}(\Pi_{\text{Trunc}}^{l_1, l_2}(\langle x \rangle^{(l_1)})) = \Pi_{\text{Ext}}^{l_1, l_3}(\Pi_{\text{Ext}}^{l_1 - l_2, l_1}(\Pi_{\text{TR}}^{l_1, l_2}(\langle x \rangle^{(l_1)}))) = \Pi_{\text{Ext}}^{l_1 - l_2, l_3}(\Pi_{\text{TR}}^{l_1, l_2}(\langle x \rangle^{(l_1)})).$$

Combining Proposition 1 and 2, this fusion reduces communication by around  $2\times$  from  $O(\lambda(2l_1 + 4))$  to  $O(\lambda(l_1 + 2))$ .