Resilient, Secure, and Private Coded Distributed Convolution Computing for Mobile-Assisted Metaverse

Houming Qiu, Kun Zhu, *Member, IEEE*, Dusit Niyato, *Fellow, IEEE*, and Bin Tang, *Member, IEEE*

***Abstract*—The Metaverse is recognized as the next-generation Internet that provides immersive interaction experiences for users. Convolutional neural networks (CNNs) play a crucial role in providing strong immersive experiences in the Metaverse. However, the Metaverse faces challenges in meeting the escalating demands for computing and storage resources due to the explosive growth of convolution tasks, resulting in severe performance degradation. To tackle these issues, coded distributed computing (CDC) is commonly employed. In this paper, we first propose an efficient and reliable mobile-assisted CDC framework to perform large-scale CNN training tasks for the Metaverse. In this framework, the various mobile devices act as workers contributing their resources to collaborate with each other to complete convolution operation tasks. Furthermore, we design a novel resilient, secure, and private coded convolution (RSPCC) scheme for the proposed framework. The RSPCC scheme achieves several significant performances. First, it substantially reduces computation latency compared to conventional convolution. Second, it efficiently mitigates an adverse impact of straggling workers returning results exceedingly slow. Third, we integrate a verifiable computing approach into the encoding/decoding process to check the correctness of the final computation results. Fourth, the PSPCC scheme considers the existence of colluding workers, providing information-theoretic privacy protection for input data. Finally, experimental results demonstrate that our proposed RSPCC scheme can significantly reduce execution time while ensuring the correctness of computation results within the CDC-based Metaverse framework.**

***Index Terms*—Coded distributed computing, metaverse, convolution, mobile-assisted, resiliency, security, privacy, stragglers.**

Manuscript received 20 June 2023; revised 18 April 2024; accepted 20 June 2024. Date of publication 24 June 2024; date of current version 5 November 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62071230 and Grant 62061146002, in part by the Natural Science Foundation of Jiangsu Province under Grant BK20211567, in part by the National Research Foundation, Singapore, and Infocomm Media Development Authority under its Future Communications Research and Development Programme, Defence Science Organisation (DSO) National Laboratories under the AI Singapore Programme (AISG) under Grant AISG2-RP-2020-019 and Grant FCP-ASTAR-TG-2022-003, and in part by the Singapore Ministry of Education (MOE) Tier 1 (RG87/22). Recommended for acceptance by L. Duan. *(Corresponding author: Kun Zhu.)*

Houming Qiu and Kun Zhu are with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China (e-mail: hmqiu56@nuaa.edu.cn; zhukun@nuaa.edu.cn).

Dusit Niyato is with the College of Computing and Data Science, Nanyang Technological University, Singapore 639798 (e-mail: dniyato@ntu.edu.sg).

Bin Tang is with the College of Computer Science and Software Engineering, Hohai University, Nanjing 211100, China (e-mail: cstb@hhu.edu.cn).

Digital Object Identifier 10.1109/TMC.2024.3418449

I. Introduction

THE Metaverse is propelling a novel digital revolution, combining various emerging technologies such as artificial intelligence [1], virtual reality (VR) [2], augmented reality (AR) [3], graphic rendering [4], and digital twins [5]. It aims to create a fully immersive, decentralized, and interoperable online 3D virtual world, which can be regarded as a mirror image of the physical world. Each Metaverse user can create a digital avatar to enter the Metaverse and engage in diverse virtual activities and interactions, thereby having an immersive experience [6]. Moreover, Metaverse users are allowed to create content and design the virtual world, such as creating and designing a virtual house or art works of art, and share them with any virtual world. Hence, the Metaverse will provide a diverse range of novel digital immersive experiences [7], [8].

Convolutional neural networks (CNNs) play a crucial role in providing strong immersive experiences within the Metaverse, including computing intensive tasks such as virtual environment rendering, virtual character motion capture and expression recognition [9]. These intensive computation tasks are required to be completed quickly to guarantee immersive experiences for Metaverse users. However, a single computing node with limited resources is insufficient to efficiently meet the requirements of extremely-high computing and storage resources for large-scale CNN training tasks in the Metaverse. To address these issues, distributed computing is utilized to deploy and perform large-scale convolution calculation tasks by leveraging the collaboration of multiple resource-limited computing nodes [10]. For a real-world example, AsiaInfo has realized a fully interactive Metaverse business hall based on the distributed computing framework. It provides consumers with efficient and interactive digital business hall services through a "360° immersive experience" and "virtual digital human guidance", allowing users to immerse themselves.[1]

A distributed computing system in a Metaverse typically consists of a Metaverse service provider cell station (MSPCS) and multiple computing nodes. The computing nodes are comprised of mobile devices with limited computing and storage resources. The MSPCS partitions a large-scale computation-intensive task, such as a convolution calculation task, into multiple small subtasks, which are then allocated to mobile devices for parallel

[1]https://baijiahao.baidu.com/s?id=1762217658978649311&wfr=spider&for=pc

computation. The task is completed only when the MSPCS receives the computation results returned by all participating mobile devices. However, distributed computing systems inevitably encounter "straggling workers," which are mobile devices that lag behind others in returning computation results due to limited computing resources and insufficient network bandwidth [6]. Thus, the presence of "straggling workers" can substantially increase the computation latency [11] and lead to a poor immersive experience in the Metaverse. Coded distributed computing (CDC), which combines coding theory with the distributed computing paradigm is utilized to mitigate the influence of "straggling workers" [12]. By using CDC, the MSPCS is able to complete the large-scale computation tasks without waiting for computed results from all mobile devices, which can greatly reduce the execution time. Therefore, CDC is an efficient solution to provide a high-quality immersive experience for the Metaverse.

However, several critical issues still require careful consideration. First, the integration of convolutional operation and coding theory for the Metaverse is a promising and challenging technology. Second, some mobile devices suffering from Byzantine attacks may deliberately return incorrect results to the MSPCS resulting in calculation failures and a degraded immersive experience. Finally, some mobile devices may collude with each other to extract information from the assigned data of users, resulting in privacy leakage [11], [13].

In this paper, we focus on addressing large-scale convolution computing tasks in the Metaverse using a mobile-assisted CDC framework. It is noteworthy that our proposed coding scheme is applicable to extensive Metaverse scenarios, such as vehicular Metaverse [12], industrial Metaverse [14], and healthcare Metaverse [15]. First, we abstract the computation supply-demand relationship between the MSPCS and mobile devices in the Metaverse. Subsequently, we construct a master-worker CDC system. Second, we speed up the convolution calculation process by integrating the convolution operation with our proposed coding scheme. Moreover, we consider the existence of stragglers, colluding, and malicious mobile devices in the CDC system, which have negative impacts on overall immersive experience for the Metaverse. The main contributions of this paper are listed as follows:

- We propose an efficient and reliable mobile-assisted CDC framework to perform large-scale convolution computation tasks, which is a promising solution to achieve a high-quality immersive experience for the Metaverse. Particularly, the proposed coded distributed convolution computing (CDCC) framework can be applied to a variety of Metaverse services and applications.
- We overcome the limitation of the existing coding schemes on distributed matrix multiplication, and explore the problem of coded convolution calculation for the Metaverse. We propose a novel resilient, secure, and private coded convolution (RSPCC) scheme for distributed computing system, which provides resiliency against straggling mobile devices, security against malicious mobile devices, and information-theoretic (IT) privacy against colluding mobile devices.
- The proposed RSPCC scheme provides several benefits. First, it offers tolerance for straggling mobile devices by allowing the MSPCS to recover the final convolution calculation result from partial results obtained from all mobile devices. Second, it can check the correctness of final computation results. Third, the RSPCC scheme provides IT privacy protection for the input datasets against up to T colluding mobile devices.[2]
- We integrate memory-efficient convolution (MEC) algorithm [16] and OpenBLAS (optimized libraries) into the RSPCC scheme to reduce the memory consumption and execution time for completing large-scale convolution operation tasks. The experiment results demonstrate the superior performance of the RSPCC scheme in terms of efficiency and correctness of computation.
- Finally, we provides theoretical proofs to demonstrate the resiliency, security, and privacy of the RSPCC scheme. Additionally, we conduct extensive performance evaluation to analyze the computational complexity of the proposed scheme.

A. Organization

The rest of this paper is organized as follows. Section II discusses the related works. In Section III, we introduce the system model and problem formulation of the CDC-based Metaverse. In Section IV, we describe the proposed RSPCC scheme, and present an example to illustrate the general RSPCC scheme. In Section V, we present the result and complexity analyses. In Section VI, we provide experiments on the RSPCC scheme. Finally, we conclude this paper in Section VII.

Notations: Throughout the paper, we introduce the following notations. We use $\mathcal{N}$ to denote the set $\{1, 2, \ldots, N\}$. The lowercase letters $a, b, c, \ldots$ denote scalars or variables, and uppercase letters $A, B, C, \ldots$ denote system parameters, such as the number of nodes, the recovery threshold, etc. Moreover, boldface lowercase letters $\mathbf{a}, \mathbf{b}, \mathbf{c}, \ldots$ represent vectors, and boldface uppercase letters $\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots$ represent matrices.

II. Related Work

A. Coded Distributed Convolution Computation

The emerging CDC is an effective solution to improve the performances of distributed convolution calculations [17]. In [18], the authors proposed a fast and reliable convolution scheme using CDC for two long vectors in time-sensitive distributed systems, which provides robustness against stragglers. However, the coding scheme designed in [18] does not consider the privacy issues. In [19], the authors investigated the problem of coded vector convolution and proposed a dynamic coded convolution scheme in distributed systems, which achieves high resilience and is equipped with privacy awareness. Similarly, in [20], the authors considered the same problem in distributed computing scenarios using polynomial codes. However, few works have used CDC to solve distributed convolutions problems.

[2]The number of colluding workers T satisfies (31), as given in Theorem 2.

Additionally, the existing works [18], [19], [20] only focus on one-dimensional convolution calculations, i.e., vector convolution. In this work, we focus on two-dimension convolution calculation, which have broader applications such as image classification [21], [22], [23].

Moreover, the existing works [18], [19], [20] only consider the effects of straggling workers and do not address the presence of colluding and malicious workers in the distributed system. The colluding workers refer to some workers collaborate with each other to obtain information from the input data [13], [24], [25], and malicious workers refer to some workers maliciously perturb the final computation result by returning incorrect results to the master [26]. Therefore, the efficiency and reliability performance concerns motivate us to design a new coding scheme for convolution operations while addressing the security-and-privacy caused by colluding and malicious workers.

B. Convolutional Neural Network for Metaverse

Convolutional neural networks are crucial for providing realistic and immersive experiences in the Metaverse. Recently, there are some works [27], [28], [29], [30] that utilize CNNs to enhance the immersive experience in the Metaverse. In [27], the authors proposed a improved image classification architecture based on CNN to enhance the interaction experiences. The computation complexity is reduced by using optimized CWCT mechanism. In [28], a CNN-based architecture with high-resolution guidance was proposed, which can effectively improving spatial quality while preserving spectral information. In [29], the authors focused on investigating the problem of natural and fluent responses by using CNNs in a dialogue system. We note that CNNs have the potential to achieve a more intelligent, natural, and fluent conversational experience in the Metaverse. Moreover, the authors in [30] proposed a novel CNN framework with multi-scale convolutional layers, which improves the performance of extracting emotion information from speech. However, the aforementioned works assumes sufficient computing and storage resources, without considering the impact of network congestion. In fact, with the explosive growth of high-dimensional datasets, the MSPCS is unable to meet the computing and storage requirements essential for effectively executing CNN training tasks on a single device.

Fortunately, there are a large number of mobile devices equipped with computing and storage resources in human lives. These mobile devices can collaborate with each other to complete large-scale CNN training tasks in rapid-response manner [31]. In [32], the authors addressed the problems of delay and energy consumption on CNN model training based on mobile devices. In [33], the authors dealt with the issue of high computational cost for human activity recognition by designing a collaborative scheme on mobile devices. However, few works have considered leveraging mobile devices to provide sufficient services on CNN training tasks for the Metaverse. In this work, the integration of the convolution operation and the mobile-assisted CDC framework as an efficient solution to tackle the aforementioned issues and enhance the interaction experience between virtual characters and users in the Metaverse.

III. System Model and Problem Formulation

In this section, we first introduce the system model for the Metaverse based on a mobile-assisted CDC framework. Then, we describe the problem in coded convolution calculations arising from straggling, colluding, and malicious mobile devices.

A. Why Convolution Calculation is Required by Metaverse

Convolutional neural networks as a critical technology has extensive applications in the Metaverse. Specifically, several specific applications in the metaverse as follows:

- Enhanced immersive experiences: CNNs can be applied to enhance immersive experiences by processing and generating images, videos, or sounds in the virtual scenes.
- Virtual scene generation and rendering: CNNs can be used to generate and render virtual objects of the virtual scenes, constructing a realistic scene by learning environment features.
- Face recognition and expression analysis: CNNs can be used for face recognition and expression analysis in the metaverse, allowing virtual characters to react to users according to their facial expressions and emotions.
- Pose estimation and motion capture: CNNs enable the pose estimation and motion capture of virtual characters in the metaverse, providing realistic interaction and feedback based on movements and poses of Metaverse users.

As mentioned above, CNNs are widely applied to the Metaverse. Therefore, the efficiency and accuracy of convolution calculation significantly impacts the immersive experiences for users in the Metaverse.

B. System Model

We consider a system model for the Metaverse, which consists of Metaverse users (MUs), participating in real-time interactions with both the physical and virtual worlds, MSPCSs, providing various Metaverse services for MUs, and mobile devices, e.g., smartphones, tablets, in-vehicle computing devices, and wearable devices, owning and operating idle computing and storage resources. First, the MU starts a Metaverse application, such as a massive multiplayer online 3D game. Then, the Metaverse system assigns an MSPCS to provide Metaverse services for the MU. In the Metaverse, millions of virtual objects for expansive virtual scenes need to be rendered and processed in real-time. To efficiently complete the above time-sensitive rendering and interaction tasks, the MSPCS distributes them to neighboring mobile devices with idle computing and storage resources. Each mobile device only needs to complete a very small task, which greatly saves the battery power of the mobile device. Considering the widely-used application of CNNs in the Metaverse, this paper focuses on large-scale convolution computing tasks.

As shown in Fig. 1, we construct a mobile-assisted coded distributed convolution system for the Metaverse. Unlike the cloud-based computing framework, our proposed mobile-assisted framework offloads tasks to nearby mobile devices for computing, which greatly reduces the delay caused by long-distance transmission. In the system, an MSPCS playing the role of a

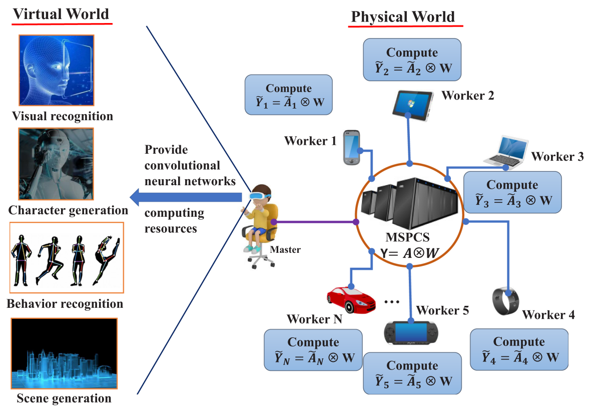

Virtual World
Visual recognition
Character generation
Behavior recognition
Scene generation
Provide convolutional neural networks
computing resources
Physical World
Compute $\tilde{Y}_1 = \tilde{A}_1 \otimes W$
Compute $\tilde{Y}_2 = \tilde{A}_2 \otimes W$
Compute $\tilde{Y}_3 = \tilde{A}_3 \otimes W$
Compute $\tilde{Y}_4 = \tilde{A}_4 \otimes W$
Compute $\tilde{Y}_5 = \tilde{A}_5 \otimes W$
Compute $\tilde{Y}_N = \tilde{A}_N \otimes W$
Worker 1
Worker 2
Worker 3
Worker 4
Worker 5
Worker N
Master
MSPCS
$Y = A \otimes W$

Fig. 1. Overview of the resilient, secure and private coded convolution for distributed computing systems in Metaverse. The system includes a master and N workers. In the system, our proposed RSPCC scheme can tolerate the presence of up to S straggling workers, T colluding workers and A malicious workers among N workers. First, the master divides the input matrix $\mathbf{A}$ and encodes the submatrices $\{\mathbf{A}_i\}_{i=1}^{M}$. Second, N workers $\{W_i\}_{i=1}^{N}$ are assigned with a subtask $\tilde{\mathbf{Y}}_i = \tilde{\mathbf{A}}_i \otimes \mathbf{W}$ by the master, and return the subtask results back to the master. Finally, the master recovers the final result, and then checks the correctness of the result.

master node has many time-consuming convolution calculation tasks, and N mobile devices acting as workers are hired to complete assigned computation tasks by the master. Denote the large-scale convolution operation task as $\mathbf{Y} = \mathbf{A} \otimes \mathbf{W}$. We consider that there are S straggling workers, T colluding workers and A malicious workers over N workers. The input matrix $\mathbf{A} \in \mathbb{F}^{r \times r}$ is a high-dimensional matrix denoting the input feature map, $\mathbf{W} \in \mathbb{F}^{v \times v}$ is the convolutional kernel. $\otimes$ denotes the convolution operation, r and v are large positive integers, $r \gg v$, and finite field $\mathbb{F}$ is sufficiently large [34]. Each worker can only store $\frac{1}{M}$ fraction of $\mathbf{A}$ and a small kernel $\mathbf{W}$. Next, we describe the system by the following steps.

1) *Data Partition&Encoding:* The input matrix $\mathbf{A}$ is divided into M submatrices $\{\mathbf{A}_i\}_{i=1}^{M}$ by the master using Algorithm 1. Then, the master uses N encoding functions to encode the input submatrices $\{\mathbf{A}_i\}_{i=1}^{M}$, i.e.,
$$\tilde{\mathbf{A}}_i = f_i(\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_M) \text{ for } i \in \mathcal{N}, \tag{1}$$
where $\tilde{\mathbf{A}}_i \in \mathbb{F}^{n \times n}$, $\mathcal{N}$ is the set of the indexes of the fastest N workers, and M is a positive integer denoting the number of submatrices of $\mathbf{A}$. Encoding function is denoted as f_i such that $f_i : \mathbb{F}^{r \times r} \to \mathbb{F}^{n \times n}$. The master sends $\tilde{\mathbf{A}}_i$ and $\mathbf{A}_M$ to worker W_i for $i \in \mathcal{N}$.
2) *Task Computing:* Each worker W_i ($i \in \mathcal{N}$) is assigned to compute the convolution operation subtask $\tilde{\mathbf{Y}}_i = \tilde{\mathbf{A}}_i \otimes \mathbf{W}$. To further speed up the process of convolution operation, we transform the subtask $\tilde{\mathbf{Y}}_i = \tilde{\mathbf{A}}_i \otimes \mathbf{W}$ into a matrix multiplication task using MEC algorithm [16]. Then, we compute the matrix multiplication tasks by optimized libraries, e.g., OpenBLAS. Once worker W_i completes its assigned subtask $\tilde{\mathbf{Y}}_i$, the computed result is immediately sent back to the master. However, in real-world scenarios, some workers may fail to finish the subtask $\tilde{\mathbf{Y}}_i$ or longer than others to return the result to the master. Even worse, some workers may return incorrect results to the master.
3) *Result Recovering:* The master collects the results $\{\tilde{\mathbf{Y}}_i\}_{i \in \mathcal{K}}$ from the fastest $K = |\mathcal{K}|$ workers, where $\mathcal{K}$ is the set of the indexes of the fastest K workers and satisfies $\mathcal{K} \subseteq \mathcal{N}$. Then, the master recovers the final product $\mathbf{Y}$ by applying decoding functions $\{d_i\}_{i \in \mathcal{K}}$, denoted as
$$\mathbf{Y} = d_{\mathcal{K}}\big(\{\tilde{\mathbf{Y}}_i\}_{i \in \mathcal{K}}\big). \tag{2}$$
4) *Result Checking:* After the master recovers the final product $\mathbf{Y}$, we use (28) to check whether the recovered result is correct. If (28) holds, the master accepts the result $\tilde{\mathbf{Y}}_i$, and discards it otherwise. This verification algorithm is inspired by [35], the probability of identifying the invalid results will be discussed later.

Remark 1: It is worth mentioning that the integration of coding theory and distributed computing architecture can achieve excellent performance, such as high reliability, flexible scalability, and fast computation speed, in performing the convolution operation tasks [18], [19], [20]. The master can recover the desired final result by utilizing partial returned subtask results which significantly reduces the execution time of the convolution tasks. Also, it is an efficient solution to mitigate the influence of straggling workers. Moreover, we provide a verification method to ensure computation correctness and to protect data from colluding workers.

C. Related Definitions

Before describing the problem of resilient, secure and private CDC for a convolution calculation task, we give the following definitions.

Definition 1: (*Recovery Threshold*) The recovery threshold $R^*(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{d}, \tilde{\mathbf{y}}, N)$ refers to the minimum number of subtask results returned by the workers to the master that can be correctly decoded to recover the final desired result. A lower recovery threshold means that the master needs fewer subtask results returned by the workers to recover the final result. The recovery threshold of a coding scheme reflects the ability to tolerate stragglers.

Definition 2: (*Straggling Workers*) The straggling workers, also known as stragglers, refer to some workers fail to compute the subtasks or return computed results to the master exceedingly slowly.

Definition 3: (*Colluding Workers*) The colluding workers refer to some workers are curious and honest. They collude with each other to obtain information from the assigned data from the master.

Definition 4: (*Malicious Workers*) The malicious workers, also called adversarial workers, may deliberately return wrong results to the master, aiming to contaminate the final computation result.

D. Problem Formulation

Before describing the problem addressed in this paper, we present an illustrative example of a convolution operation in Fig. 3. The sizes of the input image, convolutional kernel and

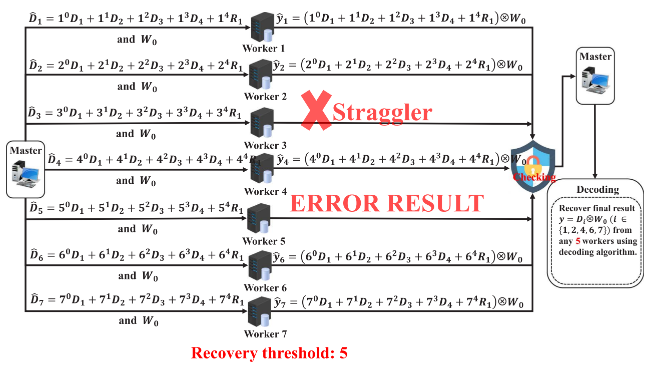

$\hat{D}_1 = 1^0D_1 + 1^1D_2 + 1^2D_3 + 1^3D_4 + 1^4R_1$ and W_0
$\hat{y}_1 = (1^0D_1 + 1^1D_2 + 1^2D_3 + 1^3D_4 + 1^4R_1)\otimes W_0$
Worker 1
$\hat{D}_2 = 2^0D_1 + 2^1D_2 + 2^2D_3 + 2^3D_4 + 2^4R_1$ and W_0
$\hat{y}_2 = (2^0D_1 + 2^1D_2 + 2^2D_3 + 2^3D_4 + 2^4R_1)\otimes W_0$
Worker 2
$\hat{D}_3 = 3^0D_1 + 3^1D_2 + 3^2D_3 + 3^3D_4 + 3^4R_1$ and W_0
Straggler
Worker 3
Master
$\hat{D}_4 = 4^0D_1 + 4^1D_2 + 4^2D_3 + 4^3D_4 + 4^4R_1$ and W_0
$\hat{y}_4 = (4^0D_1 + 4^1D_2 + 4^2D_3 + 4^3D_4 + 4^4R_1)\otimes W_0$
Worker 4
Checking
ERROR RESULT
$\hat{D}_5 = 5^0D_1 + 5^1D_2 + 5^2D_3 + 5^3D_4 + 5^4R_1$ and W_0
Worker 5
$\hat{D}_6 = 6^0D_1 + 6^1D_2 + 6^2D_3 + 6^3D_4 + 6^4R_1$ and W_0
$\hat{y}_6 = (6^0D_1 + 6^1D_2 + 6^2D_3 + 6^3D_4 + 6^4R_1)\otimes W_0$
Worker 6
$\hat{D}_7 = 7^0D_1 + 7^1D_2 + 7^2D_3 + 7^3D_4 + 7^4R_1$ and W_0
$\hat{y}_7 = (7^0D_1 + 7^1D_2 + 7^2D_3 + 7^3D_4 + 7^4R_1)\otimes W_0$
Worker 7
Master
Decoding
Recover final result $y = D_i\otimes W_0$ ($i \in \{1,2,4,6,7\}$) from any 5 workers using decoding algorithm.
Recovery threshold: 5

Fig. 2. An example for the resilient, secure and private coded convolution for a master-worker distributed computing systems with $N = 7$ workers. W_3 is a straggler and W_5 is a malicious worker. (a) Encoding: the master sends encoded data $\hat{\mathbf{D}}_i = \mathbf{D}_1 + i\mathbf{D}_2 + i^2\mathbf{D}_3 + i^3\mathbf{D}_4 + i^5\mathbf{R}_1$ to worker W_i. (b) Computing: each worker computes $\hat{\mathbf{y}}_i = (\mathbf{D}_1 + i\mathbf{D}_2 + i^2\mathbf{D}_3 + i^3\mathbf{D}_4 + i^5\mathbf{R}_1)\otimes\mathbf{W}_0$ and returns the result $\hat{\mathbf{y}}_i$ to the master. (c) Decoding: the master recovers the final result by decoding any 5 results from all workers.

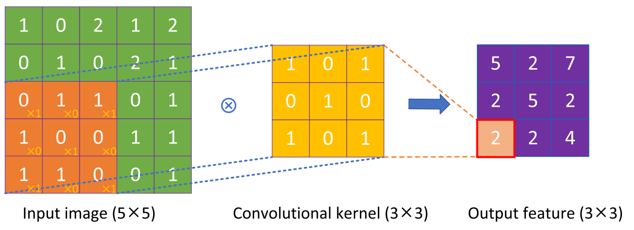


Fig. 3. Example of convolution operation with a 3×3 kernel on a 5×5 input image to obtain 3×3 feature output.

output feature map are 5×5, 3×3 and 3×3, respectively. Fig. 3 demonstrates that the convolution kernel glides over the input image, moving from left to right and top to bottom, with a specified stride (in this case, 1). During each step, the convolutional kernel is bitwise multiplied with the local region of the input image, and the resulting values are summed to produce the corresponding value in the output map. For instance, the convolution operation performed by a convolutional kernel with a yellow background on a local region with an orange background of the input image outputs the value of 2, as framed in red in the output map.

Evidently, the convolution operation involves numerous repeated calculations, which significantly degrades the computing efficiency of the convolutional algorithm. Also, we note that the impact on computing efficiency will be more pronounced with larger input datas. This paper focuses on providing immersive experience for the Metaverse by improving the convolutional algorithm. We aim to design a resilient, secure, and private coded convolution scheme, i.e., the RSPCC scheme, which can speed up the convolution calculation process while considering the existence of stragglers, colluding, and malicious workers in the mobile-assisted CDC system.

In next section, we describe our proposed resilient, secure and private coded convolution scheme based on a distributed computing framework.

IV. Resilient, Secure and Private Coded Convolution Scheme Based on Distributed Computing Framework

In this section, we propose a novel resilient, secure and private coded convolution (RSPCC) scheme based on distributed computing scenarios for the Metaverse. As described in Section II, the coding scheme includes four phases. First, we use an illustrating examples to explain the main ideas of the RSPCC scheme before presenting the general description.

A. Illustrating Example

We now consider a convolution operation task, i.e., $\mathbf{y}_0 = \mathbf{D}\otimes\mathbf{W}_0$, in a master-worker system with one master and $N = 7$ workers. In the system, W_3 is a straggler, W_5 is a malicious worker and the number of colluding workers is $T = 1$. The stride is 1. Matrices $\mathbf{D}$ and $\mathbf{W}_0$ have dimensions of 5×5 and 3×3, respectively, as follows:

$$\mathbf{D} = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{15} \\ d_{21} & d_{22} & \cdots & d_{25} \\ \vdots & \vdots & \ddots & \vdots \\ d_{51} & d_{52} & \cdots & d_{55} \end{bmatrix}, \mathbf{W}_0 = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}. \tag{3}$$

Then, matrix $\mathbf{D}$ is divided as follows:

$$\mathbf{D}_1 = \begin{bmatrix} d_{11} & d_{12} & d_{13} & d_{14} \\ d_{21} & d_{22} & d_{23} & d_{24} \\ d_{31} & d_{32} & d_{33} & d_{34} \\ d_{41} & d_{42} & d_{43} & d_{44} \end{bmatrix}, \mathbf{D}_2 = \begin{bmatrix} d_{12} & d_{13} & d_{14} & d_{15} \\ d_{22} & d_{23} & d_{24} & d_{25} \\ d_{32} & d_{33} & d_{34} & d_{35} \\ d_{42} & d_{43} & d_{44} & d_{45} \end{bmatrix},$$

$$\mathbf{D}_3 = \begin{bmatrix} d_{21} & d_{22} & d_{23} & d_{24} \\ d_{31} & d_{32} & d_{33} & d_{34} \\ d_{41} & d_{42} & d_{43} & d_{44} \\ d_{51} & d_{52} & d_{53} & d_{54} \end{bmatrix}, \mathbf{D}_4 = \begin{bmatrix} d_{22} & d_{23} & d_{24} & d_{25} \\ d_{32} & d_{33} & d_{34} & d_{35} \\ d_{42} & d_{43} & d_{44} & d_{45} \\ d_{52} & d_{53} & d_{54} & d_{55} \end{bmatrix}. \tag{4}$$

Thus, the convolution operation task can be expressed by

$$\mathbf{y}_0 = \mathbf{D}\otimes\mathbf{W}_0 = \begin{bmatrix} \mathbf{y}_1 & \mathbf{y}_2 \\ \mathbf{y}_3 & \mathbf{y}_4 \end{bmatrix}, \tag{5}$$

where $\mathbf{y}_i = \mathbf{D}_i\otimes\mathbf{W}_0$ for $i \in \{1, 2, 3, 4\}$.

The master encodes the submatrices $\{\mathbf{D}_i\}_{i=1}^4$ using the encoding function, defined as follows:

$$\hat{\mathbf{D}}_i = \mathbf{D}_1 + i\mathbf{D}_2 + i^2\mathbf{D}_3 + i^3\mathbf{D}_4 + i^4\mathbf{R}_1, \tag{6}$$

where $\hat{\mathbf{D}}_i$ has a dimension of 4×4. $\mathbf{R}_1 \in \mathbb{F}^{4\times 4}$ is random generated independently from $\mathbf{A}$ and $\mathbf{W}_0$ by the master.

As shown in Fig. 2, the master sends encoded matrix $\hat{\mathbf{D}}_i$ and convolutional kernel $\mathbf{W}_0$ to worker W_i for $i \in \{1, 2, \ldots, 7\}$. Then, each worker stores the convolutional kernel $\mathbf{W}_0$. After receiving the assigned encoded data, the worker W_i performs the following computing task.

$$\begin{aligned} \hat{\mathbf{y}}_i &= \hat{\mathbf{D}}_i\otimes\mathbf{W}_0 \\ &= (\mathbf{D}_1 + i\mathbf{D}_2 + i^2\mathbf{D}_3 + i^3\mathbf{D}_4 + i^4\mathbf{R}_1)\otimes\mathbf{W}_0. \end{aligned} \tag{7}$$

Upon completion of the computation by worker W_i, the computation result $\hat{\mathbf{y}}_i$ is returned back to the master. It is worth noting that the above convolution operation is performed by the MEC algorithm [16] based on optimized libraries, e.g., OpenBLAS [36], which significantly reduces the execution time and memory overhead.

In practical scenarios, some mobile devices as the workers may fail to complete their tasks or complete the tasks slowly. As shown in Fig. 2, the computed result of worker W_3 is not returned back to the master in a timely manner. Nevertheless, the master is capable of recovering the correct final result $\mathbf{Y}$. In this example, the recovery threshold is 5. To begin with, the master has received the following results:

$$\begin{aligned}
\hat{\mathbf{y}}_1 &= (\mathbf{D}_1 + 1\cdot\mathbf{D}_2 + 1^2\cdot\mathbf{D}_3 + 1^3\cdot\mathbf{D}_4 + 1^4\cdot\mathbf{R}_1)\otimes\mathbf{W}_0,\\
\hat{\mathbf{y}}_2 &= (\mathbf{D}_1 + 2\cdot\mathbf{D}_2 + 2^2\cdot\mathbf{D}_3 + 2^3\cdot\mathbf{D}_4 + 2^4\cdot\mathbf{R}_1)\otimes\mathbf{W}_0,\\
\hat{\mathbf{y}}_4 &= (\mathbf{D}_1 + 4\cdot\mathbf{D}_2 + 4^2\cdot\mathbf{D}_3 + 4^3\cdot\mathbf{D}_4 + 4^4\cdot\mathbf{R}_1)\otimes\mathbf{W}_0,\\
\hat{\mathbf{y}}_6 &= (\mathbf{D}_1 + 6\cdot\mathbf{D}_2 + 6^2\cdot\mathbf{D}_3 + 6^3\cdot\mathbf{D}_4 + 6^4\cdot\mathbf{R}_1)\otimes\mathbf{W}_0,\\
\hat{\mathbf{y}}_7 &= (\mathbf{D}_1 + 7\cdot\mathbf{D}_2 + 7^2\cdot\mathbf{D}_3 + 7^3\cdot\mathbf{D}_4 + 7^4\cdot\mathbf{R}_1)\otimes\mathbf{W}_0,
\end{aligned} \tag{8}$$

The above equations can be rewritten as

$$\begin{bmatrix}\hat{\mathbf{y}}_1\\ \hat{\mathbf{y}}_2\\ \hat{\mathbf{y}}_4\\ \hat{\mathbf{y}}_6\\ \hat{\mathbf{y}}_7\end{bmatrix} = \begin{bmatrix}1&1&1&1&1\\ 2^0&2^1&2^2&2^3&2^4\\ 4^0&4^1&4^2&4^3&4^4\\ 6^0&6^1&6^2&6^3&6^4\\ 7^0&7^1&7^2&7^3&7^4\end{bmatrix}\begin{bmatrix}\mathbf{D}_1\otimes\mathbf{W}_0\\ \mathbf{D}_2\otimes\mathbf{W}_0\\ \mathbf{D}_3\otimes\mathbf{W}_0\\ \mathbf{D}_4\otimes\mathbf{W}_0\\ \mathbf{R}_1\otimes\mathbf{W}_0\end{bmatrix}. \tag{9}$$

Obviously, the coefficient matrix in (9) is the Vandermonde matrix. It is invertible, and then we have

$$\begin{bmatrix}\mathbf{D}_1\otimes\mathbf{W}_0\\ \mathbf{D}_2\otimes\mathbf{W}_0\\ \mathbf{D}_3\otimes\mathbf{W}_0\\ \mathbf{D}_4\otimes\mathbf{W}_0\\ \mathbf{R}_1\otimes\mathbf{W}_0\end{bmatrix} = \begin{bmatrix}1&1&1&1&1\\ 2^0&2^1&2^2&2^3&2^4\\ 4^0&4^1&4^2&4^3&4^4\\ 6^0&6^1&6^2&6^3&6^4\\ 7^0&7^1&7^2&7^3&7^4\end{bmatrix}^{-1}\begin{bmatrix}\hat{\mathbf{y}}_1\\ \hat{\mathbf{y}}_2\\ \hat{\mathbf{y}}_4\\ \hat{\mathbf{y}}_6\\ \hat{\mathbf{y}}_7\end{bmatrix}. \tag{10}$$

We observe that $\mathbf{D}_i\otimes\mathbf{W}_0$ for $i\in\{1,2,3,4\}$ can be fully obtained from (10). In addition, the presence of malicious workers that may deliberately return wrong computation results to the master. For instance, worker W_5 return the computed result $\hat{\mathbf{y}}_5 + \mathbf{z}_5$ to the master, where $\mathbf{z}_5$ is a nonzero perturbation term. Then, the master checks the correctness of the result from W_5 based on (28) is given by

$$\begin{aligned}
\Delta &= \sum_{j=1}^{6} S'_{6,j}\times(\hat{\mathbf{y}}_5 + \mathbf{z}_5)\\
&= \sum_{j=1}^{6} S'_{6,j}\left(\sum_{t=1}^{5} 5^{t-1}\mathbf{D}_t\otimes\mathbf{W}_0 + \mathbf{z}_5\right)\\
&= \sum_{t=1}^{5}\left(\sum_{j=1}^{6} 5^{t-1}S'_{6,j}\right)\mathbf{D}_t\otimes\mathbf{W}_0 + \sum_{j=1}^{6} S'_{6,j}\mathbf{z}_5\\
&= \sum_{j=1}^{6} S'_{6,j}\mathbf{z}_5,
\end{aligned} \tag{11}$$

where $S'_{6,j}$ is obtained by (24) and denotes the element of the 6th row and j-th column of the inverse of the coefficient matrix in (9). Thus, the master discards the computed result from worker W_5 due to $\Delta\neq 0$. The specific derivation process can be found in Section IV-B.

Considering the expensive computing cost of the inverse operation of the matrix, we recover the results $\mathbf{y}_0$ using polynomial interpolation [34]. Specifically, the computed result of worker W_i is essentially the value of the 4th-degree polynomial $\hbar(x)$ at point $x = x_i$, where $\hbar(x)$ is given as follows:

$$\hbar(x) = (\mathbf{D}_1 + x\mathbf{D}_2 + x^2\mathbf{D}_3 + x^3\mathbf{D}_4 + x^4\mathbf{R}_1)\otimes\mathbf{W}_0. \tag{12}$$

Then, all coefficients of $\hbar(x)$ can be evaluated over 5 distinct points by polynomial interpolation. Therefore, the master can successfully recover the desired final result by obtaining any 5 fastest results from the 7 workers. This implies that the recovery threshold is 5.

B. General RSPCC Scheme Design

In this section, we present the general description of the RSPCC scheme for the Metaverse. In the system, the master wants to complete a large-scale convolution operation task $\mathbf{Y} = \mathbf{A}\otimes\mathbf{W}$ with N workers. The system includes S straggling workers, T colluding workers and A malicious workers out of N workers while satisfying $M + T + A + S \le N$. The following five steps are given to describe the procedure.

1) Data Partition & Encoding: Recalling Section II, we note that the dimension of matrix $\mathbf{A}$ is $r\times r$. Let the dimension of each divided submatrix be $n\times n$, the number of submatrices can be determined by

$$M = \left(\frac{r - n + 2P}{s} + 1\right)^2, \tag{13}$$

where P and s denote the padding and stride, respectively. The concrete procedure of dividing the matrix $\mathbf{A}$ is presented in Algorithm 1. Then, the submatrices $\{\mathbf{A}_i\}_{i=1}^{M}$ can be written in the form of the following matrix:

$$\breve{\mathbf{A}} = \begin{bmatrix}\mathbf{A}_1 & \mathbf{A}_2 & \cdots & \mathbf{A}_m\\ \mathbf{A}_{m+1} & \mathbf{A}_{m+2} & \cdots & \mathbf{A}_{2m}\\ \vdots & \vdots & \ddots & \vdots\\ \mathbf{A}_{(m-1)m+1} & \mathbf{A}_{(m-1)m+2} & \cdots & \mathbf{A}_M\end{bmatrix}, \tag{14}$$

where $m = \sqrt{M}$. Thus, the original task $\mathbf{Y} = \mathbf{A}\otimes\mathbf{W}$ of the master is converted into $\mathbf{Y} = \breve{\mathbf{A}}\otimes\mathbf{W}$.

The master selects N distinct elements $\{x_1, x_2, \ldots, x_N\}$ of $\mathbb{F}_q$, and then encodes the submatrices $\{\mathbf{A}_i\}_{i=1}^{M}$ using the following encoding function:

$$\begin{aligned}
\tilde{\mathbf{A}}_i &= \tilde{f}_{\mathbf{A}}(x_i)\\
&= \sum_{j=1}^{M} x_i^{j-1}\mathbf{A}_j + \sum_{k=1}^{T} x_i^{M+k-1}\mathbf{R}_k,\ \text{for } i\in\mathcal{N},
\end{aligned} \tag{15}$$

Algorithm 1: Matrix Division Algorithm.

```
Input: $\mathbf{A}$, $M$, $s$, $n$, $r$;
Output: $\{\mathbf{A}_i\}_{i=1}^{M}$;
1  Initialize parameters $m_0 = \sqrt{M}$ and $i = 1$;
2  Start to divide matrix $\mathbf{A}$;
3  for $ii = 1 : m_0$ do
4      for $jj = 1 : m_0$ do
5          New a matrix: $\mathbf{A}_i$;
6          II = $ii \times s$;
7          JJ = $jj \times s$;
8          for $kk = 1 : n$ do
9              for $k = 1 : n$ do
10                 $\mathbf{A}_i[kk][k] = \mathbf{A}[\mathrm{II}+kk][\mathrm{JJ}+k]$;
11             end
12         end
13         $i = i + 1$;
14     end
15 end
16 Return: $\{\mathbf{A}_i\}_{i=1}^{M}$
```

where $\tilde{\mathbf{A}}_i \in \mathbb{F}^{n\times n}$, and $\{\mathbf{R}_k\}_{k=1}^{T} \in \mathbb{F}^{n\times n}$ are random matrices and generated independently from $\mathbf{A}$ and $\mathbf{W}$ by the master. Then, the master sends encoded matrix $\tilde{\mathbf{A}}_i$ and convolutional kernel $\mathbf{W}$ to worker W_i for $i \in \mathcal{N}$. The convolutional kernel $\mathbf{W}$ is stored on worker W_i.

2) Task Computing: When worker W_i receives the data $\tilde{\mathbf{A}}_i$ and $\mathbf{W}$ from the master, they compute the subtask:

$$\tilde{\mathbf{Y}}_i = \tilde{\mathbf{A}}_i \otimes \mathbf{W} = \left(\sum_{j=1}^{M} x_i^{j-1}\mathbf{A}_j + \sum_{k=1}^{T} x_i^{M+k-1}\mathbf{R}_k\right)\otimes\mathbf{W}. \tag{16}$$

To further speed up the process of convolution operation, the subtask $\tilde{\mathbf{Y}}_i = \tilde{\mathbf{A}}_i\otimes\mathbf{W}$ is transformed into matrix multiplication tasks using the MEC algorithm [16]. Additionally, we compute the matrix multiplication tasks by optimized libraries, e.g., OpenBLAS [36]. The MEC algorithm can greatly reduce memory overhead, and we present it in Section III-C. Once the worker W_i completes the subtask, the worker immediately returns the computed result $\tilde{\mathbf{Y}}_i$ to the master.

3) Result Recovering: Note that the step (*2) Task Computing*, we observe that (16) is essentially the value of the $(M+T-1)$th-degree polynomial $h(x)$ at point $x = x_i$, where $h(x)$ is given by

$$h(x) = \left(\sum_{j=1}^{M} x^{j-1}\mathbf{A}_j + \sum_{k=1}^{T} x^{M+k-1}\mathbf{R}_k\right)\otimes\mathbf{W}. \tag{17}$$

Obviously, the coefficients of powers of x in $h(x)$ contain all values of the final desired result, i.e., $\{\mathbf{A}_i\otimes\mathbf{W}\}_{i=1}^{M}$. Thus, the result recovering problem can be transformed into a polynomial interpolation problem. The master only needs to receive $M+T$ correct subtask results from N workers. Then, all coefficients of $h(x)$ can be evaluated over $M+T$ distinct points by polynomial interpolation. Therefore, the recovery threshold is $M+T$.

4) Result Checking: Let $\mathcal{K}'$ ($\mathcal{K} \subseteq \mathcal{N}$) be a set of the indexes of the fastest $(M+T+1)$ workers which return the subtask results back to the master. The returned result from worker W_{k_j} is denoted by $\tilde{\mathbf{Y}}_{k_j}$ for $j \in \{1,2,\ldots,M+T+1\}$ and $k_j \in \mathcal{K}'$. Given a Vandermonde matrix $\mathbf{V}$ as

$$\mathbf{V} = \begin{bmatrix} x_{k_1}^0 & x_{k_1}^1 & \cdots & x_{k_1}^{M+T} \\ x_{k_2}^0 & x_{k_2}^1 & \cdots & x_{k_2}^{M+T} \\ \vdots & \vdots & \ddots & \vdots \\ x_{k_{M+T+1}}^0 & x_{k_{M+T+1}}^1 & \cdots & x_{k_{M+T+1}}^{M+T} \end{bmatrix}. \tag{18}$$

The matrix $\mathbf{V}$ is invertible because the parameters of $x_{k_1}, x_{k_2}, \ldots, x_{M+T+1}$ are distinct. We use $\mathbf{S}$ to represent the inverse of matrix $\mathbf{V}$. Then, we extract the first $M+T$ columns of matrix $\mathbf{V}$ to form a new matrix $\mathbf{U}$, i.e.,

$$\mathbf{U} = \begin{bmatrix} x_{k_1}^0 & x_{k_1}^1 & \cdots & x_{k_1}^{M+T-1} \\ x_{k_2}^0 & x_{k_2}^1 & \cdots & x_{k_2}^{M+T-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{k_{M+T+1}}^0 & x_{k_{M+T+1}}^1 & \cdots & x_{k_{M+T+1}}^{M+T-1} \end{bmatrix}. \tag{19}$$

From (19) and $\mathbf{SV} = \mathbf{I}$, we have.

$$\sum_{k=1}^{M+T+1} S_{i,k}V_{k,i} = 1,\ \forall i \in \{1,2,\ldots,(M+T)\}, \tag{20}$$

and

$$\sum_{k=1}^{M+T+1} S_{t,k}V_{k,c} = 0, \tag{21}$$

where $t \in \{1,2,\ldots,(M+T+1)\}$ and $c \in \{1,2,\ldots,(M+T)\}$. From (18)–(21), we have

$$\mathbf{S}\cdot\mathbf{U} = \begin{bmatrix} \mathbf{I}_{M+T} \\ \mathbf{0}_{1,M+T} \end{bmatrix}, \tag{22}$$

where $\mathbf{I}_{M+T}$ denotes the identity matrix with dimension $(M+T)\times(M+T)$, and $\mathbf{0}_{1,M+T}$ denotes the row vector with dimension $M+T$. From (22), we can obtain

$$\sum_{j=1}^{M+T+1} S_{M+T+1,j}\cdot x_{k_j}^{t-1} = 0, \tag{23}$$

where $S_{M+T+1,j}$ represents the element of $(M+T+1)$-th row and j-th column of the matrix $\mathbf{S}$, and $t \in \{1,2,\ldots,M+T\}$. To reduce the computing cost of the inverse of matrix $\mathbf{V}$, we obtain the element $S_{M+T+1,j}$ using the proposed approach from [37] as follows:

$$S_{M+T+1,j} = \frac{1}{\prod_{i\in\mathcal{I},i\neq j}(x_{k_j} - x_{k_i})}, \tag{24}$$

where $\mathcal{I} \triangleq \{1,2,\ldots,M+T+1\}$, $k_j \in \mathcal{K}'$, and $k_i \in \mathcal{K}'$.

If there exists a malicious worker in $\{W_{k_j}\}_{j=1}^{M+T+1}$ where $k_j \in \mathcal{K}'$, the returned subtask result from worker W_{k_j} can be expressed as

$$\tilde{\mathbf{Y}}_{k_j} = \left(\sum_{t=1}^{M} x_{k_j}^{t-1}\mathbf{A}_t + \sum_{k=1}^{T} x_{k_j}^{M+k-1}\mathbf{R}_k\right)\otimes\mathbf{W} + \mathbb{I}_{\mathcal{K}'}(k_j)\mathbf{Z}_{k_j}. \tag{25}$$

Algorithm 2: RSPCC Algorithm.

```
   Input: $\mathbf{A}, \mathbf{W}, N, M, T$
   Output: $\mathbf{Y}$
1  [ I ] Data Partition&Encoding: the master divides and
     encodes $\mathbf{A}$;
2  The master divides $\mathbf{A}$ into $M$ submatrices $\{\mathbf{A}_i\}_{i=1}^{M}$ by
     Algorithm 1;
3  for $i = 1 : N$ do
4      $\tilde{\mathbf{A}}_i = \sum_{j=1}^{M} x_i^{j-1}\mathbf{A}_j + \sum_{k=1}^{T} x_i^{M+k-1}\mathbf{R}_k$;
5  end
6  [ II ] Task Computing: the worker $W_i$ computes
     $\tilde{\mathbf{Y}}_i = \tilde{\mathbf{A}}_i \otimes \mathbf{W}$;
7  Each subtask is transformed into $\breve{\mathbf{y}}_i = \breve{\mathbf{A}}_i\breve{\mathbf{w}}$ by Algorithm 3;
8  The worker $W_i$ compute $\breve{\mathbf{y}}_i = \breve{\mathbf{A}}_i\breve{\mathbf{w}}$ by OpenBLAS;
9  $W_i$ transforms computed $\breve{\mathbf{y}}_i$ into original result $\tilde{\mathbf{Y}}_i$;
10 $W_i$ returns result $\tilde{\mathbf{Y}}_i$ to the master;
11 [ III ] Result Recovering: the master recovers $\mathbf{Y}$;
12 The master checks the size of the set $\mathcal{K}$;
13 if $|\mathcal{K}| \geq M + T$ then
14     The master decodes $\mathbf{Y} = d_{\mathcal{K}}(\{\tilde{\mathbf{Y}}_i\}_{i\in\mathcal{K}})$;
15 end
16 [ IV ] Result Checking: the master check $\mathbf{Y}$ by Eq. (28);
17 if $\mathfrak{R}_{\mathcal{K}'} = \mathbf{0}_{\mathrm{H_{out}}\times\mathrm{W_{out}}}$ then
18     The recovered final result $\mathbf{Y}$ is correct;
19 else
20     The recovered final result $\mathbf{Y}$ is incorrect;
21 end
22 Return: $\mathbf{Y}$
```

where $\mathbf{Z}_{k_j}$ is the nonzero perturbation term, and $\mathbb{I}_{\mathcal{K}'}(k_j)$ represents the indicator function, it is 0 if worker W_{k_j} is a Byzantine worker, and 0 otherwise. Let $\mathbf{A}_{M+1} = \mathbf{R}_1, \mathbf{A}_{M+2} = \mathbf{R}_2, \ldots, \mathbf{A}_{M+T} = \mathbf{R}_T$, then we have

$$\sum_{t=1}^{M} x_{k_j}^{t-1}\mathbf{A}_t + \sum_{k=1}^{T} x_{k_j}^{M+k-1}\mathbf{R}_k = \sum_{t=1}^{M+T} x_{k_j}^{t-1}\mathbf{A}_t. \tag{26}$$

From (26), (25) can be rewritten as

$$\tilde{\mathbf{Y}}_{k_j} = \sum_{t=1}^{M+T} x_{k_j}^{t-1}\mathbf{A}_t\otimes\mathbf{W} + \mathbb{I}_{\mathcal{K}'}(k_j)\mathbf{Z}_{k_j}. \tag{27}$$

Now, the master checks the returned subtask results $\{\tilde{\mathbf{Y}}_{k_j}\}_{k_j\in\mathcal{K}'}$ by

$$\begin{aligned}\mathfrak{R}_{\mathcal{K}'} &= \sum_{j=1}^{M+T+1} S_{M+T+1,j}\times\tilde{\mathbf{Y}}_{k_j}\\ &= \sum_{j=1}^{M+T+1} S_{M+T+1,j}\left(\sum_{t=1}^{M+T} x_{k_j}^{t-1}\mathbf{A}_t\otimes\mathbf{W} + \mathbb{I}_{\mathcal{K}'}(k_j)\mathbf{Z}_{k_j}\right)\\ &= \sum_{t=1}^{M+T}\left(\sum_{j=1}^{M+T+1} S_{M+T+1,j}x_{k_j}^{t-1}\right)\mathbf{A}_t\otimes\mathbf{W}\\ &\quad+\sum_{j=1}^{M+T+1} S_{M+T+1,j}\mathbb{I}_{\mathcal{K}'}(k_j)\mathbf{Z}_{k_j}\\ &= \sum_{j=1}^{M+T+1} S_{M+T+1,j}\mathbb{I}_{\mathcal{K}'}(k_j)\mathbf{Z}_{k_j}.\end{aligned} \tag{28}$$

From the above equation, we have $\mathfrak{R}_{\mathcal{K}'} = \mathbf{0}_{\mathrm{H_{out}}\times\mathrm{W_{out}}}$ where $\mathrm{H_{out}} = \mathrm{W_{out}} = (\mathrm{n} + 2\mathrm{P} - \mathrm{v})/\mathrm{S}$ if no Byzantine nodes return the computation results to the master. However, we know that $\mathfrak{R}_{\mathcal{K}'} = \mathbf{0}_{\mathrm{H_{out}}\times\mathrm{W_{out}}}$ can still hold even if there are Byzantine nodes participating in result recovering. Its probability is $1/Q$ [35]. Q denotes the size of the finite field $\mathbb{F}$. Therefore, we can choose a sufficiently large Q such that the probability of $1/Q$ tends towards zero.

The specific procedure of the RSPCC scheme is presented in Algorithm 2.

Remark 2: Our proposed RSPCC scheme offers several advantages for distributed systems. Specifically, it accelerates the convolution process while also reducing memory overhead for workers. Additionally, the RSPCC scheme exhibits excellent performance given its ability to handle slower workers or nodes that do not return results to the master. Furthermore, the RSPCC scheme provides verifiable computing, which ensures that incorrect results can be detected. Finally, the RSPCC scheme can protect confidential data from being compromised, even in scenarios where up to T workers may collude. Overall, the RSPCC scheme represents a significant contribution to the field of distributed computing, offering improved speed, efficiency, and security.

C. Simple Introduction of MEC Algorithm

In this section, we introduce the main ideas of the MEC algorithm [16]. The MEC algorithm improves the im2col algorithm [38] in terms of memory consumption.

As depicted in Fig. 4, the core idea of im2col algorithm is to slide a window over the input matrix and extract data blocks. These small data blocks are then rearranged into columns and stacked together to construct a new matrix. It is evident that this algorithm transforms the input matrix into a larger matrix $\mathbf{B} \in \mathbb{F}^{9\times 4}$ with a significant amount of redundancy. In contrast, Fig. 5 demonstrates the same example utilizing the MEC algorithm, and we observe that the input matrix is transformed into a matrix $\tilde{\mathbf{B}}$ with a reduced dimension of 3×8. Notably, the dimension of matrix $\tilde{\mathbf{B}}$ is 33% smaller at 3×8 than the 9×4 dimension attained by the im2col algorithm. Consequently, the MEC algorithm outperforms the im2col algorithm in terms of memory efficiency. The transformation process is elaborated in Algorithm 3.

In our master-worker system, the memory resources of each worker are limited. Thus, in the *Task Computing* phase of the proposed RSPCC scheme, each worker transforms the subtask $\tilde{\mathbf{Y}}_i = \tilde{\mathbf{A}}_i\otimes\mathbf{W}$ into matrix multiplications using the MEC algorithm to save memory.

V. Result and Complexity Analyses

In this section, we present our main theoretical results on the proposed optimization algorithm for convolution operation based on coded distributed computing. Then, we give the complexity analysis about the proposed RSPCC scheme.

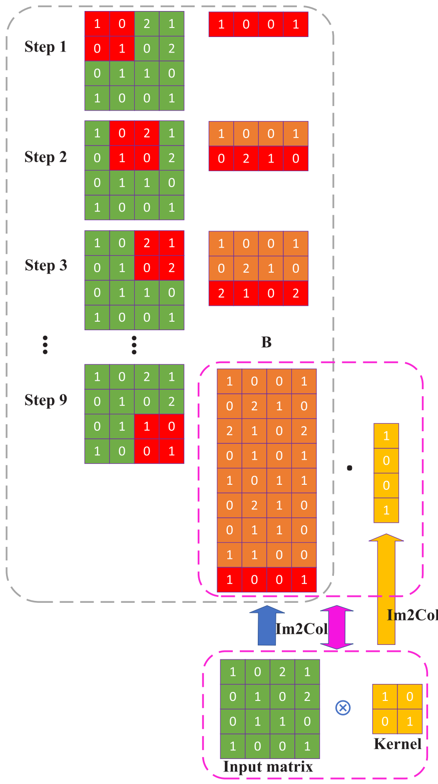


Fig. 4. Example of describing the main idea of im2col algorithm. By using im2col algorithm, the convolution operation with a 2 × 2 kernel on a 4 × 4 input image is transformed into a matrix-vector multiplication.

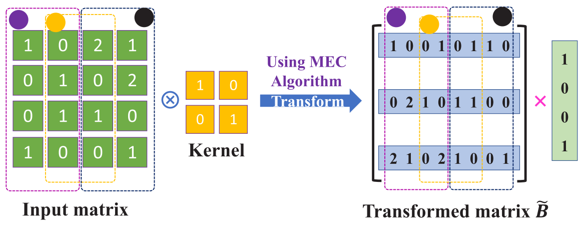


Fig. 5. Example of describing the main idea of MEC algorithm [16]. By using MEC algorithm, the convolution operation with a 2 × 2 kernel on a 4 × 4 input image is transformed into matrix-vector multiplications.

A. Some Theorems of the RSPCC Scheme

The first theorem introduces the recovery threshold obtained by the RSPCC scheme. The master-worker system comprises a master and N workers, which focus on completing task $\mathbf{Y} = \mathbf{A} \otimes \mathbf{W}$.

Theorem 1: For a distributed convolution optimization problem of computing $\mathbf{Y} = \mathbf{A} \otimes \mathbf{W}$ for N workers, where T colluding workers and A malicious workers, each worker pre-stores the convolutional kernel $\mathbf{W}$, using the proposed RSPCC scheme with parameters R, M and T, the optimum recovery threshold R is given by

$$R = M + T, \tag{29}$$

where M denotes the number of the submatrices from the large-scale input matrix $\mathbf{A}$.

Algorithm 3: Task Transforming Algorithm.

```
   Input: $\tilde{\mathbf{Y}}_i$, $\tilde{\mathbf{A}}_i$, $\mathbf{W}$;
   Output: $\check{\mathbf{A}}_i$, $\check{\mathbf{w}}$;
1  [ I ] Transform $\mathbf{W}$ into $\check{\mathbf{w}}$;
2  Firstly, new a matrix: $\check{\mathbf{w}}$;
3  for $ii = 1 : v$ do
4      for $jj = 1 : v$ do
5          index = $(ii - 1) \times v + jj$;
6          $\check{\mathbf{w}}$[index]=$\mathbf{W}[ii][jj]$;
7      end
8  end
9  [ II ] Transform $\tilde{\mathbf{A}}_i$ into $\check{\mathbf{A}}_i$;
10 New a matrix: $\check{\mathbf{A}}_i$;
11 The number of the submatrices: $u = (\frac{n-v+2P}{S} + 1)$;
12 for $k = 1 : u$ do
13     $\check{\mathbf{A}}_k = \tilde{\mathbf{A}}_i[1 : n, (k-1) \times S + 1 : v]$;
14 end
15 New a matrix: $\check{\mathbf{A}}_i$;
16 for $k = 1 : u$ do
17     for $ii = 1 : n$ do
18         for $jj = 1 : v$ do
19             index = $(ii - 1) \times v + jj$;
20             $\check{\mathbf{A}}_i[k][\text{index}] = \check{\mathbf{A}}_k[ii][jj]$;
21         end
22     end
23 end
24 Return: $\check{\mathbf{A}}_i$, $\check{\mathbf{w}}$
```

Proof: The proof of the theorem is equivalent to proving that can recover the desired result $\mathbf{Y} = \mathbf{A} \otimes \mathbf{W}$ by leveraging any R returned subtask results from N workers. From (17), we can see that the coefficients of powers of x in $h(x)$ contain all elements $\{\mathbf{A}_i \otimes \mathbf{W}\}_{i=1}^{M}$ of the final result. Then, the result recovering problem is transformed into a polynomial interpolation problem. Because the degree of the polynomial $h(x)$ is $(M + T - 1)$. Therefore, all coefficients of powers of x in $h(x)$ can be evaluated at $M + T$ distinct points by using polynomial interpolation [34]. Thus, we can conclude that the master can recover the final computed result by the any fastest $M + T$ returned results, i.e., the recovery threshold is $M + T$. This completes the proof. ■

To the best of our knowledge, most of the existing works [20], [24], [34] use polynomial interpolation to decode the final result for the coded distributed matrix multiplication problem. In Theorem 1, we overcome the limitation of those works and successfully decode the final result by using polynomial interpolation for the coded distributed convolution problem.

Theorem 2: For the CDC framework, the minimum testable size of the proposed RSPCC scheme for verifying the correctness of final computed result is

$$R'_{\min} = M + T + 1, \tag{30}$$

where M denotes the number of the submatrices from the large-scale input matrix $\mathbf{A}$.

Proof: From (17), the degree of the polynomial $h(x)$ is $M + T - 1$. The master needs $M + T$ results to recover the final result $\mathbf{Y}$. The dimension of the matrix V of (18) is $(M + T + 1) \times (M + T + 1)$. From (18), (22) and, (28) the master check the correctness of the final result using less than $M + T + 1$

returned computed results. Thus, the minimum testable size of the proposed RSPCC scheme is $R'_{\min} = M + T + 1$. ■

The third theorem characterizes the set of all feasible S-resilient, A-secure, and T-private schemes that can be achieved by our RSPCC scheme.

Theorem 3: Given the mobile-assisted distributed computing system with a master and N workers, the proposed RSPCC scheme provides an S-resilient, A-secure and T-private for computing $\mathbf{Y} = \mathbf{A} \otimes \mathbf{W}$, if

$$N \geq M + T + A + S + 1, \tag{31}$$

where M denotes the number of the submatrices from the large-scale matrix $\mathbf{A}$.

Proof: The proof of the theorem 2 is shown as follows:

- *S-Resiliency:* The degree of the polynomial $h(x)$ in (17) is $M + T - 1$. Thus, to decode the final result, at least $M + T$ results must be returned back to the master. The presence of S stragglers may result in S subtask results not being received normally by the master. Similarly, there are A adversaries that return incorrect computed results to the master. Thus, A subtask results cannot be accepted by the master. From Theorem 2, the minimum testable size of the proposed RSPCC scheme is $M + T + 1$. Therefore, to successfully decode the final result, the total number of workers N should satisfy $N \geq M + T + S + A + 1$.
- *A-Security:* In the RSPCC scheme, the master verifies the returned subtask result from each worker using the verification algorithm. Only the correct subtask results are accepted. Thus, the RSPCC scheme can tolerate A incorrect results from A malicious workers or adversaries.
- *T-Privacy:* Recalling the encoding function $\tilde{f}_{\mathbf{A}}(x_i)$ in (15), which can be rewritten as

$$\tilde{f}_{\mathbf{A}}(x_i) = [\mathbf{A}_1, \ldots, \mathbf{A}_M, \mathbf{R}_1, \ldots, \mathbf{R}_T]\mathbf{X}_i, \tag{32}$$

where $\mathbf{X}_i \in \mathbb{F}^{M+T}$ is the i-th column of encoding matrix $\mathbf{X} \in \mathbb{F}^{(M+T)\times N}$, i.e.,

$$\mathbf{X} = \begin{bmatrix} x_1^0 & x_2^0 & \cdots & x_N^0 \\ x_1^1 & x_2^1 & \cdots & x_N^1 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{M+T-1} & x_2^{M+T-1} & \cdots & x_N^{M+T-1} \end{bmatrix}. \tag{33}$$

We define $\mathbf{X} = [\mathbf{X}^{top}; \mathbf{X}^{bottom}]$, where $\mathbf{X}^{bottom} \in \mathbb{F}^{T\times N}$ and $\mathbf{X}^{top} \in \mathbb{F}^{M\times N}$ are the bottom and top submatrix of encoding matrix $\mathbf{X}$, respectively. Note that the $T \times T$ submatrix of the $\mathbf{X}^{bottom}$ is invertible. Hence, the encoding data $\tilde{\mathbf{A}}_{\mathcal{T}} = \hat{\mathbf{A}}\mathbf{X}^{top}_{\mathcal{T}} + \mathbf{R}\mathbf{X}^{bottom}_{\mathcal{T}}$ is any set of T colluding workers, where $\mathcal{T}$ denotes the set of the indexes of the T colluding workers, $\hat{\mathbf{A}} = [\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_M]$, and $\mathbf{R} = [\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_T]$. We can see that the encoded data $\hat{\mathbf{A}}\mathbf{X}^{top}_{\mathcal{P}}$ is fully masked by adding random padding $\mathbf{R}\mathbf{X}^{bottom}_{\mathcal{T}}$, where $\mathbf{R}\mathbf{X}^{bottom}_{\mathcal{T}}$ is uniformly random, and $\mathbf{X}^{bottom}_{\mathcal{T}}$ is invertible. Thus, the RSPCC scheme is T-private.

As aforementioned, our RSPCC scheme provides an S-resilient, A-secure and T-private for distributed convolution calculations. This completes the proof. ■

In LCC [24], there are $2A$ additional computed results required to check the correctness of the final result in the presence of A Byzantine workers. Unlike LCC, our RSPCC scheme provides a verification approach to check the correctness of the final result, only requiring an additional computed result.

Theorem 4: Given the mobile-assisted distributed computing system with a master and N workers, the proposed RSPCC scheme guarantees that each worker W_i for $i \in \mathcal{T}$ ($\mathcal{T} \subseteq \mathcal{N}$) cannot obtain any information about input matrix $\mathbf{A}$ from encoded matrices $\tilde{\mathbf{A}}_i$, i.e., the following privacy constraint is satisfied.

$$\mathbf{I}(\tilde{\mathbf{A}}_{\mathcal{T}}; \mathbf{A}) = 0. \tag{34}$$

Proof: We prove Theorem 4 in a similar manner to [39]. For any worker W_i for $i \in \mathcal{N}$, we have the mutual information

$$\begin{aligned}
\mathbf{I}(\mathbf{A}; \tilde{\mathbf{A}}_{\mathcal{T}}) &= \mathbf{H}(\tilde{\mathbf{A}}_{\mathcal{T}}) - \mathbf{H}(\tilde{\mathbf{A}}_{\mathcal{T}}|\mathbf{A}) && \text{(35a)} \\
&= \mathbf{H}(\tilde{\mathbf{A}}_{\mathcal{T}}|\mathbf{A}, \mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_T) \\
&\quad + \mathbf{H}(\tilde{\mathbf{A}}_{\mathcal{T}}) - \mathbf{H}(\tilde{\mathbf{A}}_{\mathcal{T}}|\mathbf{A}) && \text{(35b)} \\
&= \mathbf{H}(\tilde{\mathbf{A}}_{\mathcal{T}}) - \mathbf{I}(\tilde{\mathbf{A}}_{\mathcal{T}}; \mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_T|\mathbf{A}) && \text{(35c)} \\
&= \mathbf{H}(\tilde{\mathbf{A}}_{\mathcal{T}}) - \mathbf{H}(\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_T|\mathbf{A}) \\
&\quad + \mathbf{H}(\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_T|\mathbf{A}, \tilde{\mathbf{A}}_{\mathcal{T}}) && \text{(35d)} \\
&= \mathbf{H}(\tilde{\mathbf{A}}_{\mathcal{T}}) - \mathbf{H}(\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_T) && \text{(35e)} \\
&\leq \mathbf{H}(\tilde{\mathbf{A}}_{\mathcal{T}}) - \sum_{i=1}^{T} \mathbf{H}(\mathbf{R}_i) && \text{(35f)} \\
&= \mathbf{H}(\tilde{\mathbf{A}}_{\mathcal{T}}) - n^2 T \log|\mathbb{F}| && \text{(35g)} \\
&\leq \sum_{i=1}^{T} \mathbf{H}(\tilde{\mathbf{A}}_i) - n^2 T \log|\mathbb{F}| && \text{(35h)} \\
&= n^2 T \log|\mathbb{F}| - n^2 T \log|\mathbb{F}| \\
&= 0, && \text{(35i)}
\end{aligned}$$

where (35b) is derived by the fact that $\tilde{\mathbf{A}}_{\mathcal{T}}$ is a deterministic function of $\mathbf{A}$ and $\{\mathbf{R}_i\}_{i=1}^{T}$; (35e) follows the fact that random matrices $\{\mathbf{R}_i\}_{i=1}^{T}$ are independently generated from $\mathbf{A}$ by the master. (35f) and (35h) are due to the upper bounding of the joint entropy; (35g) due to the i.i.d. uniform random elements of $\{\mathbf{R}_i\}_{i=1}^{T}$; (35i) is derived using the similar method that employed in (35g). This completes the proof.

B. Complexity Analysis of the RSPCC Scheme

In this section, we analyze the computational complexities of the proposed RSPCC scheme for encoding, decoding, checking, communication and per-worker computation.

1) Encoding Complexity: For the encoding complexity, we need to evaluate the polynomial $\tilde{f}_{\mathbf{A}}(x_i)$ in (15). We can observe that there are $M + T$ matrices of dimension $n \times n$ summed in $\tilde{f}_{\mathbf{A}}(x_i)$. Thus, the computational complexity of encoding function $\tilde{f}_{\mathbf{A}}(x_i)$ for each subtask is $\mathcal{O}((M + T)n^2)$. Accordingly, the overall encoding complexity for N workers is $\mathcal{O}((M + T)Nn^2)$.

2) Decoding Complexity: Note that the decoding phase in the RSPCC scheme, the master decodes the final result $\mathbf{Y}$ by evaluating all coefficients of the polynomial $h(x)$ with degree $M+T-1$. In [40], the complexity of a $(k-1)$-th degree polynomial can be interpolated by polynomial interpolation algorithm is $\mathcal{O}(k\log^2 k \log\log k)$. Thus, in the RSPCC scheme, the decoding complexity of each coefficient matrix element is $\mathcal{O}((M+T)\log^2(M+T)\log\log(M+T))$. The interpolation is repeated for n^2 times for the coefficient matrices with dimension $n \times n$. Thus, the overall decoding complexity is $\mathcal{O}((n^2(M+T))\log^2(M+T)\log\log(M+T))$.

3) Result Checking Complexity: As described in Section III-B, we check the recovering computation results by (28). We can obtain the computational complexity of result checking is $\mathcal{O}(n^2(M+T))$.

4) Communication Complexity: The communication complexity consists of two parts: (1) master-to-worker and (2) worker-to-master. For the first part, master transmits $\mathcal{O}(n^2+v^2)$ symbols to each worker. Thus, the total number of symbols of the master transmits to N workers is $\mathcal{O}(N(n^2+v^2))$. For the second part, the worker that successfully completes the calculation, returns $\mathcal{O}(n^2)$ symbols to the master, and the recovery threshold is $\mathcal{O}(M+T)$. Thus, the total number of symbols required by the master to successfully recover the final result is $\mathcal{O}((M+T)n^2)$.

5) Computational Complexity of Each Worker: Each worker is assigned to compute a subtask, i.e., $\tilde{\mathbf{Y}}_i = \tilde{\mathbf{A}}_i \otimes \mathbf{W}$ for $i \in \mathcal{N}$. $\tilde{\mathbf{A}}_i$ and $\mathbf{W}$ have dimensions $n \times n$ and $v \times v$, respectively. From the ***Task Computing*** phase of Section IV-B, each worker has computational complexity $\mathcal{O}(n^2v^2/s^2)$.

VI. Experiments

In this section, we conduct experiments to evaluate the performance of the proposed RSPCC scheme for convolution calculation over distributed computing systems in terms of the execution time and success rate.

A. Experimental Settings

In our experiments, we implement the proposed RSPCC scheme using the mpi4py package [41] in Python on a cluster of 31 computing instances consisting of one master and 30 workers. Among $N=30$ workers, there are S straggling workers, T colluding workers, and A malicious workers. We provide specific values for these parameters later. In the system, S out of $N=30$ workers are randomly selected as straggling workers. We simulate the straggling effects and dynamic network delay in the practical computing scenario by adding artificial delays through the *sleep()* function of the *time* packet [42]. The available workers are selected by using random functions, e.g., *random.random()*.

Consider the presence of malicious workers with Byzantine attack in the system. We employ the following Byzantine attack models [43] in our experiments.

- *Gaussian Matrix Model:* In this model, the malicious worker always sends a matrix with the same dimensions as that of the valid workers, where each element of the matrix follows a Gaussian distribution $\mathcal{N}(0,\sigma^2)$.
- *Reversed Matrix Model:* In this model, the malicious worker always returns the reversed value back to the master. Specifically, the malicious worker sends the reversed result $-\nu\tilde{\mathbf{Y}}_i$, where each element of $\tilde{\mathbf{Y}}_i$ is multiplied by a negative scalar $\nu > 0$, to the master instead of sending $\tilde{\mathbf{Y}}_i$ for $i \in \mathcal{N}$. In our experiments, we set $\nu = 1$.
- *Constant Matrix Model:* In this model, the malicious worker always returns a constant matrix back to the master which has the same dimension as that of valid workers.

B. Experimental Results

1) Execution Time: We measure the execution time T_e of the proposed RSPCC scheme using two methods: computing a convolution operation task and training a convolutional neural network model on the the VegFru dataset [44]. We conduct the experiments by varying the number of straggling workers S based on the following four computation scenarios:

- *Scenario 1:* $N=30, A=2, T=3$ and $S=0$.
- *Scenario 2:* $N=30, A=2, T=3$ and $S=3$.
- *Scenario 3:* $N=30, A=2, T=3$ and $S=5$.
- *Scenario 4:* $N=30, A=2, T=3$ and $S=7$.

Furthermore, we compare the performance of our RSPCC scheme with the following two distributed computing schemes:

- *Scheme 1:* This scheme computes the convolution task using conventional distributed computing scheme, named, CDCS.
- *Scheme 2:* This scheme computes the convolution task using Winograd algorithm [45] over distributed computing framework, i.e., distributed Winograd-based convolution algorithm (DWCA).

Consider the RSPCC scheme over a finite field $\mathbb{F}$, but the dataset is defined in real domain. Hence, we quantize the dataset by the following function [46]:

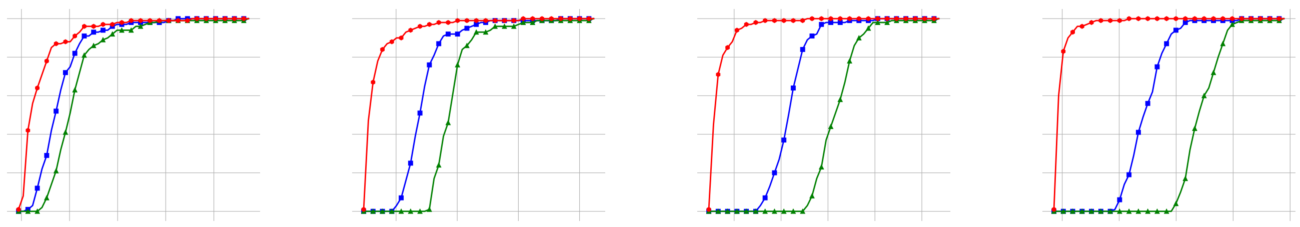
$$\bar{x} = \psi\big(\text{Round}(2^\ell \cdot x)\big), \tag{36}$$

where x denotes a floating point number, and ℓ is the quantization parameter. Function $\psi : \mathbb{Z} \to \mathbb{F}$ is defined to process an negative integer using two's complement representation over the finite field $\mathbb{F}$, is given by

$$\psi(x) = \begin{cases} x & \text{if } x \geq 0, \\ q + x & \text{if } x < 0. \end{cases} \tag{37}$$

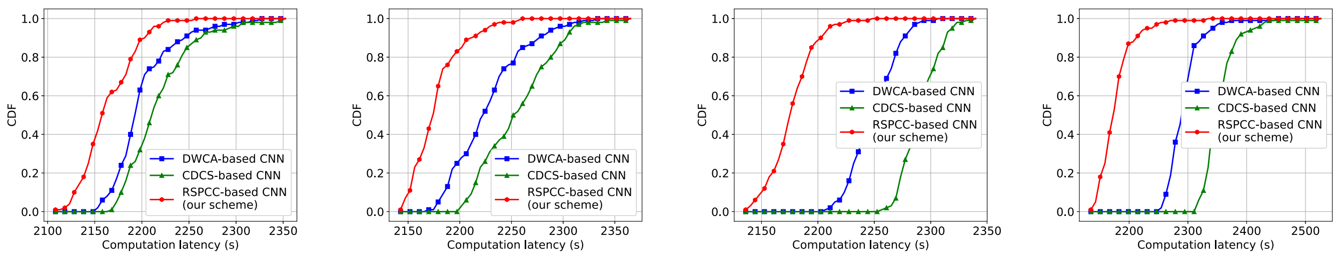
In order to avoid overflow issues, q should satisfy $q \geq 2^{\ell+1}\max\{|x|\}+1$.

The execution time is a crucial metric in a CDC system. We consider a convolution task with a 3×3 kernel on a 256×256 input image to evaluate the performance of the proposed RSPCC scheme. In our experiments, the master records the execution time of completing a convolution operation task using the CDCS, DWCA, and our RSPCC scheme for different scenarios. As depicted in Fig. 6, we compare the cumulative distribution function (CDF) of execution time T_e obtained by the CDCS, DWCA and our RSPCC scheme in four scenarios. We observe that the probability that the master completes the convolution operation



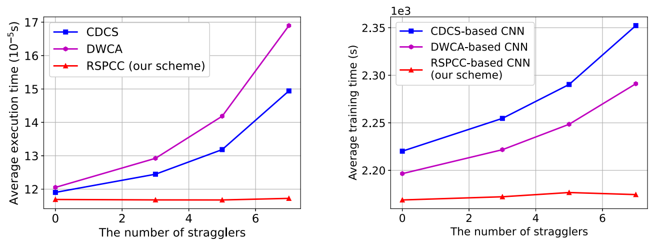
(a) Scenario 1 (b) Scenario 2 (c) Scenario 3 (d) Scenario 4

Fig. 6. Comparison of CDF of T_e for a volution operation task using the CDCS, DWCA and our RSPCC scheme in different four scenarios. Specifically, scenario 1 under the parameters $N = 30, A = 2, T = 3$ and $S = 0$, scenario 2 under the parameters $N = 30, A = 2, T = 3$ and $S = 3$, scenario 3 under the parameters $N = 30, A = 2, T = 3$ and $S = 5$, and scenario 4 under the parameters $N = 30, A = 2, T = 3$ and $S = 7$.



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3 (d) Scenario 4

Fig. 7. Comparison of CDF of T_e for training a convolutional neural network model using the CDCS-based CNN, DWCA-based CNN and RSPCC-based CNN in different four scenarios. Specifically, scenario 1 under the parameters $N = 30, A = 2, T = 3$ and $S = 0$, scenario 2 under the parameters $N = 30, A = 2, T = 3$ and $S = 3$, scenario 3 under the parameters $N = 30, A = 2, T = 3$ and $S = 5$, and scenario 4 under the parameters $N = 30, A = 2, T = 3$ and $S = 7$.



(a) CDC Schemes (b) CDC-Based CNNs

Fig. 8. Comparison of the average execution time obtained by the CDCS, DWCA and RSPCC (our scheme) for performing a convolution operation task and a image classification task, respectively, on a CDC system with $S = 0, 3, 5,$ and 7 straggling workers.

task using the proposed RSPCC scheme is higher than those of the other schemes in all scenarios for a given time t. Moreover, as the number of straggling workers increases, the probability of the master complete the same task using our RSPCC scheme is significantly higher than that of other schemes for a given time t. For instance, when $t = 1.24 \times 10^{-6}s$ in scenarios 3 and 4, the probabilities of the master completing the same task using the CDCS, DWCA and our RSPCC scheme are 0,0 and 9.9, respectively. In addition, we find that the distribution of T_e of the proposed RSPCC scheme is more concentrated than that of other schemes. That is because the RSPCC scheme provides resiliency against stragglers by using the encoding function.

Fig. 8(a) depicts the comparison of average execution time by CDCS, DWCA and the proposed RSPCC scheme for performing a convolution operation task under different numbers of straggling workers. The RSPCC scheme outperforms the CDCS and DWCA schemes in terms of average execution time under varying numbers of straggling workers. The average execution time of our RSPCC scheme saves up to 6.10% and 9.60% compared to DWCA and CDCS schemes, respectively, under $S = 3$ straggling workers. When $S = 5$, the average execution time of our RSPCC scheme saves up to 11.38% and 17.79% compared to DWCA and CDCS schemes, respectively. When $S = 7$, the average execution time of our RSPCC scheme saves up to 21.55% and 30.77% compared to DWCA and CDCS schemes, respectively. We observe that the RSPCC scheme can save more average execution time than that of the CDCS and DWCA schemes with the increase of the number of straggling workers. Furthermore, the average execution time of our RSPCC scheme remains steady under different number of straggling workers. That is because the RSPCC scheme can recover the final result without waiting for results from all worker. It also indicates that the proposed RSPCC scheme has strong capability on tolerating stragglers.

Second, we conducted the image classification experiments based on a CNN using the VegFru dataset [44], which contains 3600 fruit images with size 256×256. We randomly split the dataset into training and test datasets following $4 : 1$ ratio, i.e., 2880 images for training and 720 images for testing. The CNN comprises of an input layer, four convolutional layers, three pooling layers, two fully connected layers, and an output layer. Specifically, we compared the performance of three CNN models: CDCS-based, DWCA-based, and RSPCC-based. The conventional convolution operations of the two convolutional layers were replaced by the DWCA-based CNN and our RSPCC-based CNN scheme, respectively. The models were trained using mini-batch stochastic gradient descent (SGD) algorithm with a batch size of 128 and a learning rate of 0.01.

As depicted in Fig. 7, we compare the cumulative distribution function (CDF) of execution time T_e for training a convolutional neural network model by the CDCSR-based CNN, DWCA-based CNN and our RSPCC-based CNN scheme for the four different scenarios. Similar to Fig. 6, we observe that the probability of the master completing the model training task using the proposed RSPCC scheme is higher than those of the other schemes for all scenarios under the given time t. As the number of straggling workers increases, the probability of the master completing the same task by our RSPCC-based CNN scheme is significantly higher than that of other schemes within the given time t. For instance, when $t = 2210\,s$ in scenarios 3 and 4, the probabilities of the master completing the same training task on the CDCS-based CNN, DWCA-based CNN and our RSPCC-based CNN are 0,0 and 9.5, respectively.

Fig. 8(b) depicts a comparison of the average training time for the CDCS-based CNN, DWCA-based CNN, and RSPCC-based CNN under different numbers of straggling workers. It is observed that the average training time of RSPCC-based CNN is greatly shorter than that of the other schemes. Specifically, under $S = 3$ straggling workers, the average training time of our RSPCC-based CNN saves up to 12.07% and 18.92% compared to the DWCA-based CNN and CDCS-based CNN, respectively. Moreover, when $S = 5$, the average training time of our RSPCC scheme saves up to 16.72% and 24.11% compared to DWCA-based CNN and CDCS-based CNN, respectively. Similarly, when $S = 7$, the average training time of our RSPCC scheme saves up to 24.70% and 33.33% compared to the DWCA-based CNN and CDCS-based CNN, respectively. It is evident that using our RSPCC scheme significantly reduces the training time of the CNN. Furthermore, with an increase in the number of nodes, this trend becomes increasingly prominent. These results demonstrate the effectiveness of our proposed RSPCC scheme in significantly alleviating the straggler effects.

Our RSPCC scheme is capable of providing resilience against straggling workers due to the integration of coding techniques into the distributed computing process. As a result, the master can recover the original computation result successfully, without waiting for results from all workers. This significantly reduces the waiting time of the master, thereby enhancing the overall efficiency of the system.

2) Success Rate: In our master-worker system, not all workers can be trusted, as some workers, specifically, malicious nodes, may intentionally return inaccurate results to corrupt the final computation result. In our experiments, we randomly select $A = 0, 2, 4, 6, 8$ and 10 workers as malicious workers out of $N = 30$ workers. We perform a convolutional operation task 1000 times using the CDCS, DWCA, and our proposed RSPCC scheme. Fig. 9 illustrates the comparison of the average success rate achieved by the CDCS, DWCA, and our RSPCC scheme when performing the convolutional operation task under different numbers of malicious workers.

From Fig. 9, it can be observed that the RSPCC scheme achieves an average success rate exceeding 99% under different numbers of malicious workers. Furthermore, the average success rate achieved by the RSPCC scheme is considerably higher than that of the CDCS and DWCA schemes. As the number of malicious workers increases, the average success rates obtained by the CDCS and DWCA schemes decrease considerably. For instance, when the number of malicious workers is $A = 10$, the average success rates obtained by CDCS and DWCA schemes are less than 40%. It is evident that the RSPCC scheme surpasses the CDCS and DWCA schemes in terms of computational accuracy due to the implementation of verifiable computing at the master during the reception of worker results.

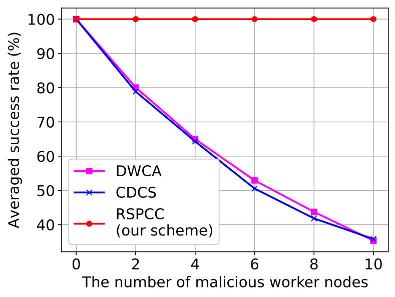


Fig. 9. Comparison of the success rate obtained by the CDCS, DWCA and RSPCC (our scheme) for performing a convolution computation task on a CDC system with $A = 0, 2, 4, 6, 8$ and 10 malicious workers.

VII. Conclusion

In this paper, we proposed a mobile-assisted coded distributed framework for the Metaverse. To provide a immersive experience for the Metaverse, especially on efficiently completing large-scale convolution computing tasks, we investigated the coded distributed convolution problem and designed a new coded distributed convolution scheme for resiliency, security and privacy. The proposed coding scheme, i.e., RSPCC scheme, which considered the existence of straggling, colluding and malicious workers simultaneously. The resiliency, security and privacy of the RSPCC scheme were demonstrated by the theoretical proofs. In addition, the RSPCC scheme integrated with MEC algorithm [16] to greatly reduce the memory consumption of the convolution operation. The optimized libraries: OpenBLAS, was utilized to speed up the matrix multiplication tasks. Our experimental results demonstrate that the proposed RSPCC scheme achieves high computing efficiency and reliability.

References

[1] D. Bega, M. Gramaglia, A. Banchs, V. Sciancalepore, and X. Costa-Perez, “A machine learning approach to 5G infrastructure market optimization,” *IEEE Trans. Mobile Comput.*, vol. 19, no. 3, pp. 498–512, Mar. 2020.

[2] Z. Lai, Y. C. Hu, Y. Cui, L. Sun, N. Dai, and H.-S. Lee, “Furion: Engineering high-quality immersive virtual reality on today’s mobile devices,” *IEEE Trans. Mobile Comput.*, vol. 19, no. 7, pp. 1586–1602, Jul. 2020.

[3] G. S. Park, R. H. Kim, and H. Song, “Collaborative virtual 3D object modeling for mobile augmented reality streaming services over 5G networks,” *IEEE Trans. Mobile Comput.*, vol. 22, no. 7, pp. 3855–3869, Jul. 2023.

[4] Y. Han and S. Oh, “Investigation and research on the negotiation space of mental and mental illness based on metaverse,” in *Proc. Int. Conf. Inf. Commun. Technol. Convergence*, 2021, pp. 673–677.

[5] X. Lin, J. Wu, J. Li, W. Yang, and M. Guizani, "Stochastic digital-twin service demand with edge response: An incentive-based congestion control approach," *IEEE Trans. Mobile Comput.*, vol. 22, no. 4, pp. 2402–2416, Apr. 2023.

[6] Y. Wang et al., "A survey on metaverse: Fundamentals, security, and privacy," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 1, pp. 319–352, First Quarter, 2023.

[7] M. Xu et al., "A full dive into realizing the edge-enabled metaverse: Visions, enabling technologies, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 1, pp. 656–700, First Quarter, 2023.

[8] Y. Wang and J. Zhao, "A survey of mobile edge computing for the metaverse: Architectures, applications, and challenges," in *Proc. IEEE 8th Int. Conf. Collaboration Internet Comput.*, 2022, pp. 1–9.

[9] Z. Chen, C. Cai, T. Zheng, J. Luo, J. Xiong, and X. Wang, "RF-based human activity recognition using signal adapted convolutional neural network," *IEEE Trans. Mobile Comput.*, vol. 22, no. 1, pp. 487–499, Jan. 2023.

[10] J. Yue and M. Xiao, "Coding for distributed fog computing in internet of mobile things," *IEEE Trans. Mobile Comput.*, vol. 20, no. 4, pp. 1337–1350, Apr. 2021.

[11] Y. Saputra, D. Nguyen, H. Dinh, Q.-V. Pham, E. Dutkiewicz, and W.-J. Hwang, "Federated learning framework with straggling mitigation and privacy-awareness for ai-based mobile application services," *IEEE Trans. Mobile Comput.*, vol. 22, no. 9, pp. 5296–5312, Sep. 2023.

[12] Y. Jiang et al., "Reliable distributed computing for metaverse: A hierarchical game-theoretic approach," *IEEE Trans. Veh. Technol*, vol. 72, no. 1, pp. 1084–1100, Jan. 2023.

[13] J. Ni, K. Zhang, Q. Xia, X. Lin, and X. S. Shen, "Enabling strong privacy preservation and accurate task allocation for mobile crowdsensing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 6, pp. 1317–1331, Jun. 2020.

[14] S. Zeng et al., "HFedMS: Heterogeneous federated learning with memorable data semantics in industrial metaverse," *IEEE Trans. Cloud Comput.*, vol. 11, no. 3, pp. 3055–3069, Third Quarter 2023.

[15] M. Corbett, J. Shang, and B. Ji, "GazePair: Efficient pairing of augmented reality devices using gaze tracking," *IEEE Trans. Mobile Comput.*, vol. 23, no. 3, pp. 2407–2421, Mar. 2024.

[16] M. Cho and D. Brand, "MEC: Memory-efficient convolution for deep neural network," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 815–824.

[17] N. Felemban et al., "PicSys: Energy-efficient fast image search on distributed mobile networks," *IEEE Trans. Mobile Comput.*, vol. 20, no. 4, pp. 1574–1589, Apr. 2021.

[18] S. Dutta, V. Cadambe, and P. Grover, "Coded convolution for parallel and distributed computing within a deadline," in *Proc. IEEE Int. Symp. Inf. Theory*, 2017, pp. 2403–2407.

[19] B. Zhou, J. Xie, and B. Wang, "Dynamic coded distributed convolution for UAV-based networked airborne computing," in *Proc. Int. Conf. Unmanned Aircr. Syst.*, 2022, pp. 955–961.

[20] H. Yang and J. Lee, "Secure distributed computing with straggling servers using polynomial codes," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 1, pp. 141–150, Jan. 2019.

[21] B. Zhu, S. Lin, Y. Zhu, and X. Wang, "Collaborative hyperspectral image processing using satellite edge computing," *IEEE Trans. Mobile Comput.*, vol. 23, no. 3, pp. 2241–2253, Mar. 2024.

[22] Y. Zhang, W. Sun, and M. Li, "WiRITE: General and practical Wi-Fi based hand-writing recognition," *IEEE Trans. Mobile Comput.*, vol. 23, no. 4, pp. 2943–2957, Apr. 2024.

[23] Y. Li et al., "Towards domain-independent and real-time gesture recognition using mmWave signal," *IEEE Trans. Mobile Comput.*, vol. 22, no. 12, pp. 7355–7369, Dec. 2023.

[24] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and S. A. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security, and privacy," in *Proc. 22nd Int. Conf. Artif. Intell. Statist.*, 2019, pp. 1215–1225.

[25] Q. Tong, Y. Miao, H. Li, X. Liu, and R. H. Deng, "Privacy-preserving ranked spatial keyword query in mobile cloud-assisted fog computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 6, pp. 3604–3618, Jun. 2023.

[26] X. Lin, J. Wu, J. Li, X. Zheng, and G. Li, "Friend-as-learner: Socially-driven trustworthy and efficient wireless federated edge learning," *IEEE Trans. Mobile Comput.*, vol. 22, no. 1, pp. 269–283, Jan. 2023.

[27] M. Li, Y. Song, and B. Wang, "CWCT: An effective vision transformer using improved cross-window self-attention and CNN," in *Proc. IEEE Conf. Virtual Reality 3D User Interfaces Abstr. Workshops*, 2022, pp. 149–154.

[28] R. Ran, L.-J. Deng, T.-X. Jiang, J.-F. Hu, J. Chanussot, and G. Vivone, "GuidedNet: A general CNN fusion framework via high-resolution guidance for hyperspectral image super-resolution," *IEEE Trans. Cybern.*, vol. 53, no. 7, pp. 4148–4161, Jul. 2023.

[29] K. Kawai, R. Rzepka, and T. Nemoto, "Using convolutional neural network for improving inference of interrogative sentences in a dialogue system," in *Proc. 10th Int. Conf. Affect. Comput. Intell. Interact. Workshops Demos*, 2022, pp. 1–4.

[30] Z. Peng, Y. Lu, S. Pan, and Y. Liu, "Efficient speech emotion recognition using multi-scale CNN and attention," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2021, pp. 3020–3024.

[31] X. Yang et al., "PICO: Pipeline inference framework for versatile CNNs on diverse mobile devices," *IEEE Trans. Mobile Comput.*, vol. 23, no. 4, pp. 2712–2730, Apr. 2024.

[32] T. Tan and G. Cao, "Deep learning video analytics through edge computing and neural processing units on mobile devices," *IEEE Trans. Mobile Comput.*, vol. 22, no. 3, pp. 1433–1448, Mar. 2023.

[33] J. Liang, L. Zhang, C. Han, C. Bu, H. Wu, and A. Song, "A collaborative compression scheme for fast activity recognition on mobile devices via global compression ratio decision," *IEEE Trans. Mobile Comput.*, vol. 23, no. 4, pp. 3259–3273, Apr. 2024.

[34] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," *IEEE Trans. Inf. Theory*, vol. 66, no. 3, pp. 1920–1933, Mar. 2020.

[35] S. Sahraei and A. S. Avestimehr, "INTERPOL: Information theoretically verifiable polynomial evaluation," in *Proc. IEEE Int. Symp. Inf. Theory*, 2019, pp. 1112–1116.

[36] C. Wang, Y. Xiao, X. Gao, L. Li, and J. Wang, "A framework for behavioral biometric authentication using deep metric learning on mobile devices," *IEEE Trans. Mobile Comput.*, vol. 22, no. 1, pp. 19–36, Jan. 2023.

[37] E. Rawashdeh, "A simple method for finding the inverse matrix of vandermonde matrix," *Matematički Vesnik*, vol. 71, pp. 207–213, 2019.

[38] Z. Lu, S. Rallapalli, K. Chan, S. Pu, and T. L. Porta, "Augur: Modeling the resource requirements of ConvNets on mobile devices," *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 352–365, Feb. 2021.

[39] H. Qiu and K. Zhu, "Secure and private approximated coded distributed computing using elliptic curve cryptography," in *Proc. Int. Conf. Collaborative Comput. Netw. Appl. Worksharing*, 2023, pp. 357–374.

[40] K. S. Kedlaya and C. Umans, "Fast polynomial factorization and modular composition," *SIAM J. Comput.*, vol. 40, no. 6, pp. 1767–1802, 2011.

[41] L. Dalcín, R. Paz, and M. Storti, "MPI for python," *J. Parallel Distrib. Comput.*, vol. 65, no. 9, pp. 1108–1115, 2005.

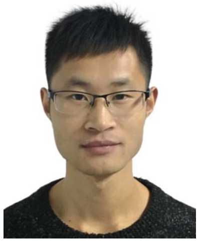
[42] Y. Chen, Q. Yang, S. He, Z. Shi, J. Chen, and M. Guizani, "FTPipeHD: A fault-tolerant pipeline-parallel distributed training approach for heterogeneous edge devices," *IEEE Trans. Mobile Comput.*, vol. 23, no. 4, pp. 3200–3212, Apr. 2024.

[43] S. D. Okegbile, J. Cai, and A. S. Alfa, "Practical Byzantine fault tolerance-enhanced blockchain-enabled data sharing system: Latency and age of data package analysis," *IEEE Trans. Mobile Comput.*, vol. 23, no. 1, pp. 737–753, Jan. 2024.

[44] J. Naranjo-Torres, M. Mora, R. Hernández-García, R. J. Barrientos, C. Fredes, and A. Valenzuela, "A review of convolutional neural network applied to fruit image processing," *Appl. Sci.*, vol. 10, no. 10, 2020, Art. no. 3443.

[45] A. Lavin and S. Gray, "Fast algorithms for convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4013–4021.

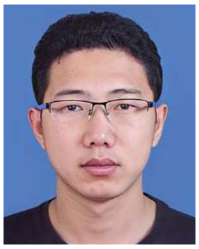
[46] J. So, B. Güler, and A. S. Avestimehr, "CodedPrivateML: A fast and privacy-preserving framework for distributed machine learning," *IEEE J. Sel. Areas Inf. Theory*, vol. 2, no. 1, pp. 441–451, Mar. 2021.



Houming Qiu received the ME degree from the Institute of Information and Control, Hangzhou Dianzi University, Hangzhou, China, in 2018. He is currently working toward the PhD degree with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. His research interests include distributed computing, coding theory, information theory, and secure/private computing.

Kun Zhu (Member, IEEE) received the PhD degree from the School of Computer Engineering, Nanyang Technological University, Singapore, in 2012. He was a research fellow with the Wireless Communications Networks and Services Research Group, University of Manitoba, Canada, from 2012 to 2015. He is currently a professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China. His research interests include resource allocation in 5G, wireless virtualization, and self-organizing networks. He has published more than seventy technical papers and has served as TPC for several conferences. He won several research awards including IEEE WCNC 2019 Best paper awards, ACM China rising star chapter award.

Dusit Niyato (Fellow, IEEE) received the BEng degree from the King Mongkuts Institute of Technology Ladkrabang (KMITL), Thailand, and the PhD degree in electrical and computer engineering from the University of Manitoba, Canada. He is a professor with the College of Computing and Data Science, Nanyang Technological University, Singapore. His research interests include the areas of mobile generative AI, edge intelligence, decentralized machine learning, and incentive mechanism design.



Bin Tang (Member, IEEE) received the BS and PhD degrees in computer science from Nanjing University, Nanjing, China, in 2007 and 2014, respectively. He was an assistant researcher with Nanjing University from 2014 to 2020, and also a research fellow with the Hong Kong Polytechnic University in 2019. He is currently a professor with Hohai University. His research interests include the area of communications, network coding, and distributed computing.