Encrypted Data Caching and Learning Framework for Robust Federated Learning-based Mobile Edge Computing

Chi-Hieu Nguyen, Yuris Mulya Saputra, Dinh Thai Hoang, Diep N. Nguyen, Van-Dinh Nguyen, Yong Xiao, and Eryk Dutkiewicz

***Abstract*—Federated Learning (FL) plays a pivotal role in enabling artificial intelligence (AI)-based mobile applications in mobile edge computing (MEC). However, due to the resource heterogeneity among participating mobile users (MUs), delayed updates from slow MUs may deteriorate the learning speed of the MEC-based FL system, commonly referred to as the straggling problem. To tackle the problem, this work proposes a novel privacy-preserving FL framework that utilizes homomorphic encryption (HE) based solutions to enable MUs, particularly resource-constrained MUs, to securely offload part of their training tasks to the cloud server (CS) and mobile edge nodes (MENs). Our framework first develops an efficient method for packing batches of training data into HE ciphertexts to reduce the complexity of HE-encrypted training at the MENs/CS. On that basis, the mobile service provider (MSP) can incentivize straggling MUs to encrypt part of their local datasets that are uploaded to certain MENs or the CS for caching and remote training. However, caching a large amount of encrypted data at the MENs and CS for FL may not only overburden those nodes but also incur a prohibitive cost of remote training, which ultimately reduces the MSP's overall profit. To optimize the portion of MUs' data to be encrypted, cached, and trained at the MENs/CS, we formulate an MSP's profit maximization problem, considering all MUs' and MENs' resource capabilities and data handling costs (including encryption, caching, and training) as well as the MSP's incentive budget. We then show that the problem is convex and can be efficiently solved using an interior point method. Extensive simulations on a real-world human activity recognition dataset show that our proposed framework can achieve much higher model accuracy (improving up to 24.29%) and faster convergence rate (by 2.86 times) than those of the conventional *FedAvg* approach when the straggling probability varies between 20% and 80%. Moreover, the proposed framework can improve the MSP's profit up to 2.84 times compared with other baseline FL approaches without MEN-assisted training.**

I. Introduction

Along with the ever-growing number of connected Internet of Things (IoT) devices, machine learning (ML) has been playing a critical role in learning and extracting knowledge from distributed data sources, e.g., from digital healthcare [1]–[3], Industry 4.0 [4]–[6], and the emerging Metaverse [7], [8]. Among various distributed learning frameworks, federated learning (FL) has been considered as the most potential one for the mobile edge computing (MEC) networks to reduce communication overhead and address the users' privacy issues [9], [10]. In this approach, each participating mobile user (MU) can first locally train an ML model by using its own dataset. Then, a central aggregator (e.g., a cloud server (CS)) will aggregate the local models (i.e., trained parameters) from the involved MUs to iteratively update the global model. At each iteration, the global model is first broadcast to all MUs for the local model update. This procedure is executed repeatedly until a certain level of accuracy of the global model is reached. By exchanging only the model parameters during the training process, the privacy of the user's data is preserved while saving the required bandwidth for raw local data sharing (otherwise).

Like all other distributed learning frameworks, FL also faces fundamental challenges due to the unreliable connectivity from the MUs to the server (particularly for the wireless connections), the inherent limitation of MUs' local computing resources [11]–[15], and the high-dimensional statistical characterization of non-i.i.d. local datasets [16], [17]. These lead to the well-known challenge, the *straggling problem* [15], in which the centralized global model aggregation is delayed or even stalled by the delayed update from one or several MUs (referred to as *stragglers*). The straggling problem can be caused by the insufficient computational capability of MUs to train their local models, in addition to unreliable wireless communication links to upload the local model parameters to the aggregator during each learning round. As a result, the FL may experience a substantial latency as the aggregator has to defer the aggregation process until it receives the local models from all involved MUs in every learning round.

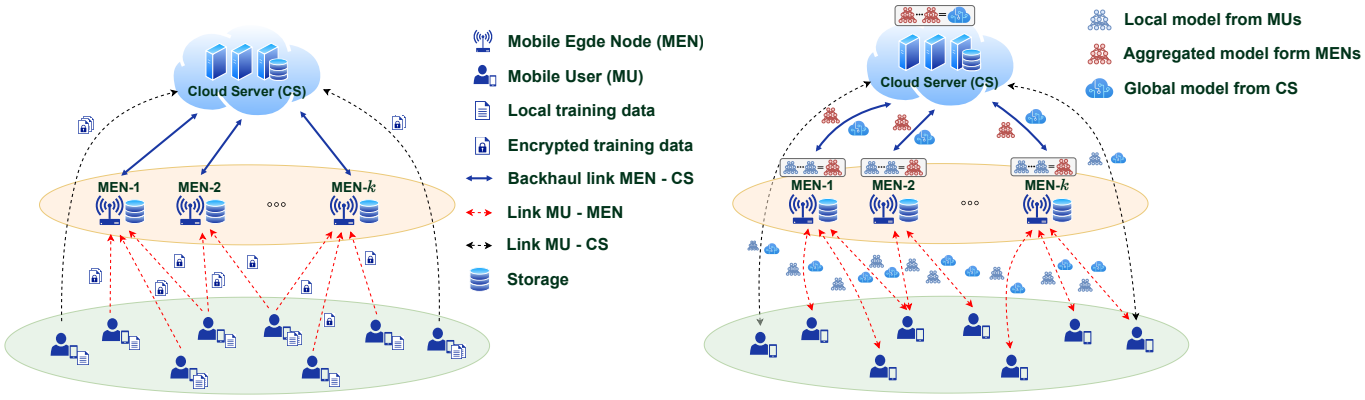
To address the above issues, several solutions focusing on data caching and computing capabilities using mobile edge nodes (MENs) in the vicinity of MUs have been investigated. One possible solution is to allow the MUs with low computation and communication resources to offload their local data to an MEN, thereby reducing the workload on those MUs. In particular, the works in [18]–[20] suggested local data offloading from MUs to a cloud or edge servers where the FL model training is carried out. However, these works assume that each MU needs to upload its whole dataset

Chi-Hieu Nguyen, D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz are with the School of Electrical and Data Engineering, University of Technology Sydney, Sydney, NSW 2007, Australia (e-mail: hieu.c.nguyen@student.uts.edu.au, {hoang.dinh, diep.nguyen, eryk.dutkiewicz}@uts.edu.au).

Y. M. Saputra is with Department of Electrical Engineering and Informatics, Vocational College, Universitas Gadjah Mada, Yogyakarta 55281, Indonesia (e-mail: ym.saputra@ugm.ac.id).

Van-Dinh Nguyen is with the College of Engineering and Computer Science, VinUniversity, Vinhomes Ocean Park, Hanoi 100000, Vietnam (e-mail: dinh.nv2@vinuni.edu.vn).

Y. Xiao is with the School of Electronic Information and Communications at the Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: yongxiao@hust.edu.cn).



(a) Pre-training phase

(b) Iterative training phase

Fig. 1: The proposed privacy-preserving FL framework with the additional MEN-assisted training: (a) pre-training phase and (b) iterative training phase.

to an MEN or the CS. Such a data-sharing approach not only raises MUs' data privacy concerns but also is infeasible to deploy due to the limited communications resources of MUs (e.g., the local dataset is too large to be uploaded). To address the privacy concern, encryption-based techniques, e.g., homomorphic encryption (HE), can be used to secure offloaded data as proposed in [21] and [22]. HE in [23] is a form of public-key encryption that permits certain computations, such as additions and multiplications, to be conducted directly on ciphertexts without requiring the private key. When two ciphertexts are computed with HE, the result is another ciphertext, and its decryption produces the outcome of the corresponding operations on the original data. Hence, users only need to upload their encrypted data to the untrusted server for secure storage and processing. Nevertheless, most existing works based on HE approach, e.g., [21], [22], are restricted to using a single learning node, such as a CS or MEN, to assist the MUs in training the ML model over the encrypted dataset. As a result, an MEN may face challenges in handling a massive amount of data from several MUs due to its inherent resources limitation. At the same time, some MUs can also experience high communication costs if all their encrypted data is directly offloaded to the CS [24].

In this work, we propose a novel MEC-assisted FL framework that incorporates HE-based training executions at all available MENs and the CS to address the straggling problem while preserving the MU's privacy. More specifically, given a pre-determined deadline for each learning round and the resource capabilities of the participating MUs, the mobile service provider (MSP) can decide the percentage of data to be used for local training at each MU without causing an excessive delay. The remaining local data is encrypted by the MUs using HE and then shared with the MSP for remote training. As illustrated in Fig. 1(a), all encrypted data are uploaded and cached at the destined MENs and the CS prior to the FL training. Subsequently, in each learning round, all the MENs and the CS perform additional training on their encrypted datasets and produce the encrypted models. All the trained models from the MUs and MENs are then uploaded to the CS for updating the global model (as illustrated in Fig. 1(b)). The MUs then receive incentives (e.g., monetary rewards) from the MSP in return for sharing their local data and performing local model training.

The advantage of our proposed FL framework is multi-fold. First, through monetary rewards to the participants, the MSP can incentivize the MUs to devote their computation and data resources to the FL process, thereby improving the accuracy of the whole system. Second, it can relieve the computational burden on resource-constrained MUs and resolve the straggling problem thanks to the MEN-aided training processes. Third, by collecting encrypted data of different distributions from multiple MUs to train models at the MENs and CS, the framework can counteract the bias that arises from non-i.i.d. data in practical settings. This leads to significant improvements in learning performance and resilience of the FL framework.

Nonetheless, a major challenge in implementing such desired HE-based FL framework is the computational overhead caused by HE operations, which may significantly retard the model training at the MENs/CS. To overcome this barrier, we develop an efficient HE ciphertext packing method together with the single instruction multiple data (SIMD) techniques to accelerate the neural network processing with encrypted data. We then formulate an optimization problem to evaluate the proper percentage of data to be encrypted and cached at MENs. The objective is to maximize the total profit of the MSP throughout the FL training process, while considering the duration of each learning round, the MSP's incentive budget, the available resources at MUs and MENs, as well as the data handling costs caused by encryption, caching, and training. We then prove that the resulting optimization problem is convex and can be solved using an interior point method. The experimental results demonstrate that our privacy-preserving FL framework accelerates convergence speed by 2.86 times and enhances accuracy levels up to 24.29% in non-i.i.d. data scenarios compared to the conventional FL approach. Additionally, the proposed framework can enhance the profit of the MSP by 2.84 times in comparison with other baseline approaches. Our main contributions can be summarized as follows:

- Propose a novel privacy-preserving FL framework for MEC networks that incorporates encrypted training processes at both the CS and MENs to mitigate the straggling problem. Here, a tunable amount of the MUs' raw data is encrypted using HE before caching to the MENs/CS to allow further processing without exposing sensitive information. Additionally, the local training updates from the MUs are encrypted to prevent the FL framework from being vulnerable to the gradient leakage attack.
- Develop an efficient ciphertext packing based on the CKKS encryption scheme to enable SIMD parallel computation at the MENs and CS. The proposed technique facilitates the acceleration of additional training processes over encrypted data.
- Formulate an encrypted data caching optimization problem for the MEC-enabled FL to obtain the optimal portions of MUs' data to be encrypted and cached at MENs/CS. The objective is to maximize the MSP's profit while considering the restricted processing capabilities of MUs and MENs, duration of each learning round, the limited incentive budget of the MSP, and all data handling costs. The resulting optimization function is proven to be convex and hence can be effectively solved with the interior point method.
- Conduct extensive experiments on a real-world human activity recognition dataset [25]. These results provide insightful information to help the MSP in designing the effective privacy-preserving FL framework in the MEC networks.

The rest of this paper is organized as follows. Section II presents the related works. Sections III and IV introduce the proposed system model and the detailed implementation of the proposed FL framework. The problem formulation and our approach to obtain the optimal encrypted data caching and learning solution are presented in Section V. Section VI demonstrates the experimental results. Finally, Section VII presents the conclusion and future directions of the work.

II. Related Work and Main Contributions

A. Straggling Mitigation in Federated Learning

The heterogeneity of communication and computation capability among participating devices poses a significant challenge when FL framework is implemented in real-world wireless networks, referred to as the straggling problem. To mitigate that problem, three appealing approaches are often considered in the literature: (i) *asynchronous updating*, (ii) *computation offloading*, and (iii) *coded computing*.

1) Asynchronous updating: In the asynchronous updating approach, the CS only needs to wait until a certain number of participants have finished uploading the model updates before starting the aggregating operation [16], [17], [26], [27]. By utilizing this updating strategy, the straggling clients no longer hinder the aggregation process, thereby reducing delay of the system. For example, the authors in [28] provided empirical findings that an asynchronous strategy is resilient to clients participating partway during a training round as well as when the FL includes clients with diverse processing capabilities. In [12], the authors proposed a tier-based FL method where the clients are split into several tiers based on their response delays. The global model is then updated across tiers asynchronously, while the model update within a tier is performed in a synchronous fashion. However, this mechanism may result in high-bandwidth utilization as they require frequent model exchanges between the aggregating servers and FL clients. Another challenge in applying the asynchronous updating approach is that the uploaded clients' models might be calculated from different versions of the global model, which can deteriorate the convergence of the global model [17]. The FedAsync algorithm in [26] suggested to limit such adverse effect by utilizing the client's staleness measurement to determine the weight that should be given to the recently uploaded local model in the aggregation step. Nonetheless, it is shown that when the input is non-i.i.d. and imbalanced, the asynchronous approach may significantly deteriorate the model's convergence due to the unequal contribution of FL clients to the training process [13], [28]. Furthermore, all of the aforementioned studies have neglected the potential of using the computational power of MENs to support the FL training of mobile users.

2) Computation offloading: Several works in [14], [18]–[20] have explored the effectiveness of local data offloading from MUs to a cloud or edge server to mitigate the straggling effect. In [14], the authors proposed an effective edge-assisted FL scheme, called EAFL, where the optimal data size for offloading at each FL client is determined by solving a non-convex optimization problem. However, since the EAFL scheme requires MUs to share their raw data with remote servers, it violates a fundamental benefit of the FL framework, i.e., privacy conservation. The authors in [29] developed FedAdapt, a comprehensive framework that accelerates the FL training process through allowing FL stragglers to offload specific layers of their local neural network to a remote server for training. FedAdapt employs a reinforcement learning algorithm to identify optimal neural network offloading partitions for each client. However, this framework may lead to a significant increase in bandwidth utilization due to the frequent exchange of gradients and labels between remote servers and FL clients. Furthermore, transmitting local gradients without proper protection or encryption can potentially leak sensitive information from the users' raw data.

3) Coded computing: Coded computing is another potential approach to mitigate the straggling problem in FL by introducing redundancy in the calculations [15], [30]–[35]. Particularly, this approach aims to obtain the overall result of the desired computation task from a subset of participating workers by establishing additional allocation and computation over distributed data [30]–[32], thereby reducing the waiting time and improving the reliability.

Specifically, [32] is one of the first works that apply coded computing in FL to learn from decentralized datasets. However, the proposed solution was strictly applicable to linear regression models. This limitation was then addressed by [15], which extended the solution to enable non-linear model training. Moreover, [33] considers applying the coded FL approach for hierarchical setups such as multi-access edge computing

environments. The authors propose two coding approaches, i.e., Aligned Repetition Coding (ARC) and Aligned Minimum Distance Separable Coding (AMC), leveraging redundancy in communication links. In light of this, the authors in [34] propose to use the layered MDS codes to achieve a good transition between the ARC and AMC schemes. However, both works in [33] and [34] only cover the communication aspect of the FL system, neglecting the computing capability of FL clients which is another cause of the straggling problem. Furthermore, the authors in [35] propose the *CodedPaddedFL* and *CodedSecAgg* schemes, which are regarded as the state-of-the-art work using coded computing paradigm. These schemes adopt coding techniques to mitigate the impact of client dropout (an extreme case of straggling) and employ a secure aggregation protocol to enhance security. They are designed specifically for linear regression problems but can be also adaptable to non-linear problems using kernel embedding.

The coded FL approach has demonstrated its effectiveness in addressing the straggling problem by introducing redundancy in the computing task. However, a core limitation of existing coded computing schemes is the need for clients to share parity data with untrusted servers, risking privacy leakage. To address these privacy concerns, some recent works propose additional methods to preserve data privacy before sharing. For instance, Sun *et al.* [36] propose enhancing the privacy protection of the parity dataset by injecting Gaussian noise to the coded data. However, this method introduces an inherent privacy-performance tradeoff, as larger additive noise levels can cause biased gradient estimates. In the *CodedPaddedFL* scheme [35], authors combine coded computing with the one-time padding technique to provide data privacy. However, this scheme remains vulnerable to gradient inversion attacks due to the exposure of raw local gradients to the central server. Alternatively, their proposed *CodedSecAgg* scheme [35] addresses the information leakage issue by leveraging Shamir's secret sharing scheme. While this mechanism prevents information leakage, it necessitates an intricate aggregation protocol that incurs extra communication overhead per learning round.

B. Neural Network (NN) for Training Encrypted Data

Most of the existing works (i.e., [21], [37]–[42]) investigate the training of NNs on encrypted data rely on homomorphic encryption (HE). An example of HE implementation is the CKKS scheme [43], which can support approximate computation with floating-point numbers. Additionally, the CKKS scheme employs SIMD techniques for parallel computation by packing multiple plaintexts into a single ciphertext [43]. As such, CKKS is considered the most favorable HE scheme for securing neural network processing [44].

Nandakumar *et al.* [38] made an initial effort to train NNs on encrypted data using HE. Their approach enables a user to encrypt data with a public key and send it to a remote server for training. The server is unable to extract any information about the data, and only the data owner with the private key can decrypt the final model. However, their solution necessitates continuous communication between the server and the client to refresh the encrypted parameters when the noise resulting from the HE operations reaches a certain threshold. To address this issue, Hesamifard *et al.* [37] proposed a noninteractive NN training scheme that utilizes HE and employs the bootstrapping technique to alleviate the noise in the ciphertext. Both works [38] and [37] employ bit-wise HE, which operates on individual bits of the input data. However, it is widely acknowledged that bit-wise HE is less computationally efficient compared to word-wise HEs such as the CKKS scheme, which enable packing to perform simultaneous operations on multiple integer or real inputs. Additionally, [39] introduced techniques for evaluating CNNs on encrypted data by replacing the nonlinear activation functions with low-degree polynomials that involve only addition and multiplication operations.

Meftah *et al.* [40] introduced Doren, a method that enhances amortized performance and reduces ciphertext expansion by employing a single instruction, SIMD packing technique. Doren utilizes the BGV scheme, which supports encryption and homomorphic operations on vectors of bits. Nonetheless, this work focuses solely on model inference and does not consider the training aspect. Lee *et al.* [41] made an effort to implement very deep neural networks, such as ResNet-20, with high accuracy using HE. They employed the RNS-CKKS scheme, a variant of the CKKS scheme, to accelerate the HE operations. Similarly, Kim *et al.* [42] designed a framework for secured skeleton-based action recognition by employing a FHE-compatible CNN. Both works [41] and [42] concentrate on efficiently implementing HE-based neural networks to achieve low latency and high inference throughput, assuming the availability of pre-trained models on plaintext datasets.

Unlike the aforementioned works, which primarily focus on HE-based inference on centralized servers, our work tackles the challenge of training neural networks using encrypted datasets. We propose a novel method for SIMD data packing that is specifically tailored for the training task. Moreover, to the best of our knowledge, no prior research has explored the deployment of encrypted training processes at MENs to address the straggling problem in MEC-enabled FL systems. Therefore, our study aims to fill this gap in the literature and explore the feasibility and effectiveness of this approach.

III. System Model

A. Overview of the Proposed Privacy-Preserving FL Framework

We consider an MEC-based FL system as illustrated in Fig. 1 where a CS and multiple MENs are controlled by an MSP to deliver services to various MUs. Let $\mathcal{K} = \{1, \ldots, k, \ldots, K\}$ denote the set of MENs with MEN-K being considered as the CS. Typically, the CS is integrated with a macrocell base station and has abundant computational and storage resources. The other MENs, numbered from 1 to $K-1$, are assumed to have limited computational power and constrained storage capacity [20], [24]. Additionally, we define $\mathcal{N} = \{1, \ldots, n, \ldots, N\}$ as the set of MUs participating in the network's FL process. An MU is in the serving area of MEN-k if they are connected directly via a wireless link (e.g., Wi-Fi). It is important to note that an MU can be

concurrently connected to several MENs. Meanwhile, some MUs may establish direct communication with the CS directly through cellular networks, e.g., 4G or 5G. TABLE I provides an overview of the notations utilized in this paper.

Fig. 2 illustrates the overall architecture of the proposed framework. During the pre-training phase, each MU shares its capability profile with the CS. This profile includes metrics such as processing frequency, CPU cycles per sample, and the number of successful uplink/downlink transmissions. These details can be derived from statistical records and serve as parameters in the proposed system model described in Section III.B. By utilizing these profiles, the CS optimizes the portion of cached training data from each MU to satisfy the deadline constraints in each learning round, as described in the optimization problem introduced in Section V. This approach mitigates the occurrence of straggling updates by assigning workloads that align with the capabilities of the devices.

Subsequently, each participating MU is allowed to upload its local dataset (a portion or all) to the available MENs or CS for caching and remote training (as illustrated in Fig. 1(a)) so as to relax its computation burden and accelerate the learning process. In order to maintain data privacy, the CKKS homomorphic encryption scheme [43] is used to encrypt the local data. Such an encryption technique enables a third party (e.g., MEN/CS) to train a conventional deep learning model over the data in an encrypted format without having to decrypt it [37]. Specifically, the CKKS scheme implemented at an MU includes the key generation, encryption/decryption, and homomorphic evaluation algorithms as follows:

- $SKGen(n)$ to randomly generate the secret key g_n^{sk} for MU-n.
- $PKGen(g_n^{sk})$ to create the public key g_n^{pk} for MU-n based on the secret key g_n^{sk}.
- $Enc(g_n^{pk}, \pi)$ to encrypt a plain vector π into a ciphertext $\tilde{\pi}$ using the public key g_n^{pk}.
- $Dec(g_n^{sk}, \tilde{\pi})$ to retrieve original vector π from the ciphertext $\tilde{\pi}$ using the secret key g_n^{sk} .
- $Add(\tilde{\pi}_1, \tilde{\pi}_2)$, $Sub(\tilde{\pi}_1, \tilde{\pi}_2)$ and $Mul(\tilde{\pi}_1, \tilde{\pi}_2)$ to, respectively, perform element-wise addition, subtraction, and multiplication between two ciphertexts $\tilde{\pi}_1$ and $\tilde{\pi}_2$. The respective outputs $\tilde{\pi}_{add}$, $\tilde{\pi}_{sub}$, and $\tilde{\pi}_{mul}$ are also ciphertexts where:

$$Add(\tilde{\pi}_1, \tilde{\pi}_2) = \tilde{\pi}_{add} \text{ s.t. } Dec(g_n^{sk}, \tilde{\pi}_{add}) \approx \pi_1 + \pi_2, \quad (1)$$

$$Sub(\tilde{\pi}_1, \tilde{\pi}_2) = \tilde{\pi}_{sub} \text{ s.t. } Dec(g_n^{sk}, \tilde{\pi}_{sub}) \approx \pi_1 - \pi_2, \quad (2)$$

$$Mul(\tilde{\pi}_1, \tilde{\pi}_2) = \tilde{\pi}_{mul} \text{ s.t. } Dec(g_n^{sk}, \tilde{\pi}_{mul}) \approx \pi_1 \times \pi_2. \quad (3)$$

  Here, $\pi_1 \times \pi_2$ is the cross product between two vectors π_1 and π_2. In other words, homomorphic operations on ciphertexts $\tilde{\pi}_1$ and $\tilde{\pi}_2$ produce ciphertexts that decrypt to the desired arithmetic result. This allows computations directly on encrypted data.

Suppose that each MU-n has a private local dataset $\mathbf{\Omega}_n = \left(\mathbf{F}_n, \mathbf{y}_n\right)$ consisting of $|\mathbf{\Omega}_n|$ data samples, where $\mathbf{F}_n$ and $\mathbf{y}_n$ are respectively the collections of training samples and labels. In particular, we have

$$\mathbf{F}_n = \begin{pmatrix} \mathbf{f}_n^1 \\ \vdots \\ \mathbf{f}_n^j \\ \vdots \\ \mathbf{f}_n^{|\mathbf{\Omega}_n|} \end{pmatrix}, \ \mathbf{y}_n = \begin{pmatrix} y_n^1 \\ \vdots \\ y_n^j \\ \vdots \\ y_n^{|\mathbf{\Omega}_n|} \end{pmatrix}, \quad (4)$$

where $\mathbf{f}_n^j, \ j \in \{1, \ldots, |\mathbf{\Omega}_n|\}$ is a feature vector that is associated with a label y_n^j. Based on the aforementioned cryptographic algorithms, each MU-n can generate the secret key $g_n^{sk} = SKGen(n)$ and public key $g_n^{pk} = PKGen(g_n^{sk})$ at the beginning of the learning process. The secret key g_n^{sk} is stored privately at the MU-n, while the public key g_n^{pk} of MU-n is broadcasted to other MUs and the MENs/CS. After that, each MU-n splits its local dataset $\mathbf{\Omega}_n$ into multiple subdatasets $\mathbf{\Omega}_{n,k} = \left(\mathbf{F}_{n,k}, \mathbf{y}_{n,k}\right)$ containing $|\mathbf{\Omega}_{n,k}|$ samples, where $k \in \{0, \ldots, K\}$, such that

$$|\mathbf{\Omega}_{n,0}| + \ldots + |\mathbf{\Omega}_{n,K}| = |\mathbf{\Omega}_n|, \quad (5)$$

and

$$\mathbf{F}_{n,k} = \begin{pmatrix} \mathbf{f}_n^{\Phi_k} \\ \vdots \\ \mathbf{f}_n^{\Phi_k + |\mathbf{\Omega}_{n,k}|} \end{pmatrix}, \ \mathbf{y}_{n,k} = \begin{pmatrix} y_n^{\Phi_k} \\ \vdots \\ y_n^{\Phi_k + |\mathbf{\Omega}_{n,k}|} \end{pmatrix}, \ \forall k \in \{0, \ldots, K\}. \quad (6)$$

Here, we define $\Phi_0 = 1$ and $\Phi_k = \Phi_{k-1} + |\mathbf{\Omega}_{n,k-1}| + 1, \ \forall k \in \{1, \ldots, K\}$. An MU-$n$ then utilizes its public key g_n^{pk} to generate K encrypted subdatasets $\tilde{\mathbf{\Omega}}_{n,k}$, where $k = 1, \ldots, K$, by performing encryption on each data sample and label in $\mathbf{\Omega}_{n,k}$ individually. The encrypted dataset $\tilde{\mathbf{\Omega}}_{n,k}$ is then uploaded to the MEN-k for the accumulation process prior to the iterative training execution. The aforementioned data encryption and caching process are the final steps in the initial pre-training phase.

Once the pre-training phase completes, the iterative FL training phase is carried out as illustrated in Fig. 1(b). Particularly, each MEN acts as an intermediate aggregator to collect the local model updates from its associated MUs and sends the aggregated results to the CS for updating the global model. As such, the CS can be regarded as a master coordinator that will aggregate the trained models from all participating entities in the FL system, i.e., MENs and MUs. The CS and MENs also perform their assigned model training tasks using the encrypted datasets received from the pre-training phase. Additionally, during a learning round, each MU locally trains its current model over the unencrypted data (i.e., the dataset portion that is not offloaded to an external node) before uploading the model update directly to the CS or through an intermediate MEN. The model training time at an MU or MEN in each learning round is limited by a predefined system's threshold. Then, each MEN can collect the MUs' trained models within its serving area and combine them together with its own trained model to produce an MEN-aggregated model. All MEN-aggregated models are then uploaded to the CS to perform the final aggregation step, which produces a

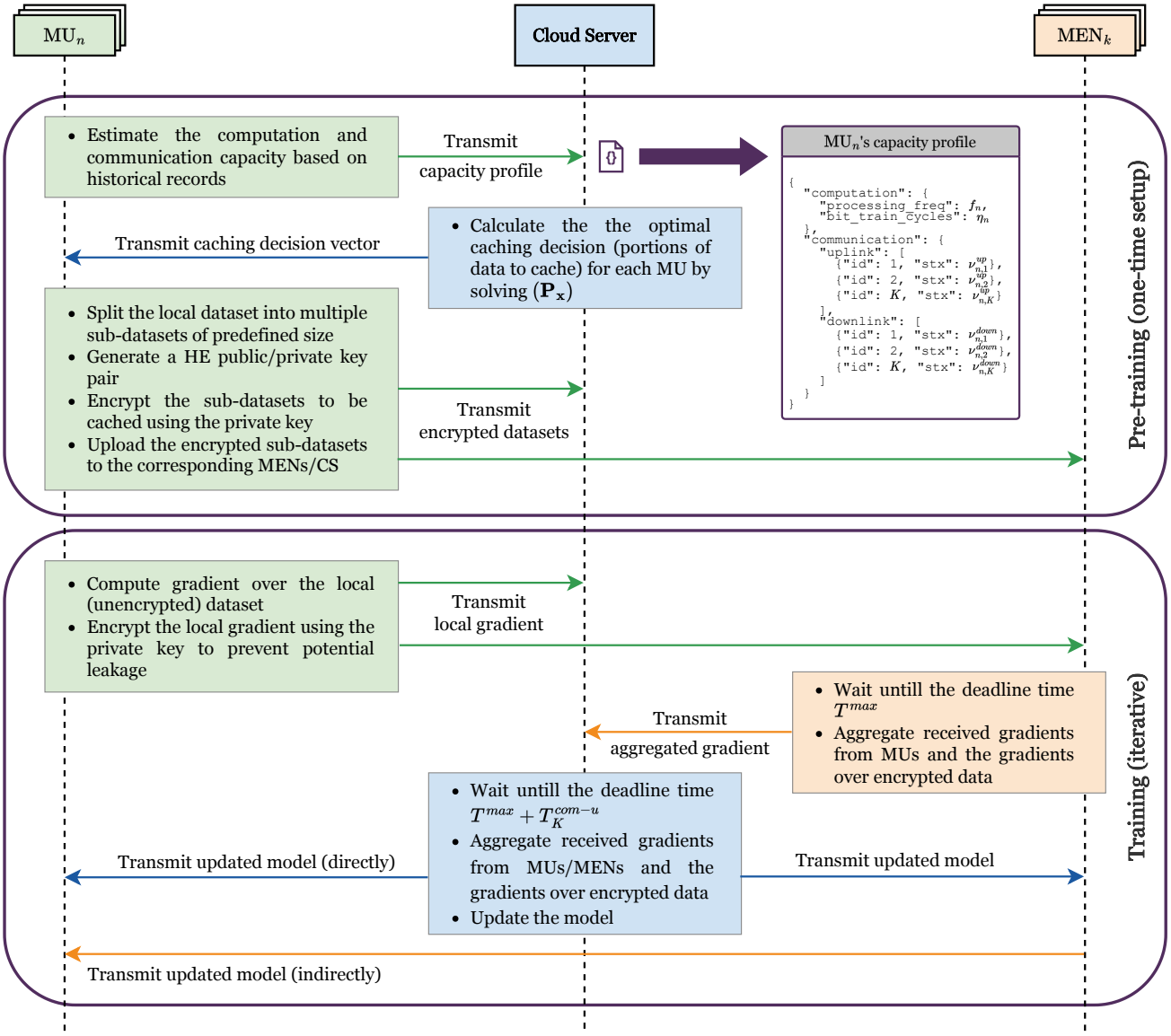


Fig. 2: Illustration of the main processing steps in the proposed framework. These steps are organized into two distinct phases: one-time pre-training and iterative training.

new global model. It is noteworthy that the CS and the MENs are set up to only aggregate local model updates from MUs collected within a predetermined training time threshold.

We assume that a sample in each local dataset $\mathbf{\Omega}_n$ has a fixed size of ξ bits; thus the total size of a dataset $\mathbf{\Omega}_n$ is defined as $b_n = \xi|\mathbf{\Omega}_n|$ (bits). A continuous variable $x_{n,k}$, where $0 \leq x_{n,k} \leq 1$, is used to determine the portion of the local dataset at MU-n to be encrypted and cached at MEN-k (with MEN-K referred to as the CS). The portion of dataset that would be trained locally at MU-n is denoted as $x_{n,0}$, where $x_{n,0} = \left(1 - \sum_{k=1}^{K} x_{n,k}\right)$. Let $\mathbf{x}$ denote the vector containing all variables $x_{n,k}$, i.e., $\mathbf{x} = [x_{1,1}, \ldots, x_{1,k}, \ldots, x_{1,K}, x_{2,1} \ldots, x_{n,k}, \ldots, x_{N,K}]$. We also define the bandwidth between MU-n and MEN-k, the bandwidth between MU-n and the CS, and the bandwidth between MEN-k and the CS as $B_{n,k}$, $B_{n,K}$, and $B_{k,K}$, respectively. Due to the limited computing resources, each participating MU-n is only able to perform local training over a dataset of size up to c_n during the whole FL process [11], [15]. As a result, the MU-n must cache a portion of its private dataset $\mathbf{\Omega}_n$ to the selected MENs or CS if $b_n > c_n$. Similarly, the computing resources of an MEN-k $(1 \leq k < K)$ are only sufficient to train an encrypted dataset up to size c_k that is gathered from all the MUs.

B. Computation and Communication Model

In the proposed FL system, each MU-n can utilize its own processor with processing frequency of f_n (Hz) to train a portion of dataset with size $x_{n,0}b_n$ bits. Let η_n be the number of CPU cycles needed for training 1-bit of data at MU-n, and thus the computation time required for training the local model at MU-n at a learning round can be calculated as follows:

$$T_n^{cmp} = \frac{\eta_n x_{n,0} b_n}{f_n}. \tag{7}$$

Notation	Description
$\mathcal{N}$	Set of MUs
$\mathcal{K}$	Set of MENs
$\mathbf{\Omega}_n$	Local dataset at MU-n
b_n	Size of local dataset at MU-n
$x_{n,k}$	Portion of the dataset at MU-n to be cached and trained at MEN-k
$x_{n,0}$	Portion of the dataset at MU-n to be trained locally
$B_{n,k}$	Bandwith between MU-n - MEN-k
$B_{n,K}$	Bandwith between MU-n - CS
$B_{k,K}$	Bandwith between MEN-k - CS
c_n	Maximum size of dataset can be trained by MU-n
c_k	Maximum size of dataset can be trained by MEN-k
T_n^{cmp}	Computation time required for training the local dataset at MU-n
T_k^{cmp}	Computation time required for training the encrypted dataset at MEN-k
$s_{\bar{\mathbf{\Gamma}}^{(r)}}$	Size of the global model at learning round r
T^{max}	Deadline time for all MENs to start uploading the aggregated models
T_K^{com-u}	Time delay for uploading an MEN-aggregated model to the CS from an MEN
P_n	MSP's profit for the local training process at MU-n
P_k	MSP's profit for the encrypted training process at MEN-k

TABLE I: Summary of Notations

Likewise, the encrypted training process carried out at an MEN-k is achieved using its processor's frequency f_k (Hz) which requires η_m (cycles/bit) for 1-bit data training. Since the total encrypted dataset collected at MUs has size of $\sum_{n=1}^{N} x_{n,k} b_n$ (bits), the computation time of an MEN-k can be obtained by

$$T_k^{cmp} = \frac{\eta_k \sum_{n=1}^{N} x_{n,k} b_n}{f_k}. \tag{8}$$

Regarding the communication model, let $s_{\bar{\mathbf{\Gamma}}^{(r)}}$, $s_{\mathbf{\Gamma}_n^{(r)}}$, and $s_{\mathbf{\Gamma}_k^{(r)}}$ denote the sizes in bits of the global model, the local trained model at MU-n, and the MEN-aggregated model at MEN-k, respectively, in which $s_{\bar{\mathbf{\Gamma}}^{(r)}} = s_{\mathbf{\Gamma}_n^{(r)}} = s_{\mathbf{\Gamma}_k^{(r)}}$ [15], [45]. The number of successful transmissions for the uplink and downlink communication between the CS, MEN-k, and MU-n are, respectively, defined as $\nu_{k,K}^{up}, \nu_{n,K}^{up}, \nu_{n,k}^{up}$, and $\nu_{k,K}^{down}, \nu_{n,K}^{down}, \nu_{n,k}^{down}$. As a result, the time to download the global model $\bar{\mathbf{\Gamma}}^{(r)}$ from the CS to MEN-k, from the CS to MU-n and from MEN-k to MU-n at the r^{th} learning round can be respectively calculated as follows:

$$\begin{aligned} &T_{k,K}^{com-d} = \nu_{k,K}^{down} \frac{s_{\bar{\mathbf{\Gamma}}^{(r)}}}{B_{k,K}}, T_{n,K}^{com-d} = \nu_{n,K}^{down} \frac{s_{\bar{\mathbf{\Gamma}}^{(r)}}}{B_{n,K}}, \text{ and} \\ &T_{n,k}^{com-d} = \nu_{n,k}^{down} \frac{s_{\bar{\mathbf{\Gamma}}^{(r)}}}{B_{n,k}}, k < K. \end{aligned} \tag{9}$$

Next, the time required for successfully uploading a new trained model $\mathbf{\Gamma}_n^{(r)}$ from MU-n to MEN-k/the CS, and for uploading an MEN-aggregated model $\mathbf{\Gamma}_k^{(r)}$ from MEN-k to the CS is given as follows

$$\begin{aligned} &T_{n,k}^{com-u} = \nu_{n,k}^{up} \frac{s_{\mathbf{\Gamma}_n^{(r)}}}{B_{n,k}}, T_{n,K}^{com-u} = \nu_{n,K}^{up} \frac{s_{\mathbf{\Gamma}_n^{(r)}}}{B_{n,K}}, \text{ and} \\ &T_{k,K}^{com-u} = \nu_{k,K}^{up} \frac{s_{\mathbf{\Gamma}_k^{(r)}}}{B_{k,K}}, k < K. \end{aligned} \tag{10}$$

Here, we assume that the time required to aggregate the MUs' trained models at an MEN-k is significantly less than the model training time, and thus it can be negligible [15].

From (7) to (10), if an MU-n connects indirectly with the CS through an intermediate MEN-k, then the total time span from the distribution of the global model to the successful uploading of the local model from MU-n to MEN-k can be calculated as

$$T_n^{k,\dagger} = T_{k,K}^{com-d} + T_{n,k}^{com-d} + T_n^{cmp} + T_{n,k}^{com-u}. \tag{11}$$

Nevertheless, every MEN-k ($k < K$) has a fixed deadline T_k^{max} for starting to upload its aggregated model to the CS in order to avoid the straggling problem [46]. In this case, we suppose that the MSP assigns the same deadline for each MEN, i.e., $T_1^{max} = \ldots = T_k^{max} = \ldots = T_{K-1}^{max} = T^{max}$. During each learning round, an MEN-k must finish collecting the local models from its associated MUs and send the aggregated model to the CS before the deadline T_k^{max}. In this case, we can state the aforementioned requirement as $0 \leq T_n^{k,\dagger} \leq T^{max}, \forall 1 \leq n \leq N, \forall 1 \leq k < K$. Consequently, the overall time span for one learning round is $T^{max} + T_K^{com-u}$. Here, we assume that the MSP uses the same setting of $\nu_{m,M}^{up}$ and $B_{k,K}$ for all its MENs, thus the time delay for uploading an MEN-aggregated model to the CS from all MENs are the same, which is depicted by $T_{k,K}^{com-u} = T_K^{com-u}$, where $1 \leq k < K$. Lastly, the total time span at MEN-k from the global model distribution to the completion of MEN-aggregated model uploading to the CS (from MEN-k) can be derived as

$$T_n^k = T_n^{k,\dagger} + T_{k,K}^{com-u}, \forall n \in \mathcal{N}, \tag{12}$$

where $0 \leq T_n^k \leq T^{max} + T_K^{com-u}, \forall n \in \mathcal{N}, \forall 1 \leq k < K$.

If an MU-n is connected directly to the CS, then the total time span from the global model distribution to the completion of local model uploading to the CS (from MU-n) can be given as follows

$$T_n^K = T_{n,K}^{com-d} + T_n^{cmp} + T_{n,K}^{com-u}. \tag{13}$$

Here, the value of T_n^K is upper-bounded by the threshold $T^{max} + T_M^{com-u}$ in a learning round, i.e., $0 \leq T_n^K \leq T^{max} + T_K^{com-u}, \forall n \in \mathcal{N}$. Moreover, we can obtain the time required for the additional training over encrypted data at MEN-k, $1 \leq k < K$ and the CS respectively as follows:

$$T_k^* = \begin{cases} T_{k,K}^{com-d} + T_k^{cmp} + T_{k,K}^{com-u}, \\ \text{if } k < K, \\ T_K^{cmp}, \text{ otherwise}, \end{cases} \tag{14}$$

where $T_k^* \leq T^{max} + T_K^{com-u}$. It is worth noting that the size

of the raw (unencrypted) and encrypted datasets that will be used for training at MUs and the MENs, respectively, may vary depending on how the deadline T_{max} is selected. Intuitively, if the value of T_{max} is small, then large portions of the local datasets from the MUs will be cached at the MENs and CS for remote training as they have superior processing capabilities to compensate for many straggling MUs in the system.

IV. Implementation of MEC-based FL with encrypted training for straggling mitigation

A. Efficient NN Training with Encrypted Data

As mentioned in Section III, the generation of encrypted datasets at the MUs involves encrypting individual raw data samples into a single ciphertext. However, it is worth noting that traditional HE schemes, such as CKKS, can facilitate parallel computation in an SIMD fashion by packing multiple plaintext instances into a ciphertext value. This approach significantly reduces the expansion rate, leading to better amortized space and time complexity [47]. Specifically, CKKS can encode and encrypt a plaintext vector comprising S slots into a ciphertext, thus enabling element-wise arithmetic operations to be performed on the plaintext slots simultaneously. To handle calculations across inputs located in different slots, CKKS employs the rotation operation, denoted as $Rot(\tilde{\pi}, \ell)$. This operation transforms a ciphertext representation $\tilde{\pi}$ of a plaintext vector $\pi = (v_1, \ldots, v_S) \in \mathbb{R}^S$ into a ciphertext of $\sigma(\pi, \ell) := (v_\ell, \ldots, v_S, v_1, \ldots, v_{\ell-1})$. In this context, the value of ℓ can be either positive or negative, and a rotation by $(-\ell)$ is equivalent to a rotation by $(S - \ell)$.

By exploiting the SIMD property of the CKKS scheme, we develop an effective ciphertext packing method to parallelize the encrypted NN training at the MEN side. Since the number of slots S in a ciphertext is typically greater than the number of features F of a training sample, we can speed up the training process by packing multiple training samples into a single ciphertext. To achieve this, we divide a plaintext vector into several blocks. Each block contains $F + Q$ slots, where Q denotes the number of neurons in the first layer of the NN. Within each block, the first F slots are used to store the values of a training sample, while the last Q slots are padded with zeros. The use of the zero-slot in the encoding simplifies the computation of the encrypted vector-matrix product in the gradient calculations. By putting each training sample into one block separately, we can encrypt a batch of $\left\lfloor \frac{S}{F+Q} \right\rfloor$ training samples in a single ciphertext, as illustrated in Fig. 3. With this packing technique, an MEN can simultaneously calculate the gradients of $\left\lfloor \frac{S}{F+Q} \right\rfloor$ training samples by performing only one execution of the NN's forward and backward propagation over an input ciphertext.

Moreover, we adopt the diagonal encoding scheme in [48] to encode the weight matrix of a specific NN's layer. We define the j-th "extended diagonal" ($1 \leq j \leq v$) of a weight matrix $\mathbf{W}^{u \times v}$ as $\Lambda_j(\mathbf{W}) = \left(W_{i,(i+j) \mod v}\right)_{1 \leq i \leq u}$, where u and v are the layer input and output dimensions, respectively. Then, we pack each diagonal vector into an S-slots plaintext by replicating its value per each block of size $F+Q$. The encryption of $\mathbf{W}$ results in v separate ciphertexts, each corresponding to a diagonal of $\mathbf{W}$. An illustration of the packing and diagonal encoding techniques is shown in Fig. 4. By using this weight encryption scheme, the product of an encrypted input batch $\mathbf{f}$ from the previous layer with the encrypted weight matrix $\mathbf{W}$ of the current layer can be calculated by applying a sequence of multiplication and rotation operations as follows:

$$\mathbf{z} = \sum_{i=1}^{\lceil u/v \rceil} Rot\left(\sum_{j=1}^{v} Mul\big(\Lambda_j(\mathbf{W}), Rot(\mathbf{f}, j-1)\big), v(i-1) \right), \tag{15}$$

where $Rot(.)$ and $Mul(.)$ represent the ciphertext rotation and multiplication operators, respectively, as previously defined.

We illustrate the complete process of forward propagation at an NN layer based on the implemented ciphertext packing and matrix product computation in Appendix A (Fig. 11).

B. FL Implementaion

By solving the encrypted data caching problem described in the next section, we can obtain the vector $\mathbf{x}$ that determines the optimal fraction of data to be encrypted and cached at the MENs/CS. Afterward, the entire privacy-preserving FL process can be executed at the MUs, MENs and CS. Let $\tilde{\mathbf{\Omega}}_k = (\tilde{\mathbf{F}}_k, \tilde{\mathbf{y}}_k)$ denote the whole encrypted dataset collected from all MUs in $\mathcal{N}$ at the MEN-k which has size of $\sum_{n \in \mathcal{N}} x_{n,k} b_n$, where $\tilde{\mathbf{F}}_k$ and $\tilde{\mathbf{y}}_k$ are the encrypted data feature matrix and data label vector at MEN-k, respectively. Additionally, we define $\mathbf{\Omega}_{n,0} = (\mathbf{F}_{n,0}, \mathbf{y}_{n,0})$ as the dataset to be trained locally at MU-n, which has size of $x_{n,0} b_n$. Here, $\mathbf{F}_{n,0}$ is the training feature matrix, and $\mathbf{y}_{n,0}$ is the label vector of local dataset at MU-n. Furthermore, the goal of the learning process is to minimize a pre-determined loss function $\mathcal{L}$ by gradually updating the global models' parameters $\bar{\mathbf{\Gamma}}$ (i.e., the set of models' weights and biases).

Here, we consider the implementation of a DNN model for a general classification task. Note that the proposed framework is also applicable to solve other deep learning tasks (e.g., regression) or extended for other neural network models (e.g., CNN). Recent works such as [49] have developed techniques to implement CNN layers using HE. As such, we can integrate these prior arts into our framework to achieve encrypted CNN feature extraction, by using their homomorphic convolution layers as the feature extractor and our proposed fully-connected layer as the classifier. The gradient descent algorithm is used to train the NN. Let $\mathcal{L} = \{0, 1, \ldots, l, \ldots, L\}$ be the set of NNs' layers, where layer 0 and layer L are the input and output layer, respectively. As such, the output at layer $l+1$ of the local model at MU-n, denoted by $\mathbf{a}_n^{(l+1)}$, can be calculated as

$$\mathbf{a}_n^{(l+1)} = \alpha^l\Big(\mathbf{a}_n^l \mathbf{W}_n^l + \mathbf{b}_n^l\Big), \tag{16}$$

in which $\mathbf{W}_n^l$ and $\mathbf{b}_n^l$ are respectively the weight matrix and bias vector at layer l, and $\alpha^l(.)$ represents the activation function such as *tanh* function $f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ or *sigmoid* function $f(z) = \frac{1}{1+e^{-z}}$ [50]. To overcome the over-fitting problem and reduce the generalization error, a dropout layer l_{drop} ($l_{drop} < L$) can be implemented after the last hidden

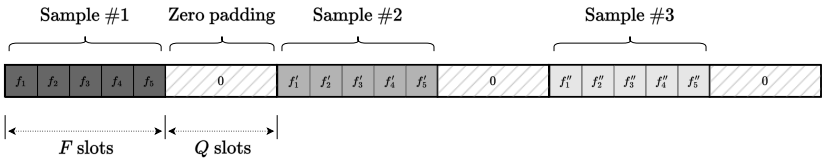


Fig. 3: An illustration of training samples packing. A plaintext vector of S slots is divided into blocks, each containing $F+Q$ slots, where F is the input size and Q is the number of units in the first NN's layer. Within each block, the first F slots store the values of an input vector (i.e., a training sample), while the remaining Q slots are filled with zeroes as padding.

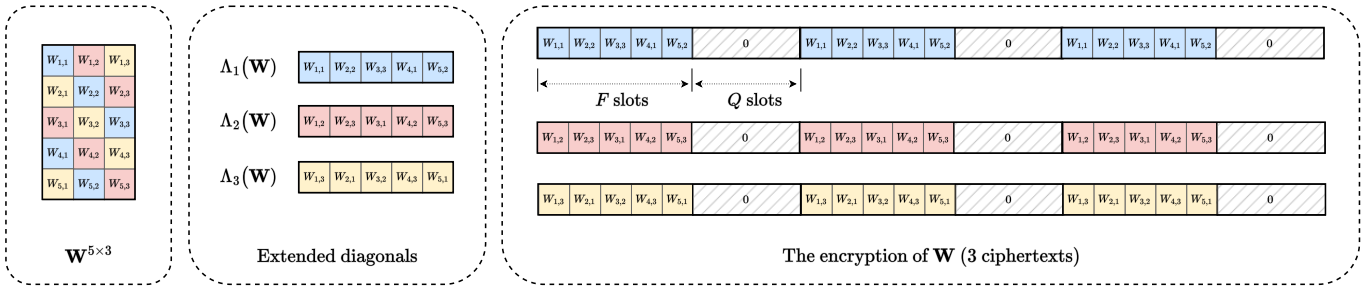


Fig. 4: An illustration of packing of a 5×3 weight matrix $\mathbf{W}$. Each extended diagonal of $\mathbf{W}$ is encoded into a plaintext of S slots by replicating its values within each block of $F+Q$ slots. These plaintexts are then encrypted by HE, resulting in a total of three ciphertexts.

layer, which will randomly sets the elements of $\mathbf{a}_n^{(l+1)}$ to 0 with a certain frequency of rate.

For an encrypted model at MEN-k, the output vector $\tilde{\mathbf{a}}_k^{(l+1)}$ at layer $l+1$ can be computed as

$$\tilde{\mathbf{a}}_k^{(l+1)} = \tilde{\alpha}^l\Big(\tilde{\mathbf{a}}_k^l \otimes \tilde{\mathbf{W}}_k^l \oplus \tilde{\mathbf{b}}_k^l\Big), \tag{17}$$

where $\tilde{\mathbf{W}}_k^l$ and $\tilde{\mathbf{b}}_k^l$ are the encrypted weight and bias at layer l, respectively, while $\oplus$ and $\otimes$ are, respectively, the encrypted version of arithmetic addition and multiplication operators (i.e., the $Add(.)$ and $Mul(.)$ operators defined in (1) and (3)). Specifically, we have $\tilde{\pi}_1 \oplus \tilde{\pi}_2 = Add(\tilde{\pi}_1, \tilde{\pi}_2)$ and $\tilde{\pi}_1 \otimes \tilde{\pi}_2 = Mul(\tilde{\pi}_1, \tilde{\pi}_2)$ with $\tilde{\pi}_1$ and $\tilde{\pi}_2$ are two ciphertexts. The output of $\Big(\tilde{\mathbf{a}}_k^l \otimes \tilde{\mathbf{W}}_k^l \oplus \tilde{\mathbf{b}}_k^l\Big)$ is then also a vector in encrypted form. In addition, $\tilde{\alpha}^l$ represents the polynomial approximation of the activation function α^l using the *Taylor* series [43]. For example, the *sigmoid* function $f(z) = \frac{1}{1+e^{-z}}$ can be polynomially approximated by $f(z) = 0.5 + 0.25z + 0.02z^3$. As the approximated function comprises only homomorphic operations (i.e. addition and multiplication), it can be calculated over an encrypted input value, as shown in (17).

Upon reaching the last layer L, the final output vector $\mathbf{a}_n^L$ at MU-n and $\tilde{\mathbf{a}}_k^{(l+1)}$ at MEN-k can be respectively expressed by

$$\mathbf{a}_n^L = \alpha^{(L-1)}\Big(\mathbf{a}_n^{(L-1)}\mathbf{W}_n^{(L-1)} + \mathbf{b}_n^{(L-1)}\Big), \tag{18}$$

and

$$\tilde{\mathbf{a}}_k^L = \alpha^{(L-1)}\Big(\tilde{\mathbf{a}}_k^{(L-1)} \otimes \tilde{\mathbf{W}}_k^{(L-1)} \oplus \tilde{\mathbf{b}}_k^l\Big), \tag{19}$$

where $\alpha^{(L-1)}$ denotes the *softmax* activation function employed to generate a probability distribution of all possible classes [50]. It should be noted that the *softmax* function involves the calculation of exponential and inverse functions that are non-homomorphic. Therefore, direct calculation of the *softmax* function using encrypted values as inputs is not feasible. For that, we adopt the approximation technique in [41] which is based on the Goldschmidt division method and Gumbel softmax function to calculate the output vector at the last layer of an MEN's encrypted model.

Given $\mathbf{y}_{n,0}$, $\tilde{\mathbf{y}}_k$, $\mathbf{a}_n^L$, and $\tilde{\mathbf{a}}_k^L$, the loss functions of each MU-$n, n \in \mathcal{N}$ and MEN-$k, k \in \mathcal{K}$, at r-th learning round can be acquired through utilizing the squared Frobenius norm, that is,

$$\begin{aligned}\mathcal{L}_n\Big(\boldsymbol{\Upsilon}^{(r)}\Big) &= \frac{1}{2x_{n,0}b_n}\Big\|\mathbf{a}_n^L - \mathbf{y}_{n,0}\Big\|_F^2 \ , \\ &= \frac{1}{2x_{n,0}b_n}\sum_{j=1}^{x_{n,0}b_n}(a_{n,j}^L - y_{n,j})^2,\end{aligned} \tag{20}$$

and

$$\begin{aligned}\mathcal{L}_k\Big(\tilde{\boldsymbol{\Upsilon}}^{(r)}\Big) &= \frac{1}{2\sum_{n\in\mathcal{N}} x_{n,k}b_n}\Big\|\tilde{\mathbf{a}}_k^L - \tilde{\mathbf{y}}_k\Big\|_F^2 \ , \\ &= \frac{1}{2\sum_{n\in\mathcal{N}} x_{n,k}b_n}\sum_{j=1}^{\sum_{n\in\mathcal{N}} x_{n,k}b_n}(\tilde{a}_{k,j}^L - \tilde{y}_{k,j})^2,\end{aligned} \tag{21}$$

in which $\boldsymbol{\Upsilon}_n^{(r)}$ and $\tilde{\boldsymbol{\Upsilon}}_k^{(r)}$ are the plain global model at MU-n and the encrypted global model at MEN-k (such that $\boldsymbol{\Upsilon}_n^{(r)} = Dec(g_{sk}, \tilde{\boldsymbol{\Upsilon}}_k^{(r)})$), respectively, $y_{n,j}$ and $\tilde{y}_{k,j}$ are the sample points of the plain and encrypted ground-truth label vector $\mathbf{y}_{n,0}$ and $\tilde{\mathbf{y}}_k$, respectively, while $a_{n,j}^L$ and $\tilde{a}_{k,j}^L$ are the elements of predicted label vector $\mathbf{a}_n^L$ and $\tilde{\mathbf{a}}_k^L$, respectively.

From (20) and (21), the local gradient at MU-n and the encrypted gradient at MEN-k can be respectively calculated as follows

$$\nabla\boldsymbol{\Upsilon}_n^{(r)} = \frac{\partial\mathcal{L}_n\Big(\boldsymbol{\Upsilon}^{(r)}\Big)}{\partial\boldsymbol{\Upsilon}^{(r)}}, \text{ and } \nabla\tilde{\boldsymbol{\Upsilon}}_k^{(r)} = \frac{\partial\mathcal{L}_k\Big(\tilde{\boldsymbol{\Upsilon}}^{(r)}\Big)}{\partial\tilde{\boldsymbol{\Upsilon}}^{(r)}}. \tag{22}$$

Suppose that the local gradient calculated at MU-n can be

simplified as

$$\nabla \boldsymbol{\Upsilon}_n^{(r)} = \frac{\partial \mathcal{L}_n\Big(\boldsymbol{\Upsilon}^{(r)}\Big)}{\partial \boldsymbol{\Upsilon}_n^{(r)}} = \frac{\partial \left[\frac{1}{2x_{n,0}b_n}\left\|\mathbf{a}_n^L - \mathbf{y}_{n,0}\right\|_F^2\right]}{\partial \boldsymbol{\Upsilon}_n^{(r)}} = \frac{1}{2x_{n,0}b_n}\mathbf{F}_n^T\Big(\mathbf{F}_{n,0}\boldsymbol{\Upsilon}^{(r)} - \mathbf{y}_{n,0}\Big), \quad (23)$$

where $\mathbf{F}_{n,0}$, $\mathbf{y}_{n,0}$, and $\mathbf{y}_{n,0}$ are respectively the matrix of training features, vector of true labels, and vector of predicted labels at MU-n. Accordingly, we can derive the encrypted gradient at the MEN-k in (22) from its encrypted dataset as follows

$$\nabla \tilde{\boldsymbol{\Upsilon}}_k^{(r)} = \frac{1}{\sum_{n\in\mathcal{N}} x_{n,k}b_n}(\mathbf{F}_i^o)^T\Big(\mathbf{F}_i^o\tilde{\boldsymbol{\Upsilon}}^{(r)} - \tilde{\mathbf{y}}_k\Big). \quad (24)$$

Using (22), each MU-n can encrypt $\nabla \boldsymbol{\Upsilon}_n^{(r)}$ into $\nabla \tilde{\boldsymbol{\Upsilon}}_n^{(r)} = Enc\big(g_{pk}, \nabla \boldsymbol{\Upsilon}_n^{(r)}\big)$ and upload $\nabla \tilde{\boldsymbol{\Upsilon}}_n^{(r)}$ to a selected MEN for model aggregation. Let $\mathcal{N}_k$ denote the set of MUs who forwards its local gradient update to the MEN-k. Thus, the total received gradient at the MEN-k is expressed by

$$\nabla \tilde{\boldsymbol{\Upsilon}}_{\mathcal{N}_k}^{(r)} = \sum_{n\in\mathcal{N}_k} x_{n,0}b_n \nabla \tilde{\boldsymbol{\Upsilon}}_n^{(r)}. \quad (25)$$

Since the MEN-k also produces its own gradient $\nabla \tilde{\boldsymbol{\Upsilon}}_k^{(r)}$ from the encrypted training process, MEN-k then uploads the total gradient $\nabla \tilde{\boldsymbol{\Upsilon}}_{\mathcal{N}_k}^{(r)} + \nabla \tilde{\boldsymbol{\Upsilon}}_k^{(r)}$ to the CS. Therefore, the global encrypted gradient at learning round r can be derived as

$$\nabla \tilde{\boldsymbol{\Upsilon}}^{(r)} = \sum_{k\in\mathcal{K}}\left(\frac{\nabla \tilde{\boldsymbol{\Upsilon}}_{\mathcal{N}_k}^{(r)} + \nabla \tilde{\boldsymbol{\Upsilon}}_k^{(r)}}{\sum_{n\in\mathcal{N}_k} x_{n,0}b_n + \sum_{n\in\mathcal{N}} x_{n,k}b_n}\right). \quad (26)$$

Consequently, the CS can update the encrypted global model $\tilde{\boldsymbol{\Upsilon}}^{(r+1)}$ for the subsequent learning round by using the gradient descent algorithm. Specifically,

$$\tilde{\boldsymbol{\Upsilon}}^{(r+1)} = \tilde{\boldsymbol{\Upsilon}}^{(r)} - \lambda \nabla \tilde{\boldsymbol{\Upsilon}}^{(r)}, \quad (27)$$

where λ is the learning rate. Additionally, the global loss function at the $(r+1)$-th learning round can be evaluated as follows:

$$\mathcal{L}\Big(\tilde{\boldsymbol{\Upsilon}}^{(r+1)}\Big) = \frac{1}{N+K}\left(\sum_{n\in\mathcal{N}}\mathcal{L}_n\Big(\boldsymbol{\Upsilon}^{(r)}\Big) + \sum_{k\in\mathcal{K}}\mathcal{L}_k\Big(\tilde{\boldsymbol{\Upsilon}}^{(r)}\Big)\right). \quad (28)$$

The learning process continues until either the global loss reaches convergence or the number of learning rounds exceeds the predetermined threshold r_{th}. Upon completion, the final global loss $\mathcal{L}^*(\tilde{\boldsymbol{\Upsilon}}^*)$ and the final encrypted global model $\tilde{\boldsymbol{\Upsilon}}^*$ are generated. The summary of the entire proposed MEC-based privacy-preserving FL process is presented in Algorithm 1.

Algorithm 1 Proposed FL Framework with MEC-assisted Encrypted Training

Set r_{th}, $\tilde{\boldsymbol{\Upsilon}}^{(0)}$, and $r = 0$
Solve problem $(\mathbf{P}_\mathbf{x})$ to obtain the vector $\mathbf{x}$ of optimal fraction of data to be encrypted
for $\forall n \in \mathcal{N}$ **do**
  Split the entire dataset $\boldsymbol{\Omega}_n$ into $K+1$ subdataset $\boldsymbol{\Omega}_{n,0}, \boldsymbol{\Omega}_{n,1}, \ldots, \boldsymbol{\Omega}_{n,K}$ where $|\boldsymbol{\Omega}_{n,k}| = x_{n,k}b_n$
  Create a secret key and a public key: $g_n^{sk} = SKGen(n)$; $g_n^{pk} = PKGen(g_n^{sk})$
  for $\forall k \in \mathcal{K}$ **do**
    Generate the encrypted subdataset $\tilde{\boldsymbol{\Omega}}_{n,k}$ from $\boldsymbol{\Omega}_{n,k}$
    Send the encrypted subdataset $\tilde{\boldsymbol{\Omega}}_{n,k}$ to the MEN-k
  end for
  Set $\mathbf{F}_{n,0}$ and $\mathbf{y}_{n,0}$ from $\boldsymbol{\Omega}_{n,0}$
end for
for $\forall k \in \mathcal{K}$ **do**
  Combine the received encrypted datasets into $\tilde{\boldsymbol{\Omega}}_k$
  Set $\tilde{\mathbf{F}}_k$ and $\tilde{\mathbf{y}}_k$ from $\tilde{\boldsymbol{\Omega}}_k$
end for
while $r \le r_{th}$ **and** $\mathcal{L}\Big(\tilde{\boldsymbol{\Upsilon}}^{(r)}\Big)$ is not converged **do**
  for $\forall n \in \mathcal{N}$ **do**
    Compute $\mathbf{a}_n^L$ using $\mathbf{F}_{n,0}$ and $\boldsymbol{\Upsilon}^{(r)}$
    Decrypt $\nabla \tilde{\boldsymbol{\Upsilon}}^{(r)}$ using the secret key: $\nabla \boldsymbol{\Upsilon}^{(r)} = Dec\Big(g_n^{sk}, \nabla \tilde{\boldsymbol{\Upsilon}}^{(r)}\Big)$
    Calculate $\mathcal{L}_n\Big(\boldsymbol{\Upsilon}^{(r)}\Big)$ and $\nabla \boldsymbol{\Upsilon}_n^{(r)}$
    Send the encrypted value $\nabla \tilde{\boldsymbol{\Upsilon}}_n^{(r)} = Enc\Big(g_n^{pk}, \nabla \boldsymbol{\Upsilon}_n^{(r)}\Big)$ and to a selected MEN
  end for
  for $\forall k \in \mathcal{K}$ **do**
    Compute $\tilde{\mathbf{a}}_k^L$ using $\tilde{\mathbf{F}}_n$ and $\tilde{\boldsymbol{\Upsilon}}^{(r)}$
    Find $\mathcal{L}_k\Big(\tilde{\boldsymbol{\Upsilon}}^{(r)}\Big)$ and $\nabla \tilde{\boldsymbol{\Upsilon}}_k^{(r)}$
    Aggregate $\nabla \tilde{\boldsymbol{\Upsilon}}_{\mathcal{N}_k}^{(r)}, \forall n \in \mathcal{N}_k$ using (25)
    Send $\mathcal{L}_k\Big(\tilde{\boldsymbol{\Upsilon}}^{(r)}\Big)$ and $\nabla \tilde{\boldsymbol{\Upsilon}}_{\mathcal{N}_k}^{(r)} + \nabla \tilde{\boldsymbol{\Upsilon}}_k^{(r)}$ to the CS
  end for
  The CS obtains $\mathcal{L}_k\Big(\tilde{\boldsymbol{\Upsilon}}^{(r)}\Big)$ and $\nabla \tilde{\boldsymbol{\Upsilon}}_{\mathcal{N}_k}^{(r)} + \nabla \tilde{\boldsymbol{\Upsilon}}_k^{(r)}$ from all MENs
  Calculate the encrypted global gradient $\nabla \tilde{\boldsymbol{\Upsilon}}^{(r)}$ using (26)
  Update the encrypted global model $\tilde{\boldsymbol{\Upsilon}}^{(r+1)}$ using (27)
  Evaluate the global loss $\mathcal{L}\Big(\tilde{\boldsymbol{\Upsilon}}^{(r+1)}\Big)$ using (28)
  $r = r + 1$
end while
Return the final global loss $\mathcal{L}^*(\tilde{\boldsymbol{\Upsilon}}^*)$ and the final encrypted global model $\tilde{\boldsymbol{\Upsilon}}^*$

V. Encrypted Data Caching Optimization Problem in MEC-based FL

A. Problem Formulation

Our proposed FL framework aims to maximize the profit of the MSP while simultaneously minimizing the straggling problem under the limited incentive budget for encrypted data caching and training at the MENs and CS. In particular, we define the profit of the MSP, which is composed of the gain and cost functions, respectively, for the local model training execution at MU-n and the encryption-based model training execution at MEN-k as follows:

$$P_k = \underbrace{\lambda_k\sqrt{\sum_{n=1}^{N} x_{n,k}b_n}}_{\textit{Gain function}} - \underbrace{(\zeta_k\mu_k f_k^2\psi_k + \beta_k)\sum_{n=1}^{N} x_{n,k}b_n}_{\textit{Cost function}}, \quad (29)$$

and

$$P_n = \underbrace{\lambda_n\sqrt{\left(1 - \sum_{k=1}^{K} x_{n,k}\right)b_n}}_{\textit{Gain function}} - \underbrace{\rho_n\left(1 - \sum_{k=1}^{K} x_{n,k}\right)b_n}_{\textit{Cost function}}, \quad (30)$$

where the conversion parameters λ_k and λ_n indicate the monetary value of utilizing the encrypted dataset cached at MEN-k and the raw local dataset at MU-n, respectively, according to the current data market prices [51]. Here, ζ_k is the processor's effective capacitance constant for MEN-k, ψ_k is the unit cost of energy usage to train a data sample in the encrypted format, and β_k represents the incentive unit for each encrypted data sample of MU-n in the remote training process at the MEN or CS. Additionally, the constant ρ_n is the incentive unit for MU-n to be involved in the local training process. As the square root function is utilized inside the gain functions of both P_k and P_n, those gain values grow when more data is used in the FL training. However, the MSP may not be motivated to increase the size of the training dataset if a substantial increase results in a reduced overall gain in the model's accuracy [52]. In light of this, we consider the optimization problem to maximize the MSP's profit as follows:

$$(\mathbf{P_x}) \quad \max_{\mathbf{x}} \sum_{n=1}^{N} P_n + \sum_{k=1}^{K} P_k, \tag{31a}$$

$$\text{s.t. } 1 - \sum_{k=1}^{K} x_{n,k} \geq 0, \forall n \in \mathcal{N}, \tag{31b}$$

$$\left(1 - \sum_{k=1}^{K} x_{n,k}\right) b_n \leq c_n, \forall n \in \mathcal{N}, \tag{31c}$$

$$\sum_{n=1}^{N} x_{n,k} b_n \leq c_k, \forall 1 \leq k < K, \tag{31d}$$

$$0 \leq T_n^m \leq T^{max} + T_K^{com-u}, \forall n \in \mathcal{N}, \forall k \in \mathcal{K}, \tag{31e}$$

$$0 \leq T_k^* \leq T^{max} + T_K^{com-u}, \forall k \in \mathcal{K}, \tag{31f}$$

$$\sum_{n=1}^{N} \rho_n \left(1 - \sum_{k=1}^{K} x_{n,k}\right) b_n + \sum_{k=1}^{K} \left(\beta_k \sum_{n=1}^{N} x_{n,k} b_n\right) \leq I, \tag{31g}$$

$$0 \leq x_{n,k} \leq 1, \forall n \in \mathcal{N}, \forall k \in \mathcal{K}, \tag{31h}$$

where constraints (31b) indicate the total portion of encrypted data at each MU-n must not be exceed 1. Constraints (31c) and (31d) guarantee that the size of the raw dataset for training at MU-n must not exceed the maximum threshold c_n, and the size of the encrypted dataset aggregated at an MEN-k must not exceed its training capacity, for $k < K$, owing to its limited computational resources. Subsequently, constraints (31e) and (31f) imply that in order to prevent the straggling effect, the training duration for each communication round must be less than or equal to the deadline time $T^{max} + T_K^{com-u}$. Finally, constraint (31g) indicates that the MSP's incentive budget, which is denoted by I, must be larger than the total incentives offered to all participating MUs.

B. Optimal Solution

To find the optimal solution in the proposed optimization problem $(\mathbf{P_x})$, we prove that the problem is convex. This can be achieved by showing that the objective function outlined in (31a) is concave, since all the constraints (31b) - (31h) are linear.

THEOREM 1. *The function represented by (31a), i.e., $\left[\sum_{n=1}^{N} P_n + \sum_{k=1}^{K} P_k\right]$, is concave w.r.t. all $x_{n,k}, \forall n \in \mathcal{N}, \forall k \in \mathcal{K}$, given that constraints (31b)-(31h) are satisfied.*

Proof. See Appendix B □

Since $(\mathbf{P_x})$ is a convex optimization problem, it can be solved using the well-known tools introduced in [53]. In this work, we utilize the interior point method, which has been proven effective in solving large-scale, sparse non-linear optimization problems [54].

VI. Performance Evaluation

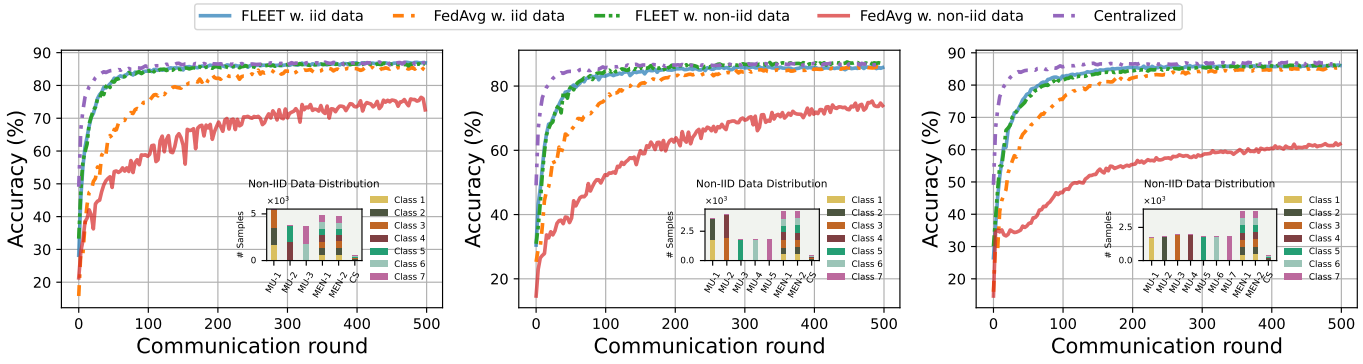
In this section, we evaluate the performance of the proposed FL framework in mitigating the straggling problem and maximizing the profit of the MSP. Specifically, we simulate an MEC network in MATLAB and solve the optimization problem outlined in Section V to find the optimal amount of data to be encrypted and cached at MENs/CS. Using the derived optimal solution, we create and distribute the encrypted datasets to the corresponding MENs and CS for training a practical ML model. The goal is to demonstrate the model accuracy and convergence rate of the proposed FL framework. Next, the MSP's profit will be evaluated and compared with other baseline solutions. To this end, we begin this section by presenting the parameters used in the simulations and then provide a detailed description of the ML dataset and model architecture in the next subsection.

A. Dataset and NN Architecture Setup

In our experiments, we evaluate the performance of the proposed FL framework using a dataset from real-world human activity recognition (HAR) collected in 2019 [25]. The dataset includes 15 million raw samples of gyroscope and accelerometer sensors' data extracted from the smartphones and smartwatches of multiple users. From the total of 18 activity labels, we select seven hand-oriented activities, including dribbling a basketball, playing catch, typing, writing, clapping, brushing teeth, and folding clothes, numbered from 1 to 7, respectively. The extracted dataset is then randomly divided into training set (80%) and testing set (20%). Additionally, all the samples are normalized through subtracting the average and dividing by the standard deviation of the training samples. We consider two data distribution scenarios for the FL training: i.i.d. and non-i.i.d.. For i.i.d. setting, each MU is randomly assigned a uniform distribution over all seven classes. In the non-i.i.d. setting, the data is sorted by class and divided to create an extreme case in which the data samples from two different MUs have no common labels. The NN architecture used in the experiments is shown in TABLE II, consisting of a simple three-layer fully connected network with a sigmoid activation function in the first hidden layer.

B. System Parameters

To evaluate the MSP's profit, we use MATLAB to simulate an MEC network with 1 CS, 4 MENs, and 100 participating MUs. The dataset size of each participating MU-n is randomly



(a) 3 MUs (b) 5 MUs (c) 7 MUs

Fig. 5: The accuracy performance under i.i.d. and non-i.i.d. data settings as the number of participating MUs increases.

TABLE II: Neural network architecture.

Layer	Number of Neurons	Activation Function
Input	91	-
Hidden Layer 1	60	Sigmoid
Hidden Layer 2	30	None
Output	7	None

chosen from 100,000 to 1,000,000 samples. According to the HAR dataset, each training sample has 91 features, resulting in a total size of 2,912 bits (assuming that each feature is a 32-bit floating point number). The communication rates between CS-MEN, MEN-MU, and CS-MU are set at $r_k^K = 200\text{Mbps}, \forall k \in \mathcal{K}, k \neq K$, $r_n^k = 30\text{Mbps}, \forall k \in \mathcal{K}, k \neq K, \forall n \in \mathcal{N}$, and $r_n^K = 20\text{Mbps}, \forall n \in \mathcal{N}$, respectively, regarding the actual rates of 5G and Wi-Fi connections [55], [56]. The monetary benefits of using encrypted datasets at the MENs/CS and raw datasets at the MUs are respectively specified as $\lambda_k = 0.5, \forall k \in \mathcal{K}$ and $\lambda_n = 0.1, \forall n \in \mathcal{N}$, owing to the non-i.i.d. nature of local datasets at individual MUs. As a result, the training updates obtained from the combined datasets at the MENs/CS are more valuable for the MSP in terms of improving the global model accuracy. We utilize $\zeta_k = 0.5 \times 10^{-26}$ [57], and the CPU frequencies of MENs and MUs are respectively $f_k = 2\text{GHz}$, and $f_n = 1.18\text{GHz}$ with respect to specifications of prevalent devices [58], [59]. We also set $\beta_k = 0.0001, \forall k \in \mathcal{K}, k \neq K$, and $\beta_K = 0.0008$ to reflect that the cost of caching the encrypted data at the CS is higher than that at the MENs. The other parameters are $\mu_k = 0.01$, $\rho_n = 0.001$, and $T^{max} = 1$ second. The proposed FL framework, denoted as FLEET (***F**ederated **L**earning with **E**dge-assisted **E**ncrypted **T**raining*), is compared to two other scenarios: (i) FLEET-CS, where the data from an MU can only be encrypted and uploaded to the CS, and (ii) the traditional FL approach using *FedAvg* method [60], where the training is only performed locally using the raw datasets at the MUs.

C. Performance on FL Accuracy

1) Accuracy with different numbers of MUs: In this subsection, we examine the FL accuracy and convergence rate of our proposed framework in various scenarios. Here, we consider a simple MEC network consisting of 1 CS, and 2 MENs, and the number of participating MUs is increased

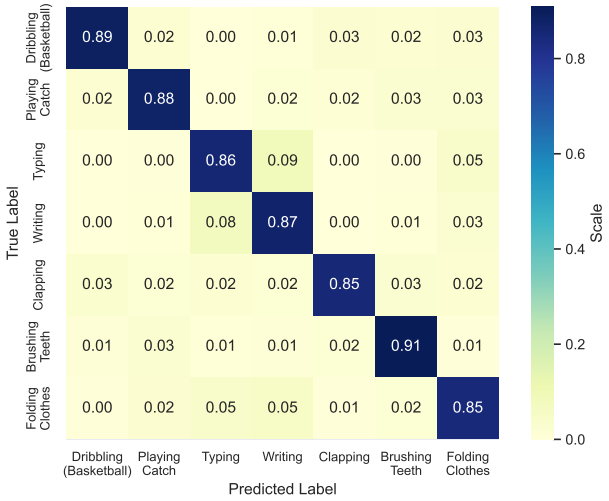
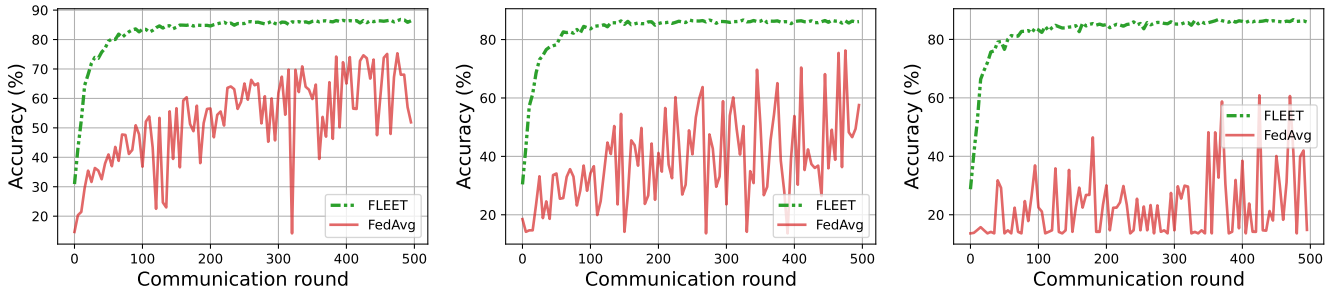


Fig. 6: Confusion matrix of FLEET with 7 participating MUs.

from 3 to 7. Fig. 5 demonstrates the training process of the conventional FL (*FedAvg* [60]) and the proposed FL (FLEET) with different numbers of MUs, where both i.i.d. and non-i.i.d. data distribution scenarios are taken into account. The figure also includes the training process of centralized deep learning for comparison. Here, it is worth mentioning that one communication round in the FL approach is equivalent to one epoch in the centralized method. According to Fig. 5, the FLEET framework can generally preserve an identical accuracy performance in both i.i.d. and non-i.i.d. data scenarios, regardless the number of participating MUs. Considering the i.i.d. scenario, the FLEET achieves higher accuracy with an improvement up to 1.24% over the *FedAvg*. Notably, in the case of 3 participating MUs, the accuracy of the FLEET is nearly equivalent to that of centralized learning, with a performance deviation of only 0.28%. Moreover, the FLEET demonstrates greater stability in terms of accuracy performance at each learning round and reaches an accuracy level of 86%, which is 1.56 to 2.86 times faster than the *FedAvg* in case of 5 and 7 participating MUs, respectively. The reason is that the *FedAvg* uses a lower number of training samples to update the global model in each learning round due to the limited size of the dataset that can be handled at



(a) 20% straggling probability (b) 50% straggling probability (c) 80% straggling probability

Fig. 7: The accuracy performance of FLEET and *FedAvg* under various straggling probabilities.

the MUs. Meanwhile, it can be observed from Fig. 5 that the convergence speed of the FLEET slightly reduces when the number of MUs increases from 3 to 7. This is because when more MUs participate in the FL process, the local datasets at the MUs become smaller, leading to an imbalance in the number of training samples at the MENs and CS, thereby degrading the training performance [11].

When the local datasets at participating MUs are in a non-i.i.d. setting, which is more reflective of practical conditions, the conventional FL approach experiences accuracy reduction. This is influenced by the biased model updates from non-i.i.d. data and the insufficient number of training samples, which hinders the improvement of accuracy levels. Typically, in the case of 7 participating MUs, the final accuracy in *FedAvg* drops to only 62.11% (as depicted in Fig. 5(c)), as the entire dataset at an MU now consists solely of data from a single class, leading to a substantial bias. In contrast, FLEET is capable of maintaining the same accuracy level as in the i.i.d. data scenario, and thus produces an accuracy gap with *FedAvg* of 10.48%, 11.99%, and 24.29% when using 3 MUs, 5 MUs, and 7 MUs, respectively. Later, the confusion matrix in Fig. 6 demonstrates the prediction performance of the proposed FL framework for each activity label with 7 participating MUs. From the above results, it can be inferred that incorporating the additional encrypted caching and training process at MENs generally accelerates the convergence rate, enhances the global model accuracy, and results in more consistent performance, particularly in a practical non-i.i.d. data distribution environment.

2) Accuracy under different straggling probabilities: To further demonstrate the superiority of the FLEET, we evaluate its performance under various straggling probabilities, i.e., the the probability that participating MUs face straggling problems such as low computation resources or poor communication links (which prevent them from sending local updates to the corresponding MENs/CS at a given learning round for model aggregation). Using a system consisting of 5 MUs for training, we consider three different straggling probabilities of 20%, 50%, and 80% so that when the straggling probability rises, fewer MUs are able to upload aggregated local models at each round. As shown in Fig. 7, the FLEET maintains an accuracy of approximately 86.8% for all straggling probability scenarios, while the *FedAvg* fails to retain its accuracy when the straggling probability gets higher. As a result, the final training accuracy of FLEET surpasses that of *FedAvg* by 9.70%, 10.66%, and 25.96%, respectively, for 20%, 50%, and 80% straggling probabilities. The results also highlight that our proposed framework can fully maintain its level of accuracy, as well as steady convergence speed in the presence of unreliable communication links and unstable processing capability of participating devices. This is owing to the additional secure training process at MENs which compensates for the straggling problems.

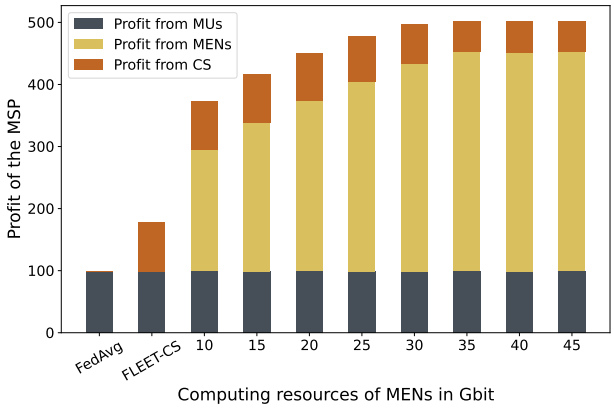


Fig. 8: The MSP's profit when MENs' computation resources increase.

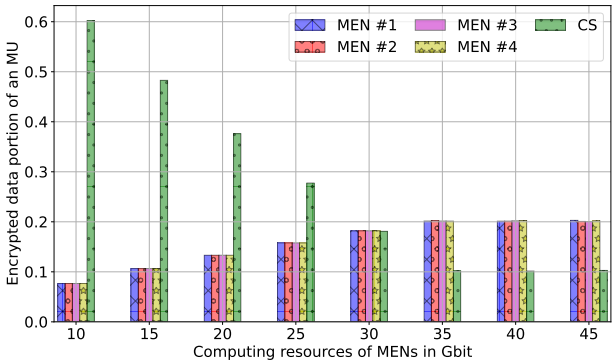


Fig. 9: The average portion of encrypted data at an MU when MENs' computation resources increase.

D. Performance on The MSP's Profit

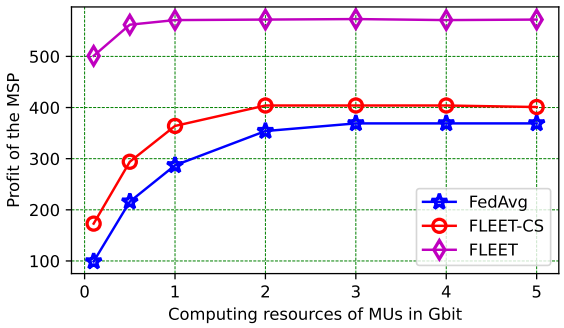
We first examine the MSP's profit obtained using the FLEET when the computation resources across all MENs increase from 0 to 50Gbit. To emphasize the straggling issue in the FL process, we will keep the computation resources at the MUs at a low setting of 0.1Gbit. As observed in Fig. 8, the FLEET can outperform FLEET-CS and *FedAvg* by, respectively, 2.84 and 5.07 times in terms of the MSP's total profit on account of the additional profit gains from training process at MENs. Furthermore, training the encrypted datasets at MENs located near the MUs helps to minimize the costs induced by caching and computing processes at the CS. This is aligned with the highest profit returns by training process at the MENs, as shown in Fig. 8. There exists a certain threshold, which is identified as 35Gbit, beyond which further enhancement of computation resources at the MENs does not improve the profit of the MSP. Starting from this threshold, the majority of data samples from a single MU are encrypted and uploaded to the MENs for additional training process in order to optimize profits, as depicted in Fig. 9. Nevertheless, an MU is still able to train around 10.19% of its entire dataset by utilizing local computation resources without encountering the straggling problem, so as to reduce the need for encrypted training with higher costs at the remote servers.

Next, we examine the improvement in the performance of FLEET as the MUs' computation resources vary between 0.1Gb and 10Gb while the MENs' computation resources remain constant. As illustrated in Fig. 10(a), the MSP's profit obtained by using *FedAvg* gradually increases as the MUs are complemented with more computation resources, so as to mitigate the straggling problem. This aligns with the results presented in TABLE III which indicate a higher percentage of local datasets that can be used to train local models at MUs without incurring the straggling problem. However, the MSP's profit from *FedAvg* reaches its maximum when the MU can train its entire local dataset locally, specifically when the computation resources of the MU exceed 3Gbits. In this experiment, the FLEET can still attain the maximum MSP's profit (regardless the MUs' computation resources) with an increase of at least 1.56 times and 1.42 times compared to *FedAvg* and FLEET-CS, respectively. In particular, despite the decrease in the total portion of encrypted data cached at the MENs/CS as the MUs' computation resources increase (as shown in TABLE III), the FLEET can still slightly enhance the MSP's profit through training a larger amount of data at the MENs and CS, thereby contributing the additional profit. Once the portions of encrypted data at MENs reach optimal levels, the MSP's profit of FLEET will no longer increase, and this trend can also be observed with a lower profit in the case of FLEET-CS.

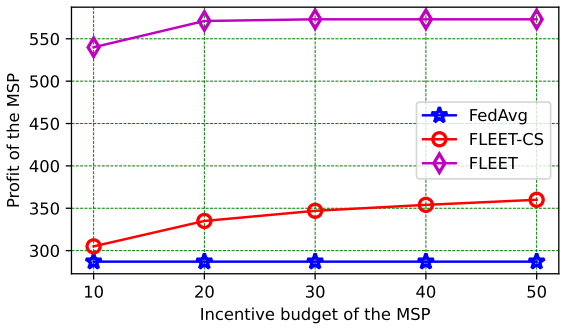
Fig. 10(b) demonstrates how the MSP's profit changes when the MSP's incentive budget varies between 10 and 50 monetary units. At a small budget of 10 monetary units, the FLEET has the lowest MSP's profit due to the insufficient budget for incentivizing MUs to cache data at MENs and the CS, resulting in the smallest portion of encrypted data at the MU (as observed in TABLE IV). As the incentive budget

TABLE III: The average portions of local and encrypted data at an MU when MUs' computation resources increase.

c_n (Gb)	*FedAvg*	FLEET-CS		FLEET	
	Local	Local	Encrypted	Local	Encrypted
0.1	0.0626	0.0626	0.9374	0.0611	0.9389
0.5	0.3063	0.3057	0.6943	0.3047	0.6953
1	0.5592	0.5485	0.4515	0.4284	0.5716
2	0.8943	0.8741	0.1259	0.4485	0.5515
3	1.0000	0.8741	0.1259	0.4485	0.5515
4	1.0000	0.8741	0.1259	0.4485	0.5515
5	1.0000	0.8741	0.1259	0.4485	0.5515



(a) Varying MUs' computation resources



(b) Varying the MSP's budget

Fig. 10: The MSP's profit when computation resources of MUs and MSP's incentive budget increase.

increases beyond 20 monetary units, both the MSP's profit and the portion of the encrypted data at the MU remain relatively constant. In this case, the FLEET can yield a profit 1.99 and 1.58 times larger than those of *FedAvg* and FLEET-CS, respectively. The profit for the *FedAvg* shown in TABLE IV remains unchanged in this experiment since the budget of 10 units is sufficient to incentivize the MUs to train 55.92% of their datasets (note that the percentage is less than 100% due to the limited MUs' computation resources). On the other hand, the profit for the MSP in FLEET-CS grows gradually from 10 to 50 monetary units. This is thanks to a higher profit gain from the training process at the CS when providing additional incentives to the MUs for encrypting and caching data. These results suggest that with an adequate budget for incentives and sufficient computational resources, the MSP can enhance its profits by encrypting and caching larger portions of local data at the MENs, which then incurs minimal costs for data encryption and caching.

TABLE IV: The average portions of local and encrypted data at an MU as the MSP's incentive budget changes.

I	*FedAvg*	FLEET-CS		FLEET	
	Local	Local	Encrypted	Local	Encrypted
10	0.5592	0.5592	0.0083	0.3489	0.6511
20	0.5592	0.5592	0.0865	0.4507	0.5493
30	0.5592	0.5592	0.1647	0.4389	0.5611
40	0.5592	0.5592	0.2435	0.4401	0.5599
50	0.5592	0.5592	0.3211	0.4482	0.5518

VII. Conclusion

In this paper, we have proposed a novel privacy-preserving FL framework to mitigate the straggling problem in the MEC network. Specifically, we have utilized the homomorphic encryption method that enables the participating MUs to encrypt their raw data prior to uploading them to the CS or nearby MENs for caching and remote training processes. In order to facilitate encrypted training at the MENs/CS, we have developed an efficient HE-based ciphertext packing method that exploits the single instruction multiple data technique. Building upon this approach, we then formulated an optimization problem aimed at identifying the optimal portions of encrypted data that can be cached and trained at the MENs/CS. The objective of this optimization problem is to maximize the MSP's profit, while taking into account various constraints such as available computation resources at the MUs and MENs, the MSP's budget for caching and training, and the deadline for each learning round. We also have proved that the optimization problem is convex, and thus the optimal solution can be efficiently obtained by using the interior point method. Through the experimental results, we have shown that our proposed framework can significantly enhance the MSP's profit and achieve the superior model accuracy and convergence speed compared with other baseline FL methods. Future works include the implementation of FL in dynamic environments, in which the MUs receive new data classes in an online fashion. Here, the proposed framework can be extended to continuously learn and cache the data of new classes to enhance the model's convergence speed and maintain system stability. Additionally, the training performance with the encrypted datasets can be further improved by using more sophisticated ML techniques, e.g., dataset pruning or few-shot learning.

Acknowledgment

This research was supported in part by the Australian Research Council under the DECRA project DE210100651. Additionally, the work of V.-D. Nguyen was supported in part by the VinUniversity Seed Grant Program.

References

[1] N. Rieke *et al.*, "The future of digital health with federated learning," *NPJ digital medicine*, vol. 3, no. 1, p. 119, Sep. 2020.

[2] H. Elayan, M. Aloqaily, and M. Guizani, "Digital twin for intelligent context-aware iot healthcare systems," *IEEE Internet Things J.*, vol. 8, no. 23, pp. 16749–16757, Dec. 2021.

[3] Y. Tian *et al.*, "Robust and privacy-preserving decentralized deep federated learning training: Focusing on digital healthcare applications," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, pp. 1–12, Mar. 2023.

[4] J. P. Usuga Cadavid *et al.*, "Machine learning applied in production planning and control: a state-of-the-art in the era of industry 4.0," *Journal of Intell. Manuf.*, vol. 31, pp. 1531–1558, Jan. 2020.

[5] W. Sun *et al.*, "Adaptive federated learning and digital twin for industrial internet of things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5605–5614, Oct. 2020.

[6] K. Gu, Y. Zhang, and J. Qiao, "Ensemble meta-learning for few-shot soot density recognition," *IEEE Trans. Ind. Informat.*, vol. 17, no. 3, pp. 2261–2270, Apr. 2020.

[7] M. Xu *et al.*, "Wireless edge-empowered metaverse: A learning-based incentive mechanism for virtual reality," in *Proc. IEEE Int. Conf. Commun. (ICC)*. IEEE, May 2022, pp. 5220–5225.

[8] J. Kang *et al.*, "Blockchain-based federated learning for industrial metaverses: Incentive scheme with optimal aoi," in *2022 IEEE International Conference on Blockchain*. IEEE, Aug. 2022, pp. 71–78.

[9] J. Konečný *et al.*, "Federated learning: Strategies for improving communication efficiency," *arxiv:1610.05492*, 2017.

[10] O. A. Wahab *et al.*, "Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1342–1397, Feb. 2021.

[11] W. Y. B. Lim *et al.*, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2031–2063, Apr. 2020.

[12] Z. Chai *et al.*, "FedAT: A high-performance and communication-efficient federated learning system with asynchronous tiers," *arxiv:2010.05958*, 2021.

[13] Z. Xu *et al.*, "Helios: Heterogeneity-aware federated learning with dynamically balanced collaboration," *arxiv:1912.01684*, 2021.

[14] Z. Ji *et al.*, "Computation offloading for edge-assisted federated learning," *IEEE Trans. Veh. Technol.*, vol. 70, no. 9, pp. 9330–9344, Sep. 2021.

[15] S. Prakash *et al.*, "Coded computing for low-latency federated learning over wireless edge networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 233–250, Jan. 2021.

[16] Y. Chen *et al.*, "Asynchronous online federated learning for edge devices with non-IID data," in *2020 IEEE International Conference on Big Data (Big Data)*, Dec. 2020, pp. 15–24.

[17] M. Chen, B. Mao, and T. Ma, "FedSA: A staleness-aware asynchronous federated learning algorithm with non-IID data," *Future Generation Computer Systems*, vol. 120, pp. 1–12, Jul. 2021.

[18] N. Yoshida *et al.*, "Hybrid-FL for wireless networks: Cooperative learning mechanism using non-IID data," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–7.

[19] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in IoT," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5986–5994, Nov. 2019.

[20] L. U. Khan *et al.*, "Federated learning for edge networks: Resource optimization and incentive mechanism," *IEEE Commun. Mag.*, vol. 58, no. 10, pp. 88–93, Oct. 2020.

[21] N. J. H. Marcano *et al.*, "On fully homomorphic encryption for privacy-preserving deep learning," in *2019 IEEE Globecom Workshops*, Dec. 2019, pp. 1–6.

[22] Z. Yue *et al.*, "Privacy-preserving time-series medical images analysis using a hybrid deep learning framework," *ACM Trans. Internet Technol.*, vol. 21, no. 3, pp. 1–21, Jun. 2021.

[23] M. Albrecht *et al.*, *Homomorphic Encryption Standard*. Springer International Publishing, Jan. 2021, pp. 31–62.

[24] Y. Mao *et al.*, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, Aug. 2017.

[25] G. M. Weiss, K. Yoneda, and T. Hayajneh, "Smartphone and smartwatch-based biometrics using activities of daily living," *IEEE Access*, vol. 7, pp. 133190–133202, Sep. 2019.

[26] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *arxiv:1903.03934*, 2019.

[27] V.-D. Nguyen *et al.*, "Fedfog: Network-aware optimization of federated learning over wireless fog-cloud systems," *IEEE Transactions on Wireless Communications*, vol. 21, no. 10, pp. 8581–8599, Oct. 2022.

[28] M. R. Sprague *et al.*, "Asynchronous federated learning for geospatial applications," in *ECML PKDD 2018 Workshops*, Mar. 2019, pp. 21–28.

[29] D. Wu *et al.*, "FedAdapt: Adaptive offloading for IoT devices in federated learning," *arxiv:2107.04271*, 2022.

[30] K. Lee *et al.*, "Speeding up distributed machine learning using codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, pp. 1514–1529, Mar. 2018.

[31] R. Tandon *et al.*, "Gradient coding: Avoiding stragglers in distributed learning," in *Proc. Int. Conf. Mach. Learn.*, Aug. 2017, pp. 3368–3376.

[32] S. Dhakal *et al.*, "Coded federated learning," in *2019 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2019, pp. 1–6.

[33] S. Prakash *et al.*, "Hierarchical coded gradient aggregation for learning at the edge," in *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 2616–2621.

[34] M. N. Krishnan, A. Thomas, and B. Sasidharan, "Hierarchical coded gradient aggregation based on layered mds codes," in *2023 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2023, pp. 2547–2552.

[35] R. Schlegel *et al.*, "Codedpaddedfl and codedsecagg: Straggler mitigation and secure aggregation in federated learning," *IEEE Transactions on Communications*, 2023.

[36] Y. Sun *et al.*, "Stochastic coded federated learning with convergence and privacy guarantees," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Aug. 2022, pp. 2028–2033.

[37] K. Nandakumar *et al.*, "Towards deep neural network training on encrypted data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2019, pp. 40–48.

[38] E. Hesamifard *et al.*, "Privacy-preserving machine learning as a service," *Proc. Priv. Enhancing Technol.*, vol. 2018, no. 3, pp. 123–142, Mar. 2018.

[39] E. Hesamifard, H. Takabi, and M. Ghasemi, "Deep neural networks classification over encrypted data," in *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy*, Mar. 2019, pp. 97–108.

[40] S. Meftah *et al.*, "DOReN: toward efficient deep convolutional neural networks with fully homomorphic encryption," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 3740–3752, Jun. 2021.

[41] J.-W. Lee *et al.*, "Privacy-preserving machine learning with fully homomorphic encryption for deep neural network," *IEEE Access*, vol. 10, pp. 30039–30054, Mar. 2022.

[42] M. Kim *et al.*, "Secure human action recognition by encrypted neural network inference," *Nat. Commun.*, vol. 13, no. 1, p. 4799, Aug. 2022.

[43] J. H. Cheon *et al.*, "Homomorphic encryption for arithmetic of approximate numbers," in *Advances in Cryptology – ASIACRYPT 2017*, Nov. 2017, pp. 409–437.

[44] P.-E. Clet, O. Stan, and M. Zuber, "BFV, CKKS, TFHE: Which one is the best for a secure neural network evaluation in the cloud?" in *Applied Cryptography and Network Security Workshops*, Jul. 2021, pp. 279–300.

[45] J. Kang *et al.*, "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10700–10714, Dec. 2019.

[46] N. Ferdinand *et al.*, "Anytime minibatch: Exploiting stragglers in online distributed optimization," *arXiv:2006.05752*, 2020.

[47] X. Jiang *et al.*, "Secure outsourced matrix computation and application to neural networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 1209–1222.

[48] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "GAZELLE: A low latency framework for secure neural network inference," in *27th USENIX Security Symposium*, Aug. 2018, pp. 1651–1669.

[49] E. Lee *et al.*, "Low-complexity deep convolutional neural networks on fully homomorphic encryption using multiplexed parallel convolutions," in *International Conference on Machine Learning*. PMLR, 2022, pp. 12 403–12 422.

[50] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2224–2287, Mar. 2019.

[51] L. Xu *et al.*, "Privacy or utility in data collection? A contract theoretic approach," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 7, pp. 1256–1269, Oct. 2015.

[52] P. A. Samuelson and W. D. Nordhaus, *Microeconomics*. Boston, MA, USA: McGraw-Hill Education, 2005.

[53] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[54] R. H. Byrd, M. E. Hribar, and J. Nocedal, "An interior point algorithm for large-scale nonlinear programming," *SIAM J. Optim.*, vol. 9, no. 4, pp. 877–900, Apr. 1999.

[55] B. Halvarsson *et al.*, "5g NR testbed 3.5 GHz coverage results," in *Proc. IEEE 87th Veh. Technol. Conf. (VTC Spring)*, Jun. 2018, pp. 1–5.

[56] Z. Wang *et al.*, "A WiFi-direct based local communication system," in *Proc. IEEE/ACM 26th Int. Symp. Qual. Service (IWQoS)*, Jun. 2018, pp. 1–6.

[57] N. Kim *et al.*, "Incentive-based coded distributed computing management for latency reduction in iot services—a game theoretic approach," *IEEE Internet Things J.*, vol. 8, no. 10, pp. 8259–8278, May 2020.

[58] *Lenovo ThinkSystem SE350 Edge Server*, Lenovo, accessed: 08-Mar-2023. [Online]. Available: https://lenovopress.lenovo.com/LP1168

[59] "Exynos w920: Wearable processor," Samsung Semiconductor Global, accessed: 08-Mar-2023. [Online]. Available: https://semiconductor.samsung.com/processor/wearable-processor/exynos-w920/

[60] B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*, Apr. 2017, pp. 1273–1282.

APPENDICES

APPENDIX A

Fig. 11 illustrates the batch forwarding process of an NN layer using the implemented ciphertext packing method. In this example, the input and output sizes are 5 and 3, respectively, and the input batch contains 3 training samples. First, the encrypted extended diagonals are multiplied by the encrypted input batch. Then, these encryptions are re-aligned using the rotation operator and added together. The resulting ciphertext includes multiple 3-slot chunks, each containing partial sums of the vector-matrix product. All these chunks are accumulated using the rotation operator to obtain the ciphertext $\mathbf{z}$, which represents the linear transformation of the input batch $\mathbf{f}$. Finally, the HE-approximated activation function is applied over the ciphertext $\mathbf{z}$ to obtain the output batch $\mathbf{a}$ of the layer.

APPENDIX B
PROOF OF THEOREM 1

First, we introduce that $\gamma_k = \zeta_k \eta_k f_k^2 \alpha_k + \beta_k$. Next, we respectively transform (29) and (30

$$P_k = \lambda_k \left[\sum_{n=1}^{N} x_{n,k} b_n \right]^{\frac{1}{2}} - \gamma_k \sum_{n=1}^{N} x_{n,k} b_n, \text{ and} \tag{32}$$

$$P_n = \lambda_n \left[\left(1 - \sum_{k=1}^{K} x_{n,k} \right) b_n \right]^{\frac{1}{2}} - \rho_n \left(1 - \sum_{k=1}^{K} x_{n,k} \right) b_n. \tag{33}$$

Then, we can calculate the first order partial derivatives of P_k and P_n with respect to $\mathbf{x}$ as follows::

$$\begin{aligned} \nabla P_k &= \left[\frac{\partial P_k}{\partial x_{1,1}}, \ldots, \frac{\partial P_k}{\partial x_{1,M}}, \ldots, \frac{\partial P_k}{\partial x_{n,k}}, \ldots, \frac{\partial P_k}{\partial x_{J,M}} \right] \\ &= \left[0, \ldots, \frac{\partial P_k}{\partial x_{n,k}}, \ldots, 0 \right], \text{ and} \end{aligned} \tag{34}$$

$$\begin{aligned} \nabla P_n &= \left[\frac{\partial P_n}{\partial x_{1,1}}, \ldots, \frac{\partial P_n}{\partial x_{1,M}}, \ldots, \frac{\partial P_n}{\partial x_{n,k}}, \ldots, \frac{\partial P_n}{\partial x_{J,M}} \right] \\ &= \left[0, \ldots, \frac{\partial P_n}{\partial x_{n,k}}, \ldots, 0 \right], \end{aligned} \tag{35}$$

where

$$\frac{\partial P_k}{\partial x_{n,k}} = \frac{1}{2} \lambda_k b_n \left[\sum_{n=1}^{N} x_{n,k} b_n \right]^{-\frac{1}{2}} - \gamma_k b_n, \tag{36}$$

$$\frac{\partial P_n}{\partial x_{n,k}} = -\frac{1}{2} \lambda_n b_n \left[\left(1 - \sum_{k=1}^{K} x_{n,k} \right) b_n \right]^{-\frac{1}{2}} + \rho_n b_n. \tag{37}$$

Also, we can calculate the second partial derivative of P_k, i.e., $\mathbf{H}_k = \nabla^2 P_k$. Subsequently, we can obtain the general expressions for the second derivative components by.

$$\frac{\partial^2 P_k}{\partial^2 x_{n,k}} = -\frac{1}{4} \lambda_k b_n^2 \left[\sum_{n=1}^{N} x_{n,k} b_n \right]^{-\frac{3}{2}}, \tag{38}$$

$$\frac{\partial^2 P_k}{\partial x_{n,k} \partial x_{n^\dagger,m}} = -\frac{1}{4} \lambda_k b_n b_{n^\dagger} \left[\sum_{n=1}^{N} x_{n,k} b_n \right]^{-\frac{3}{2}}, \forall n^\dagger \neq n, \tag{39}$$

$$\frac{\partial^2 P_k}{\partial x_{n,k} \partial x_{j,k^\dagger}} = \frac{\partial^2 P_k}{\partial x_{n,k} \partial x_{n^\dagger,k^\dagger}} = 0, \forall n^\dagger \neq n, \forall k^\dagger \neq k. \tag{40}$$

Similarly, we can write $\mathbf{H}_n = \nabla^2 P_n$, and derive that

$$\begin{aligned} \frac{\partial^2 P_n}{\partial^2 x_{n,k}} &= \frac{\partial^2 P_k}{\partial x_{n,k} \partial x_{j,k^\dagger}} \\ &= -\frac{1}{4} \lambda_n b_n^2 \left[\left(1 - \sum_{k=1}^{K} x_{n,k} \right) b_n \right]^{-\frac{3}{2}}, \forall k^\dagger \neq k, \end{aligned} \tag{41}$$

$$\frac{\partial^2 P_k}{\partial x_{n,k} \partial x_{n^\dagger,m}} = \frac{\partial^2 P_k}{\partial x_{n,k} \partial x_{n^\dagger,k^\dagger}} = 0, \forall n^\dagger \neq n, \forall k^\dagger \neq k. \tag{42}$$

Given a real vector $\mathbf{x} \in \mathbb{R}^{(N*K)\times 1}$ with $0 \leq x_{n,k} \leq 1, \forall n \in \mathcal{N}, \forall k \in \mathcal{K}$, it can be concluded that $\mathbf{x}^T \mathbf{H}_k \mathbf{x} \leq 0$ and $\mathbf{x}^T \mathbf{H}_n \mathbf{x} \leq 0$, where $\mathbf{H}_k, \mathbf{H}_n, \forall n \in \mathcal{N}, \forall k \in \mathcal{K}$ are negative semi-definite matrices. As a result, $P_k, \forall k \in \mathcal{K}$ and $P_n, \forall n \in \mathcal{N}$ are concave functions with respect to vector $\mathbf{x}$. To this end, the objective function, which is calculated by $\left[\sum_{n=1}^{N} P_n + \sum_{k=1}^{K} P_k \right]$, is also a concave function.

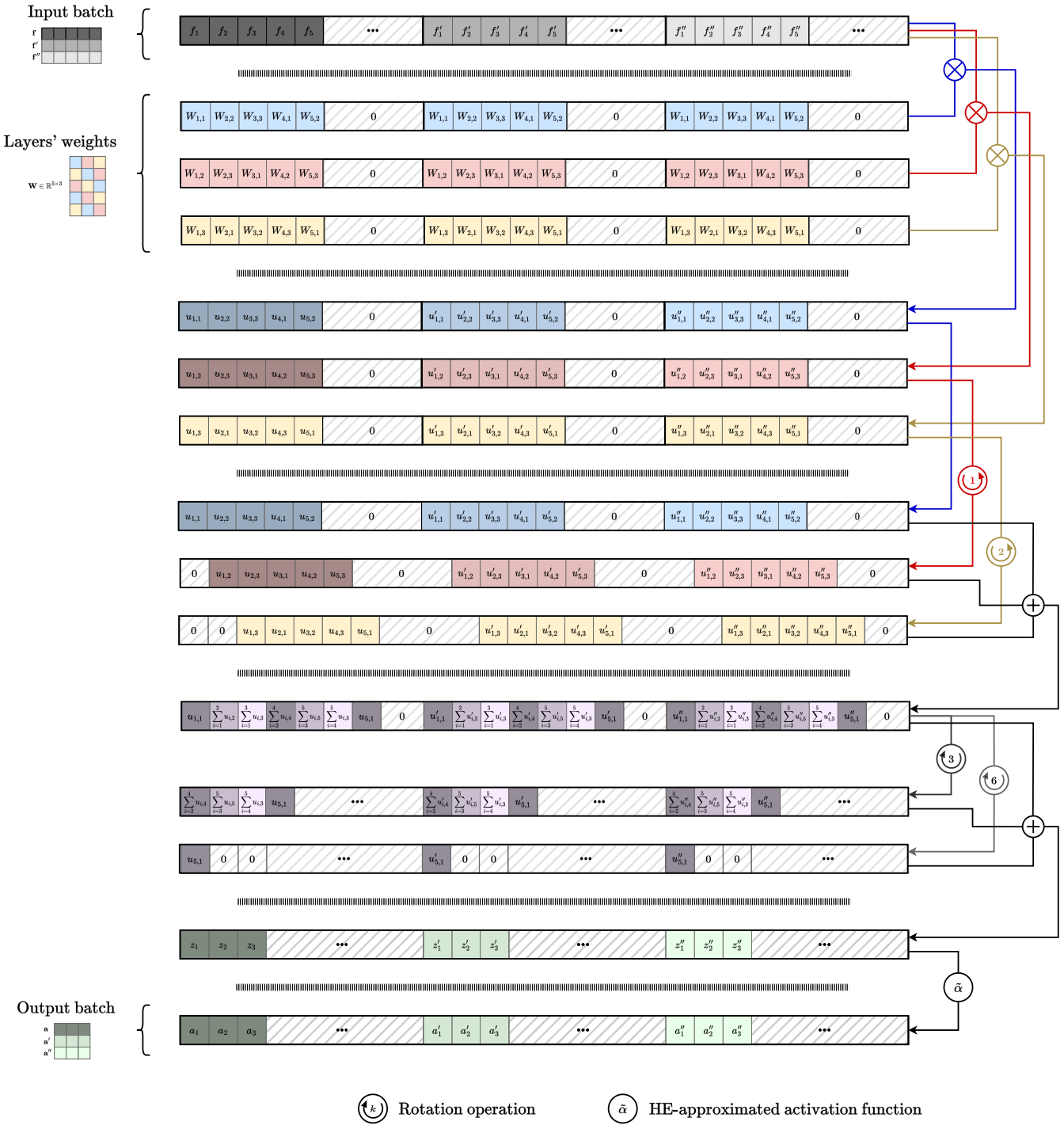


Fig. 11: Illustration of the batch forwarding process of a NN layer using the implemented ciphertext packing method.