ABNN^2: Secure Two-party Arbitrary-Bitwidth Quantized Neural Network Predictions

Liyan Shen[1], Ye Dong[2], Binxing Fang[3], Jinqiao Shi[1], Xuebin Wang[2], Shengli Pan[1], Ruisheng Shi[1]
[1]Key Laboratory of Trustworthy Distributed Computing and Service (BUPT), Ministry of Education, Beijing University of Posts and Telecommunications, Beijing, China
[2]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[3]Institute of Electronic and Information Engineering of UESTC in Guangdong, Dongguan, China
{shenliyan,shijinqiao,psl,shiruisheng@bupt.edu.cn},{dongye,wangxuebin}@iie.ac.cn,fangbx@cae.cn

ABSTRACT

Data privacy and security issues are preventing a lot of potential on-cloud machine learning as services from happening. In the recent past, secure multi-party computation (MPC) has been used to achieve the secure neural network predictions, guaranteeing the privacy of data. However, the cost of the existing two-party solutions is expensive and they are impractical in real-world setting.

In this work, we utilize the advantages of quantized neural network (QNN) and MPC to present ABNN^2, a practical secure two-party framework that can realize arbitrary-bitwidth quantized neural network predictions. Concretely, we propose an efficient and novel matrix multiplication protocol based on 1-out-of-N OT extension and optimize the the protocol through a parallel scheme. In addition, we design optimized protocol for the ReLU function. The experiments demonstrate that our protocols are about 2×-36× and $1.4\times - 7\times$ faster than SecureML (S&P'17) and MiniONN (CCS'17) respectively. And ABNN^2 obtain comparable efficiency as state of the art QNN prediction protocol QUOTIENT (CCS'19), but the later only supports ternary neural network.

KEYWORDS

Secure quantized neural network prediction, 1-out-of-N oblivious transfer extension, Secure matrix multiplication

1 INTRODUCTION

As a prevalent business model, many technology companies provide neural network (NN) prediction services for customers in a wide range of applications, such as healthcare and finance. However, the data involved in this task is usually sensitive and confidential, which greatly arouses customers' concerns about data privacy. Besides, due to the restrictive laws and regulations, such as HIPAA and GDPR, there will be higher requirements for machine learning as a service (MLaaS). The goal of MLaaS is to utilize data while protecting data privacy.

There have been naive solutions to the NN predictions, either consumers acquire the model and perform the inference task on locally trusted platforms or the service provider acquires consumers' plaintext data, both of the two solutions are prohibited [15]. Recent advances in cryptography research provide many tools like MPC and fully homomorphic encryption to address these concerns. They can ensure that during the inference, neither the customers' private data nor the service providers' models will be revealed. However, there still exists efficiency issues for the secure two-party protocols to use in real-world applications.

Recent research in the area of machine learning illustrates that the quantization technique provides a much more efficient solution for performing neural network prediction tasks practically and securely [1, 12]. More precisely, quantization reduces the size of neural networks by lowering the precision of the values involved. It is more suitable for resource-constrained devices like mobile phones or laptops.

1.1 Related work

Privacy-preserving neural network prediction has been an active research area in recent years. Different security protocols such as homomorphic encryption (HE), garbled circuits (GC) and secret sharing are utilized to solve this problem. In the following, we describe the related protocols from two perspectives. The first one is mainly focusing on the design and optimization of cryptographic protocols, and the second one is mainly combining optimization techniques of ML.

Design and optimization of cryptographic protocols: Gilad-Bachrach et al. proposed CryptoNets [5] based on leveled homomorphic encryption (LHE), which only allows a limited number of addition and multiplication operations. Due to the limitations of LHE, the author proposed low-degree polynomial alternatives to the non-linear functions, which will lessen NN prediction accuracy.

Instead of purely relying on HE, most of the work uses multiple secure protocols for better performance. In SecureML [11], the author performed multiplications based on additive secret sharing and offline-generated multiplication triplets (MTs). And they utilized GC for the non-linear activation function which is approximated using a piecewise linear function. MiniONN [9] further optimized the matrix multiplications protocols. It performed LHE operations with the SIMD batch processing technique to complete the linear transformation in offline phase.

Due to the complex cryptographic protocols between the two participants, the overhead of these methods proposed above is relatively large. In order to solve the performance bottleneck, multi-party protocols have been introduced to accelerate the computation. Chameleon [13] generated correlated randomness used in MTs with

DAC '22, July 10–14, 2022, San Francisco, CA, USA

ACM ISBN 978-1-4503-9142-9/22/07…$15.00
https://doi.org/10.1145/3489517.3530680

the help of a third-party, as well the oblivious transfer (OT) used in garbled circuits. Both SecureNN [16], Cryptflow [7] and [15] constructed novel protocols for non-linear functions with the help of third-party that completely avoid the use of GC. Besides, some other constructions based on three-party protocols have been proposed [10, 17], the authors designed efficient core module protocols based on replicated secret sharing scheme. Even though the multi-party protocols could be practical enough in real-world applications, they have a strong security assumption that the parties cannot collude. That is, any two parties colluding with each other will disclose the private information of the other semi-honest participant. Therefore, an efficient and secure two-party prediction protocol is necessary for the clients.

Combining optimization techniques of ML: DeepSecure [14] proposed a novel data projection pre-processing technique to generate lower-dimensional data and model size, and they also proposed technique using the sparse nature of DL models. Both of the two methods further reduce the computation and communication overhead of GC in the execution of DL.

There are some other work combined with quantization neural networks. The training parameters in QNN are usually encoded by floating-point numbers into fixed-point form, and the bitwidth (also called as bit-length) of the fixed-point number is small, such as 8-bit. The motivation of this combination is that the overhead of some commonly used cryptographic tools is usually proportional to the bitwidth of inputs, such as GC, OT. So the overhead of these work is relatively smaller. The author in XONN [12] proposed the secure protocols for binary neural network, in which the weights and activations are restricted to binary (i.e, ±1) values. Thus multiplications are replaced by XNOR operations, which can be completed by the GC circuit. QUOTIENT [1] was proposed to realize the secure computation of the ternary (-1,0,1) neural network. The author converted the ternary multiplication into two binary multiplication, which is completed based on 1-out-of-2 OT protocol. However, arbitrary-bitwidth quantization has been realized in the field of machine learning [2, 4], but not in the field of secure computation. It is mainly focused on quantizing the bit-length of parameters to a fixed value (binary or ternary), just as we described above.

1.2 Contribution and roadmap

In this paper, we are the first work to design and implement a novel and practical two-party framework ABNN^2, which can support the secure quantized neural network predictions with arbitrary-bitwidth. In detail, our contributions are described as follows:

- Compared with the existing two-party protocols, we utilize the advantage of QNN to design specialized protocols for it. For the quantized models with different fixed-point bitwidth weights, ABNN^2 can adaptability realize privacy-preserving QNN predictions. Concretely, the quantized computation with arbitrary-bitwidth can achieve by setting different values of the parameters N and γ in matrix multiplication sub-protocol, which will be given detailed description in section 3.
- We propose a novel and efficient matrix multiplication protocol based on 1-out-of-N OT extension, and propose novel parallel schemes for one-batchsize and multi-batchsize matrix multiplication. Among all possible combinations of protocol parameters N and γ, we give the optimal parameter values for different bitwidth of quantized weights.
- As for the non-linear layer, we first design novel protocol for ReLU, avoiding partial reconstruction operations implemented by GC. And we set the bit-length of the operand in GC to the value ℓ, where all operands are chosen from the ring $\mathbb{Z}_{2^\ell}$. So there will be no extra cost required to complete the non-XOR gates corresponding to the modulo operation.
- We give detailed correctness and proof of security under the semi-honest model. Experiments demonstrate that ABNN^2 can obtain better performance than traditional two-party protocols for QNN, especially the smaller bitwidth of the quantized weight, the more obvious this advantage. Concretely, our protocols outperform SecureML by 2×-36× for multiplication benchmarks. And it is about $1.4\times - 7\times$ faster than that in MiniONN for end-to-end network predictions. And we achieve comparable efficiency as QUOTIENT, but the later only support ternary NN.

The remainder of this paper is organized as follows. In section 2 we give the definition of symbols and primitives. Overview of ABNN^2 is given in section 3. The detailed core module constructions is described in section 4. Then, we give the implementation and benchmarks in section 5, and the conclusion in section 6.

2 PRELIMINARY

2.1 Definitions

$\mathcal{S}, C$	Server and Client resp.
$\overset{c}{\equiv}, \kappa$	computationally indistinguishable and the security parameter resp.
$x \in_R D$	uniformly random sampling from a probability distribution D
η, γ	the bitwidth and number of fragments of quantized weights resp.
$\boldsymbol{a}, \boldsymbol{A}$	lowercase and uppercase bold letter denote vector and matrix resp.
$\boldsymbol{a}[i], w[i]$	denote the ith element of $\boldsymbol{a}$ and the ith fragment of w resp.
$\odot, \oplus$	bitwise-AND and bitwise-XOR between two vectors

2.2 Quantized Neural Networks

Quantization is used to compress the NN in order to speed up the inference and reduce computational complexity. It is particularly relevant for clients with low-power infrastructure. During training, the goal is to quantize weights, activations or gradients. While during inference, the quantization scheme for activations is usually deterministic and the weights already been quantized in the training phase. The most common is to quantize values to fixed-point form, such as binary (1-bit), ternary (2-bit) and INT4/INT8 (4/8bit) schemes. It has been shown that low precision and mixed-precision quantization with INT4/INT8 works in practice [4]. In this paper, we mainly consider quantization scenario for weights with arbitrary-bitwidth. Activations will be in float-point form and be encode as fixed-point to utilize the cryptographic protocol, it is different from XONN and more complex. The structure of QNN is that many layers stacked on top of each other, and it consists of linear and non-linear layers. Concretely, the fully connected (FC) layer used in many NN can be formulated as $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}+\boldsymbol{b}$. Operations in the non-linear layer are usually comparison, exponent and so on. And the most commonly used is ReLU: $f(\boldsymbol{y}) = [max(0, y_i)]$.

2.3 Cryptographic Background

In this section, we give the cryptographic preliminaries.

Oblivious Transfer: OT is a basic building block in MPC. In a 1-out-of-2 OT protocol ($\binom{2}{1}$-OT), a sender S holds a pair of message (x_0, x_1) and a receiver R has a selection bit b, at the end of the protocol, R obtains x_b without learning anything about x_{1-b} or revealing b to S. The base $\binom{2}{1}$-OT can be implemented using the public-key operations. And the OT extension minimizes this cost by allowing S and R to perform m OTs at the cost of a small number of base OTs and $O(m)$ fast symmetric-key operations. The researchers further proposed the $\binom{N}{1}$-OT extension protocol [6] which is considered efficient enough to use in practice. Fig 1 describes the ideal functionality.

Functionality $\mathcal{F}\binom{N}{1}$-OT

The functionality works with two parties, one is the sender S and the other is the receiver R, as follows:

- On input $(SEND, sid, \{x_0, ..., x_{N-1}\})$ from S and $(SELECT, sid, b)$ from R.
- Return $(RECV, sid, x_b)$ to R.

Figure 1: The Ideal Functionality of 1-out-of-N OT.

Secret Sharing: We employ two sharing schemes: Arithmetic sharing and Yao sharing. We briefly review these schemes but refer the reader to [3] for more details. For Arithmetic sharing, a value is shared additively between two parties P_0 and P_1 such that the sum of two shares yields the true value. Suppose the value x is shared additively in ring $\mathbb{Z}_{2^\ell}$ (integers modulo 2^ℓ), the two shares of x are denoted as $\langle x\rangle_0^A$ and $\langle x\rangle_1^A$.

- The algorithm `Share`(x) generates the two shares of x over $\mathbb{Z}_{2^\ell}$ by choosing $r \in_R \mathbb{Z}_{2^\ell}$ and sets $\langle x\rangle_0^A = r$, $\langle x\rangle_1^A = x - r \bmod 2^\ell$.
- The algorithm `Reconst`$(\langle x\rangle_0^A, \langle x\rangle_1^A)$ reconstructs the value $x = \langle x\rangle_0^A + \langle x\rangle_1^A \bmod 2^\ell$ using two shares.

Given two shared values, it is easy to add the shares by P_i locally computing $\langle z\rangle_i = \langle x\rangle_i + \langle y\rangle_i$. To multiply two shared values, classical methods mainly rely on the Beaver's pre-computed MTs. As analyzed in [13], if one operand x is held in cleartext by one party and the other operand y is shared among two parties, the overhead for multiplication will be at least half of the method based on Beaver's MTs. Also, we employ GC protocol as operating on Yao sharing of inputs in the non-linear layer.

Garbled Circuit: It is a cryptographic protocol to jointly compute the function $f(x, y)$ between two parties while keeping the privacy of the inputs. The function is described as a boolean circuit consisting of AND, XOR, and other gates. The XOR operation can be evaluated locally using the Free-XOR technique, the overhead of the garbled circuit mainly comes from the non-XOR gates.

Semi-Honest Security: A semi-honest adversary is the one who corrupts parties but follows the protocol specification. That is, the corrupt parties run the protocol honestly but try to learn additional information. Let π be a two-party protocol which computes a function f. In the semi-honest model, a protocol π is secure, it must be possible that the ideal world adversary's view is indistinguishable from the real world adversary's view.

3 ABNN^2 OVERVIEW

In ABNN^2, all values are represented in fixed-point and in secret-shared fashion between the server and client. $\mathcal{S}$ and C run the interactive secure protocol for each linear and non-linear layer. Suppose the QNN architecture is defined as: $\boldsymbol{y} = \boldsymbol{W}^{l-1} \cdot f_{l-2}(...f_0(\boldsymbol{W}^0 \cdot \boldsymbol{x})...)$ with quantized weights which can be naturally represented as fixed-point form, and C owns sample $\boldsymbol{x}$ to be predicted, $\mathcal{S}$ owns the model $\boldsymbol{W}^i$. The corresponding computation process is presented in Fig 2.

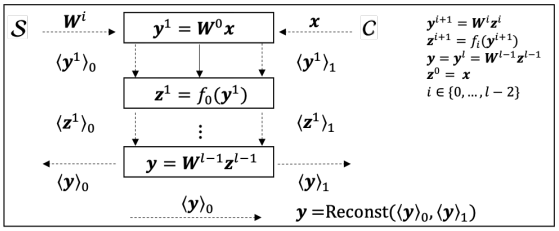

Figure 2: The flow chart of the QNN predictions.

The main overhead of the linear layer is to perform *matrix multiplication* operations. As shown in Fig 2, the first layer is to compute $\boldsymbol{W}^0\boldsymbol{x}$, which can be converted to the *dot-product* $\boldsymbol{wx} = \boldsymbol{w}(\boldsymbol{x}_0 + \boldsymbol{x}_1) = \boldsymbol{wx}_0 + \boldsymbol{wr}$, suppose $\boldsymbol{x}_1 = \boldsymbol{r}, \boldsymbol{r}[i] \in_R \mathbb{Z}_{2^\ell}$, $\boldsymbol{x}_0 = \boldsymbol{x} - \boldsymbol{r} \bmod \mathbb{Z}_{2^\ell}$. And $\boldsymbol{r}$ is randomly chosen by the clients used as shares to blind the private input $\boldsymbol{x}$. Same as MiniONN, the two-party matrix multiplication of ABNN^2 can be divided into a data-independent offline phase and an online phase that depends on the data of clients.

In the offline phase, ABNN^2 completes the dot-product triplets in the form of $\langle u, v, \boldsymbol{w}\cdot\boldsymbol{r}\rangle$, which satisfies $u+v = \boldsymbol{w}\cdot\boldsymbol{r}$. For *element-wise multiplication* wr, the key insight is that we use N-base representation of w. Concretely, $wr = \sum_{i=0}^{\gamma-1} N^i w[i] \cdot r$, where w be quantized with η bit and γ be the number of fragments each with $\log_2 N$ bit. We design customized protocol based on $\binom{N}{1}$-OT extension and additive secret sharing to solve the secure computation of $N^i w[i] \cdot r$ over the ring $\mathbb{Z}_{2^\ell}$. Our implementations take the advantage of $\binom{N}{1}$-OT for better efficiency. Further, we propose novel parallel schemes for generating one-batchsize and multi-batchsize offline triplets respectively.

Computing the matrix multiplication corresponding to the non-first layer is similar to the above. We generate $\boldsymbol{W}^i\langle \boldsymbol{z}^i\rangle_1^A = \boldsymbol{u}^i + \boldsymbol{v}^i$ in offline, where $\langle \boldsymbol{z}^i\rangle_1^A \in_R \mathbb{Z}_{2^\ell}$, $i \in 1, ..., l-1$ and $\boldsymbol{u}^i$, $\boldsymbol{v}^i$ are owned by server and client respectively.

In the online phase, C first uses algorithm `Share` to generate two arithmetic shares of the input feature vector $\boldsymbol{x}$ as $\langle \boldsymbol{x}\rangle_1^A = \boldsymbol{r}$ and $\langle \boldsymbol{x}\rangle_0^A = \boldsymbol{x} - \langle \boldsymbol{x}\rangle_1^A$ and sends $\langle \boldsymbol{x}\rangle_0^A$ to $\mathcal{S}$. Then for the multiplication operation in the first linear layer, $\mathcal{S}$ and C each obtain one share of $\boldsymbol{y}^1$, where $\langle \boldsymbol{y}^1\rangle_0^A = \boldsymbol{W}^0\langle \boldsymbol{x}\rangle_0^A + \boldsymbol{u}^0$, $\langle \boldsymbol{y}^1\rangle_1^A = \boldsymbol{v}^0$. Similar for other linear layer, $\mathcal{S}$ owns $\langle \boldsymbol{y}^{i+1}\rangle_0^A = \boldsymbol{W}^i\langle \boldsymbol{z}^i\rangle_0^A + \boldsymbol{u}^i$ and C owns $\langle \boldsymbol{y}^{i+1}\rangle_1^A = \boldsymbol{v}^i$, where $\langle \boldsymbol{z}^i\rangle_0^A$ is the output result of $\mathcal{S}$ in the non-linear layer computed by GC protocol. it is clear that

$$\begin{aligned} \boldsymbol{y}^{i+1} &= \langle \boldsymbol{y}^{i+1}\rangle_0^A + \langle \boldsymbol{y}^{i+1}\rangle_1^A = \boldsymbol{W}^i\langle \boldsymbol{z}^i\rangle_0^A + \boldsymbol{u}^i + \boldsymbol{v}^i \\ &= \boldsymbol{W}^i\langle \boldsymbol{z}^i\rangle_0^A + \boldsymbol{W}^i\langle \boldsymbol{z}^i\rangle_1^A = \boldsymbol{W}^i\boldsymbol{z}^i \end{aligned} \tag{1}$$

ABNN^2 performs non-linear activation operations based on GC after the linear layer in the online phase, details depicted in chapter 4.2. The output results are also secretly shared by the two parties and will be used as inputs for the next linear layer. Finally, each layer is performed by the corresponding protocol, $\mathcal{S}$ sends the output share $\langle \boldsymbol{y}\rangle_0$ to C who can reconstruct the final prediction results.

4 ABNN^2 CORE MODULE CONSTRUCTIONS

4.1 Quantized Matrix Multiplication

Assume that the server $\mathcal{S}$ holds quantized model $w \in \mathbb{Z}_{2^\eta}$ and the client C holds a random number $r \in_R \mathbb{Z}_{2^\ell}$, which is independent of the client's actual input x. $\mathcal{S}$ and C want to obtain the arithmetic shared value of the product wr without revealing additional information beyond the output.

4.1.1 1-out-of-N OT-based method. By decomposing the fixed-point parameter w, we can obtain the formula:

$$w \cdot r = \sum_{i=0}^{\gamma-1} N^i w[i] \cdot r,\ \gamma = \lceil \eta / log(N) \rceil \tag{2}$$

It can be seen as N-base representation of w, and uses γ instances of $\binom{N}{1}$-OT to obtain the arithmetic shared value of each $N^i w[i] \cdot r$. The special case is binary representation when N=2, which is adopted in secureML. We adopt $\binom{N}{1}$-OT extension as our underlying building block, where $\mathcal{S}$ acts as the receiver and C acts as the sender.

In the i-th $\binom{N}{1}$-OT, the input messages of C are $jN^i \cdot r - s_i$, $j \in \{0, ..., N-1\}$, where $s_i \in_R \mathbb{Z}_{2^\ell}$, and the choice vector of $\mathcal{S}$ is $w[i] \in \{0, 1, ..., N-1\}$. $\mathcal{S}$ and C obtain $\langle y_i \rangle_0$, $\langle y_i \rangle_1$ resp. as described in Fig 3. After γ OTs, $\mathcal{S}$ adds each result and obtains $\langle y \rangle_0 = \sum_{i=0}^{\gamma-1} \langle y_i \rangle_0 = \sum_{i=0}^{\gamma-1} (N^i w[i] \cdot r - s_i) \mod \mathbb{Z}_{2^\ell}$ and C obtains $\langle y \rangle_1 = \sum_{i=0}^{\gamma-1} \langle y_i \rangle_1 = \sum_{i=0}^{\gamma-1} s_i \mod \mathbb{Z}_{2^\ell}$. The correctness of the protocol is obvious: $\langle y \rangle_0 + \langle y \rangle_1 = \sum_{i=0}^{\gamma-1} N^i w[i] \cdot r = w \cdot r \mod \mathbb{Z}_{2^\ell}$.

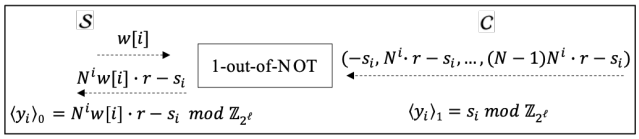

Figure 3: Multiplication protocol of one fragment based on 1-out-of-N OT.

Similarly, $\mathcal{S}$ and C can invoke $n\gamma$ $\binom{N}{1}$-OTs for dot product between two vectors $\boldsymbol{w}$ and $\boldsymbol{r}$. The protocol is described in Algorithm 1. $\boldsymbol{w} \cdot \boldsymbol{r} = \sum_{j=0}^{n-1} \boldsymbol{w}[j]\boldsymbol{r}[j] = \langle y^0 \rangle_0 + \langle y^0 \rangle_1 + ... + \langle y^{n-1} \rangle_0 + \langle y^{n-1} \rangle_1$, where y^j is the output of $\boldsymbol{w}[j]\boldsymbol{r}[j]$. The output of $\mathcal{S}$ and C are $u = \sum_{j=0}^{n-1} \langle y^j \rangle_0$ and $v = \sum_{j=0}^{n-1} \langle y^j \rangle_1$ respectively.

4.1.2 Multi-Batch Optimization. Assume that $\mathcal{S}$ holds quantized matrix $\boldsymbol{W}$ and C holds $\boldsymbol{R}$, where $\boldsymbol{W} \in \mathbb{Z}_{2^\eta}^{m \times n}$, $\boldsymbol{R} \in_R \mathbb{Z}_{2^\ell}^{n \times o}$. For simplicity, we assume the fragment of w_{ij} is $\gamma = 1$. To compute $w_{ij} \cdot r_{jk}$, $k \in \{0, 1, ..., o-1\}$, a simple non-parallel approach requires invoking the solution in 4.1.1 for o times. For fixed i, j, k, C constructs N inputs $\{t \cdot r_{jk} - s_i\}_{t \in \{0,1,...,N-1\}}$ and the outputs of Random Oracle (RO) denoted as H_{it}. $\mathcal{S}$ holds selection w_{ij} and output of RO denoted as $H_{w_{ij}}$. From RO model, $\mathcal{S}$ can only decrypt one of the N messages: $\langle y \rangle_0 = H_{w_{ij}} \oplus (H_{i\tilde{t}} \oplus (\tilde{t} \cdot r_{jk} - s_i)) = w_{ij} \cdot r_{jk} - s_i$, for there exists one $\tilde{t}$ satisfying $w_{ij} = \tilde{t}$ and $H_{w_{ij}} = H_{i\tilde{t}}$.

Instead of repeating the element-wise $\binom{N}{1}$-OT-based method $m \times n \times o$ times, we propose an efficient parallel computing solution. Our key insight is that the choice vector w_{ij} of $\mathcal{S}$ is identical for multiple multiplication. As a result, we can reuse one OT to accomplish o times multiplication $w_{ij} \cdot r_{jk}$ in parallel, where i, j are fixed and $k \in \{0, 1, ..., o-1\}$.

Algorithm 1 Dot-product triplets generation.

Input:
$\mathcal{S}$: $\boldsymbol{w} \in \mathbb{Z}_{2^\eta}^n$, C: $\boldsymbol{r} \in_R \mathbb{Z}_{2^\ell}^n$
Output:
$\mathcal{S}$: $u \in \mathbb{Z}_{2^\ell}$, s.t. $u + v \pmod{\mathbb{Z}_{2^\ell}} = \boldsymbol{w} \cdot \boldsymbol{r}$
C: a random number $v \in \mathbb{Z}_{2^\ell}$
1: **for** $j = 0$ to $n-1$ **do**
2: **for** $i = 0$ to $\gamma - 1$ **do**
3: $\mathcal{S}$ and C perform $\binom{N}{1}$-OT subprotocol, the input of $\mathcal{S}$ is $w[j][i]$ and output is $N^i w[j][i] \cdot r[j] - s_{ij}$, the inputs of C are $N^i t \cdot r[j] - s_{ij}$, $t \in \{0, 1, ..., N-1\}$ and output is s_{ij}.
4: $\mathcal{S}$: $\langle y^j \rangle_0$ += $N^i w[j][i] \cdot r[j] - s_{ij}$; C: $\langle y^j \rangle_1$ += s_{ij}.
5: **end for**
6: $\mathcal{S}$: u += $\langle y^j \rangle_0$; C: v += $\langle y^j \rangle_1$;
7: **end for**

The difference between the non-parallel scheme is that C needs to construct $o \times N$ input messages $\{t \cdot r_{jk} - s_{ik}\}$ by introducing different random numbers s_{ik}, where $k \in \{0, 1, ..., o-1\}$ and $t \in \{0, 1, ..., N-1\}$. Similarly, $\mathcal{S}$ can only decrypt o of the $o \times N$ messages: $\langle \boldsymbol{y} \rangle_0 = H_{w_{ij}} \oplus (H_{i\tilde{t}} \oplus (\tilde{t} \cdot r_{jk} - s_{ik})) = w_{ij} \cdot r_{jk} - s_{ik}$, where $k \in \{0, 1, ..., o-1\}$. The security follows the fact that $\langle \boldsymbol{y} \rangle_0$ is indistinguishable with $\langle \boldsymbol{y'} \rangle_0$ when the non-parallel approach is repeated for o times with different s_i.

4.1.3 One-batch Optimization. The method proposed above does not work for one batchsize matrix multiplication. For this situation, we propose optimization scheme for one-batch matrix multiplication. Roughly, we utilize Correlated-OT (C-OT) to reduce the communication overheads from N to $N-1$ elements as follows.

Recall that in section 4.1.1 for computing $w \cdot r$, C constructs input messages $(-s_i, N^i \cdot r - s_i, 2N^i \cdot r - s_i, ..., (N-1)N^i \cdot r - s_i)$ in the i-th OT. Here, C sets $m_0 = H_{i0} = s_i$ and $m_j = (j \cdot N^i \cdot r - s_i) \oplus H_{ij}$, where $j = \{1, ..., N-1\}$. H_{it} are the outputs of RO for $t \in \{0, 1, ..., N-1\}$, and we take ℓ bit of H_{i0} as the value of s_i. Then C sends $N-1$ messages $\{m_1, m_2, ..., m_{N-1}\}$ to $\mathcal{S}$. On the other side, $\mathcal{S}$ holds $H_{w[i]}$, which is the output of RO. If $w[i] = 0$, $\mathcal{S}$ sets $\langle y_i \rangle_0 = -H_{w[i]} = -H_0 = -s_i$. Otherwise, $\langle y_i \rangle_0 = m_{w[i]} \oplus H_{w[i]}$. Thus, $\mathcal{S}$ obtains $\langle y_i \rangle_0 = N^i w[i] \cdot r - s_i$, and C sets $\langle y_i \rangle_1 = s_i$.

Besides, we also use the packing optimization proposed in SecureML, that is, the output of random oracle can pack multiple multiplications. For matrix multiplication with size $\mathbb{Z}^{m \times n}$ and $\mathbb{Z}^{n \times o}$, the invocations of OT protocol and communication of our method and SecureML are provided in Table 1, the bit output of random oracle is 128, and ℓ is the bit length of the non-quantized elements.

Table 1: The OT Complexity of SecureML and Ours. Comm. is short for communication and M-Batch denotes Multi-Batch.

	SecureML	Ours' M-Batch	Ours' 1-Batch
# OT	$\frac{\ell(\ell+1)}{128} \times mno$	γmn	γmn
Comm.	$mno\ell(\ell+1)(1+\frac{\kappa}{64})$	$\gamma mn(o\ell N + 2\kappa)$	$\gamma mn[\ell(N-1) + 2\kappa]$

It can find out that when the value of N is relatively small, ABNN^2 will have advantages over SecureML, and we set the max value of N to be 16 in our experiment. When we combine with QNN, the value of γ will also be small (e.g. 8 bit quantized weights are decomposed into $\gamma = 4$ fragments, each with 2 bits [2,2,2,2]), it

will also reduce the cost of our solution. By setting different values of N and γ, ABNN^2 can realize the secure matrix multiplication in arbitrary-bitwidth quantization neural network adaptability.

4.2 Protocol for non-linear activation function

For the non-linear activation operation f, $\mathcal{S}$ and C run a protocol that securely implements f in Algorithm 2. Suppose that y_0 and y_1 are the secret shared output of the previous linear transformation layer. $\mathcal{S}$ and C use GC to reconstruct the input information in step 1, and compute the non-linear activation function in step 2. The output of this activation layer be additively shared in step 3, and it can be used as the secret shared input for the next linear layer. Due to the results of linear layer are elements in the ring $\mathbb{Z}_{2^\ell}$, there will be no extra cost required to complete the non-XOR gates corresponding to the modulo operation.

Besides, we also design optimized protocol for the commonly used ReLU function. We first use GC to determine whether z_0 is greater than $-z_1$, if so, then we reconstruct z and reshare it. If not, we only need to reshare zero and output $z_0 = -z_1$. For neurons with negative values, we can avoid the cost of reconstructing elements using the GC protocol.

Algorithm 2 non-linear activation function f.

Input:
$\mathcal{S}$: $y_0 \in \mathbb{Z}_{2^\ell}$, C: $y_1 \in \mathbb{Z}_{2^\ell}$
Output:
$\mathcal{S}$: $z_0 \in \mathbb{Z}_{2^\ell}$, C: a random number $z_1 \in_R \mathbb{Z}_{2^\ell}$, s.t. $z_0 + z_1$ (mod $\mathbb{Z}_{2^\ell}$)=$f((y_0 + y_1) \bmod \mathbb{Z}_{2^\ell})$
1: $\mathcal{S}$ and C use a garbled circuit to reconstruct $y = (y_0 + y_1) \bmod \mathbb{Z}_{2^\ell}$;
2: $\mathcal{S}$ and C use a garbled circuit to compute f interactively;
3: C outputs z_1 and $\mathcal{S}$ outputs $z_0 = f(y) - z_1$;

4.3 Security analysis

ABNN^2 provides security under the semi-honest model. The universal composability framework guarantees the security of arbitrary composition of different protocols. Therefore, we only need to prove the security of individual protocols: (i) the GC execution is secure since it has been proven in [8], (ii) the share type translation directly is secure which follows that of [3], and (iii) the security of our multiplication protocol, and we'll prove it using the real-ideal paradigm. *Security for the corrupted server*: The real view of $\mathcal{S}$ is $H_{it} \oplus \{w_{ij} \cdot r_{jk} - s_{ik}\}$, $k \in \{0, 1, ..., o-1\}$. Since the underlying 1-out-of-N OT is secure [6] and $\{s_{ik}\}$ are uniformly random chosen from $\mathbb{Z}_{2^\ell}$, every element in $\mathcal{S}$'s view in the real execution is indistinguishable with random number r. *Security for the corrupted client*: As the underlying 1-out-of-N OT is secure, C's real view is distinguishable from its view in the ideal execution. Therefore, ABNN^2 is secure against semi-honest adversary.

5 EXPERIMENT

Our protocol is implemented in C++, and we use the ABY library [3] for OT and GC subprotocols. The experiments are executed on a server, with an Intel Xeon E5-2650 CPU (2.30GHz) and 126GB RAM in two environments, respectively modeling LAN and WAN settings. The network bandwidth and latency are simulated using Linux Traffic Tools command. All protocols have been executed 10 times in a 3-layer deep neural network (Fig 4) over the MNIST dataset, we record the average of the experimental results.

(1) *FC*: input image 28×28, the output: $\mathbb{R}^{128\times1} \leftarrow \mathbb{R}^{128\times784} \cdot \mathbb{R}^{784\times1}$
(2) *ReLU activation*: calculates ReLU for each input.
(3) *FC*: input size 128, the output: $\mathbb{R}^{128\times1} \leftarrow \mathbb{R}^{128\times128} \cdot \mathbb{R}^{128\times1}$
(4) *ReLU activation*: calculates ReLU for each input.
(5) *FC*: input size 128, the output: $\mathbb{R}^{10\times1} \leftarrow \mathbb{R}^{10\times128} \cdot \mathbb{R}^{128\times1}$

Figure 4: The neural network structure.

First, we give the overhead of generating dot-product triplets corresponding to the 3-layer NN in the offline phase. We perform this experiment in the LAN setting and all elements and intermediate results are in the ring $\mathbb{Z}_{2^{32}}$ except the quantized weights. The experimental results are described in Table 2. The tuple (1,...,1) indicates that each fragment of the η bit weight is 1 bit, which corresponds to the case of $N = 2$. The values in the tuple represents the bit length of each fragment from the lowest bit to the highest bit (e.g. $\eta = 3$ and (2,1) indicates that the rightmost 2 bits corresponding the first fragment and the highest bit corresponding the second fragment). And ternary and binary represents quantizing the weights to $\{-1, 0, 1\}$ or $\{0, 1\}$ respectively. From Table 2, we can find that the communication overhead is relatively low when the bitlength of each fragment equals to 2 bit. It has better performance than the 1-out-of-2 OT (corresponding to (1,...,1)) used in most of the two-party protocol. And for multi-batchsize, the larger each predicted batch size, the smaller the amortized cost for each prediction.

Table 2: The overhead of generating dot-product triplets in the offline phase.

η		Run time (s)				Communication (MB)			
		1	32	64	128	1	32	64	128
8	(1,...,1)	2.07	4.93	9.46	15.65	32.42	259.32	489.82	950.82
	(2,2,2,2)	**1.58**	3.29	6.13	9.36	19.52	**244.92**	**475.12**	**936.41**
	(3,3,2)	1.66	**3.21**	**5.42**	**8.40**	**18.47**	298.93	587.06	1163.31
	(4,4)	1.99	3.89	5.54	8.76	20.72	468.21	929.21	1851.21
6	(1,...,1)	1.65	3.76	6.91	11.64	24.32	194.49	367.37	713.12
	(2,2,2)	**1.26**	**2.45**	4.77	7.12	14.87	**183.69**	**356.56**	**702.31**
	(3,3)	1.38	2.46	**4.03**	**6.16**	**13.52**	237.71	468.22	929.21
4	(1,...,1)	1.23	2.67	4.89	8.07	16.22	129.67	244.92	475.42
	(2,2)	**0.97**	**1.79**	3.08	4.68	**9.91**	**122.46**	**237.71**	**468.21**
	(4)	1.19	2.05	**2.99**	**4.49**	10.36	234.11	464.01	925.61
3	(1,...,1)	0.96	2.05	3.69	6.08	12.16	97.25	183.69	356.56
	(2,1)	0.87	1.64	2.86	4.42	9.01	**93.65**	**180.09**	**352.96**
	(3)	**0.86**	**1.38**	**2.23**	**3.19**	**6.76**	118.86	234.11	464.61
ternary		0.59	0.97	1.63	2.39	4.51	46.83	90.05	176.49
binary		0.52	0.89	1.48	2.13	4.06	32.42	61.24	118.86

The microbenchmark results for matrix multiplication protocol is described in Table 3. We consider the WAN setting with 9MB/s and 72ms RTT (round-trip time) and all nonquantized elements in the ring $\mathbb{Z}_{2^{64}}$. We set the same conditions as in SecureML and compare with it. For binary and ternary network, it is about 2×-3× and 25×-36× faster than SecureML in LAN and WAN setting respectively. For 8 bit quantized network, it is about 4×-6× faster in WAN setting. The communication cost is about 25×, 20× and 4× fewer than SecureML for three kinds of networks.

Table 3: Performance of the offline phase for matrix multiplication: $128 \times d$ quantized matrix with an d-dimensional vector.

	d	Our			SecureML
		binary	ternary	8(2,2,2,2)	
LAN(s)	100	0.48	0.52	2.01	0.86
	500	1.51	1.87	8.24	3.8
	1000	2.69	3.24	15.39	7.9
WAN(s)	100	1.67	1.75	9.88	43.2
	500	6.94	8.32	45.65	210.6
	1000	12.74	16.58	75.01	463.2
Comm(MB)	100	7.82	9.38	43.76	190
	500	39.07	46.88	218.76	1GB
	1000	78.13	93.76	437.51	1.9GB

The experimental results for secure two-party prediction are described in Table 4,5. We consider the same WAN setting as QUOTIENT with 24.3MB/s and 40ms RTT. The MiniONN protocol has been implemented on our server. The source code of MiniONN can be obtained from Github[1] and we use their code directly in our comparison experiment. In MiniONN, $\mathcal{S}$ only needs to transfer encryption of W denoted as $Enc(W)$ once for multi-batch predictions, we only statistics the communication overhead of $Enc(W)$ once. From Table 4, we can find that for our protocols have better performance in batchsize=128, it is about 3×-7× and 1.4×-4.5× faster than MiniONN in LAN and WAN setting respectively. The communication cost is about $1.1\times - 4.5\times$ fewer than MiniONN.

Table 4: Performance comparison with MiniONN

		LAN(s)		WAN(s)		Comm(MB)	
Batchsize		1	128	1	128	1	128
MiniONN		1.14	40.05	3.48	125.68	18.1	1621.3
Our $\ell = 32$	4(2,2)	1.42	8.88	3.54	48.18	11.78	707.11
	3(2,1)	1.35	8.43	3.44	41.94	10.88	591.85
	ternary	1.05	5.97	3.03	30.66	6.38	415.37
	binary	1.008	5.93	2.81	27.61	5.93	357.75
Our $\ell = 64$	4(2,2)	1.49	12.85	4.07	90.06	16.36	1408.99
	3(2,1)	1.37	12.38	3.85	76.75	14.56	1178.49
	ternary	1.09	9.84	3.11	54.59	9.16	829.14
	binary	1.06	9.58	2.98	48.41	8.26	713.89

In QUOTIENT, the author proposed secure protocol for ternary network with multi cores parallelization strategy, which will speed up the computation on average by $8 - 15\times$ over LAN and by about $10 - 100\times$ over WAN setting. Even though our experiments are executed with single-core due to hardware limitation, Table 5 indicates that our experimental results are only a little slower than theirs. Thus, we believe that our protocols are more efficient when optimized with multi-cores parallelization.

Table 5: Performance comparison with QUOTIENT

	LAN(s)		WAN(s)		Comm(MB)	
Batchsize	1	128	1	128	1	128
QUOTIENT	0.356	2.24	6.8	8.3	-	-
Our	1.008	3.13	2.44	10.84	4.33	106.06

In all, it's obvious that the overhead of our method is proportional to the bit length of weights, which are usually 1 bit,2 bit, 8bit in QNN. Therefore, for commonly used QNN, the cost of our method for prediction is lower than that using homomorphic encryption or other methods that are independent of the bit-length of elements.

[1]https://github.com/SSGAalto/minionn

6 CONCLUSIONS

In this paper, we utilize the advantages of QNN to present secure and practical two-party protocols for neural network predictions with arbitrary-bitwidth quantization. It has better performance when the bit length of quantized weight is small. We propose a novel matrix multiplication protocol based on 1-out-of N OT and further give optimization schemes. Besides, we design novel protol for ReLU. Finally, experiment results confirm the implementation of ABNN^2 is much faster than the previous two-party work.

ACKNOWLEDGMENT

This work is supported by the Key Research and Development Program for Guangdong Province under grant(No.2019B010137003).

REFERENCES

[1] Nitin Agrawal, Ali Shahin Shamsabadi, Matt J Kusner, and Adrià Gascón. 2019. QUOTIENT: two-party secure neural network training and prediction. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 1231–1247.

[2] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. 2017. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*. 1709–1720.

[3] Daniel Demmler, Thomas Schneider, and Michael Zohner. 2015. ABY-A framework for efficient mixed-protocol secure two-party computation.. In *NDSS*.

[4] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. 2021. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630* (2021).

[5] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. 2016. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*. 201–210.

[6] Vladimir Kolesnikov and Ranjit Kumaresan. 2013. Improved OT extension for transferring short secrets. In *Annual Cryptology Conference*. Springer, 54–70.

[7] Nishant Kumar, Mayank Rathee, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. 2019. Cryptflow: Secure tensorflow inference. *arXiv preprint arXiv:1909.07814* (2019).

[8] Yehuda Lindell and Benny Pinkas. 2009. A proof of security of Yao's protocol for two-party computation. *Journal of cryptology* 22, 2 (2009), 161–188.

[9] Jian Liu, Mika Juuti, Yao Lu, and Nadarajah Asokan. 2017. Oblivious neural network predictions via minionn transformations. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 619–631.

[10] Payman Mohassel and Peter Rindal. 2018. ABY3: A mixed protocol framework for machine learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 35–52.

[11] Payman Mohassel and Yupeng Zhang. 2017. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 19–38.

[12] M Sadegh Riazi, Mohammad Samragh, Hao Chen, Kim Laine, Kristin Lauter, and Farinaz Koushanfar. 2019. {XONN}: XNOR-based Oblivious Deep Neural Network Inference. In *28th USENIX Security Symposium (USENIX Security 19)*. 1501–1518.

[13] M Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M Songhori, Thomas Schneider, and Farinaz Koushanfar. 2018. Chameleon: A hybrid secure computation framework for machine learning applications. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. 707–721.

[14] Bita Darvish Rouhani, M Sadegh Riazi, and Farinaz Koushanfar. 2018. Deepsecure: Scalable provably-secure deep learning. In *Proceedings of the 55th Annual Design Automation Conference*. 1–6.

[15] Liyan Shen, Xiaojun Chen, Jinqiao Shi, Ye Dong, and Binxing Fang. 2020. An Efficient 3-Party Framework for Privacy-Preserving Neural Network Inference. In *European Symposium on Research in Computer Security*. Springer, 419–439.

[16] Sameer Wagh, Divya Gupta, and Nishanth Chandran. 2019. Securenn: 3-party secure computation for neural network training. *Proceedings on Privacy Enhancing Technologies* 2019, 3 (2019), 26–49.

[17] Sameer Wagh, Shruti Tople, Fabrice Benhamouda, Eyal Kushilevitz, Prateek Mittal, and Tal Rabin. 2020. FALCON: Honest-Majority Maliciously Secure Framework for Private Deep Learning. *arXiv preprint arXiv:2004.02229* (2020).