EdgeC3: Online Management for Edge-Cloud Collaborative Continuous Learning

Shaohui Lin[1], Xiaoxi Zhang[1], Yupeng Li[2], Carlee Joe-Wong[3], Jingpu Duan[4], Xu Chen[1]
[1]*Sun Yat-sen University*, [2]*Hong Kong Baptist University*, [3]*Carnegie Mellon University*, [4]*Peng Cheng Laboratory*
{linshh65}@mail2.sysu.edu.cn, {zhangxx89, chenxu35}@mail.sysu.edu.cn, ypengl@hkbu.edu.hk,
cjoewong@andrew.cmu.edu, duanjp@pcl.ac.cn

***Abstract*—Deep learning (DL) powered real-time applications usually need continuous training using data streams generated geographically. Enabling data offloading among computation nodes through model training is promising to mitigate the problem that devices generating large datasets may have low computation capability. However, offloading can compromise model convergence and incur communication costs, which must be balanced with the cost spent on computation and model synchronization. Therefore, this paper proposes EdgeC3, a novel framework that can optimize the frequency of model aggregation and dynamic offloading for continuously generated data streams, navigating the trade-off between long-term accuracy and cost. We first provide a new error bound to capture the impacts of data dynamics that are varying over time and heterogeneous across devices. Based on the bound, we design a two-timescale online optimization framework. We periodically learn the synchronization frequency to adapt with uncertain future offloading and network changes. In the finer timescale, we manage online offloading by extending Lyapunov optimization techniques to handle an unconventional setting, where our long-term global constraint can have abruptly changed aggregation frequencies that are decided in the longer timescale. Finally, we theoretically prove the convergence of EdgeC3 by integrating the coupled effects of our two-timescale decisions, and we demonstrate its advantage through extensive experiments.**

I. Introduction

Nowadays, a large number of emerging applications, such as the Internet of Things and video analytics, are powered by deep learning (DL) approaches [1]. Using a sufficient amount of data generated by geographically dispersed terminal devices, the DL models can be well trained to make predictions or output decisions for those applications. Since training deep neural network (DNN) models is data-intensive, sending all the data to the cloud for iterative training and periodic testing is expensive and time-consuming. In contrast, the emerging mobile edge computing (MEC) technology provides an efficient solution by enabling data processing at edge servers [2] and computationally capable mobile devices in close proximity to the data sources.

Enabling the offloading of data training from devices to edge servers also realizes the popular distributed machine learning (DML) architecture where multiple computation nodes perform the training tasks in parallel so as to speed up the model convergence [3]. More importantly, offloading is promising in balancing the geo-distributions of data and computation resources. For example, surveillance cameras on collaborative drones may generate huge numbers of data streams but need to offload the training data to different edge servers depending on their locations. Smart vehicles may join crowdsensing tasks to monitor the traffic or air condition across different cities, contribute their computation to perform a DML task. Indeed, in some cases devices may actually wish to discard data entirely, if the data does not significantly change the model but processing it incurs large computing or communication cost. However, most existing DML studies overlook the flexibility of offloading, mainly due to the following complexity challenges. 1) The computation and communication capabilities in the network exhibit large heterogeneity; 2) It is hard to quantify the model accuracy under changing and heterogeneous data amounts over computation nodes; 3) Dynamic data offloading incurs complex tradeoffs between cost consumption and model accuracy.

Apart from spatial complexity, the fourth challenge is that real-time edge DL-based applications, such as video analytics, also encounter temporal uncertainty. Unlike a pre-determined static dataset assumed by classic DML studies, continuously generated training and testing data will cause severe data shift [4]. Continuous learning can mitigate the data shift problem but requires extra work on model construction or lacks convergence guarantee. This work proposes to quantify the model convergence with respect to the spatial heterogeneity and temporal uncertainty simultaneously.

We then design efficient algorithms to navigate the trade-off between model accuracy and long-term cost expenditure, addressing the above challenges. More concretely, we focus on distributed ML training with periodical model aggregation and data offloading enabled, similar to cooperative federated learning (CFL) [5], but we consider a configurable aggregation frequency and fresh streaming data. Periodically aggregating local models can significantly save communication overhead at the expense of a lower accuracy. It is therefore critical

This work was supported by NSFC grants 62102460 and 62202402, grant 202201011392 under the Guangzhou Science and Technology Plan Project, Guangdong Basic and Applied Basic Research Foundation (grants 2022A1515011583, 2023A1515011562 and 2023A1515012982), Young Outstanding Award under the Zhujiang Talent Plan of Guangdong Province, Germany/Hong Kong Joint Research Scheme sponsored by the Research Grants Council of Hong Kong and the German Academic Exchange Service of Germany (G-HKBU203/22), NSF grant CNS-2106891, One-off Tier 2 Start-up Grant (2020/2021) of Hong Kong Baptist University (Ref. RC-OFSGT2/20-21/COMM/002), and Startup Grant (Tier 1) for New Academics AY2020/21 of Hong Kong Baptist University. The corresponding author is Xiaoxi Zhang.

to determine the aggregation frequency so that the available resource is most efficiently used in increasing model accuracy. Besides, the cost spent on computation, model synchronization, and data offloading need to be carefully balanced especially when the total budget of cost expenditure is limited. Existing FL studies have analyzed the impact of various hyper-parameters [6], [7] but overlooked the trade-off between the aggregation frequency and data offloading, and they rarely provide accuracy analysis for data shifts.

To the best of our knowledge, this is the first performance-guaranteed two-timescale online optimization framework *that jointly decides dynamic data offloading and the aggregation frequency of cooperative continuous training using heterogeneous devices and data streams.* To achieve this, we make the following **technical contributions**.

First, we propose a new online continuous training framework customized for the edge-cloud collaborative network. In deciding which devices should process which data points, our formulation accounts for resource limitations and model accuracy. While ideally more data would be processed at devices with more computing resources, sending data samples to such devices may overburden the network. Moreover, processing too many data samples can incur large processing costs relative to the gain in model accuracy. We theoretically derive a bound of the training error when data can move between devices and the global aggregation frequency may change dynamically.

Second, we propose a novel two-timescale online learning algorithm to jointly optimize the aggregation frequency and data offloading, which are decided at the beginning of each training round and at a finer timescale, respectively. Since the best aggregation frequency should be dependent with network characteristics, data distributions, and time-varying offloading outcomes, we decide it by learning from sequential feedbacks of model accuracy. We then optimize the offloading decisions by extending Lyapunov optimization into unconventional scenarios where the constraints may change abruptly, accounting for the affects of modified aggregation frequencies.

Third, we perform both rigorous theoretical analysis and extensive experiments to demonstrate the efficacy and advantages of EdgeC3. We first prove a bounded long-term performance gap (a.k.a. regret) of the training accuracy between our algorithm and the expected offline optimum. We then conduct test-bed experiments where distributed training is performed using heterogeneous and time-varying datasets at edge computation nodes. Experimental results show that our two-timescale online algorithm can make best use of limited resource and the cost budget to achieve higher model accuracy.

II. Related Work

We summarize the prior results into three categories and highlight the aspects in which they differ from our work.

Cooperative Federated Learning (CFL) [5], [8]–[10] has emerged as a new paradigm which is built on device-to-device interactions and edge-cloud collaboration. While traditional FL [6] deployment prohibits communication between distributed clients who train local models using only private data, CFL instead focuses on the distributed system built on trusted network nodes. By enabling data offloading, it leverages the interaction between clients to unleash the potential of inter-network collaboration. Sharing the model training schemes of CFL, studies of classic FL algorithms [11], [12] started considering optimizing the frequency of global aggregations under constrained network and computing resources; some [10] focused on network topology between nodes to optimize the tradeoffs between communication, computation, and model performance in federated learning. Unlike our continuous collaborative learning scenario, theirs do not have time-varying data generation and movement or volatile network capabilities.

Computation offloading allows data to be moved from one device to another. Some works have considered splitting different layers of the neural network and offloading some of them to the edge or cloud. For instance, [13] proposed a distributed computing hierarchies, where the edge and terminal devices use the shallow part of the neural network for fast and local reasoning; [14] designed a scheme to optimally partition the DNN under different network conditions; and [9] consider sharing parameters of different layers. In addition, task offloading strategies are widely studied based on different requirements, such as mobility-aware [15], data quality-aware [16] and multi-user context-aware [17]. However, these works are not for collaborative learning with periodical model aggregations.

Online data management has been studied for DNN training or federated learning. Existing literature have proposed gradient update methods for training at streaming data to achieve faster convergence, e.g., SAG [18], SDCA [19]. SVRG [20], SAGA [21]. An online modification of the Nelder-Mead optimisation technique [22] called SPT is designed for hyper-parameter tuning, to address the issue of variability in data flows. However, these works are not for distributed training in dynamic edge networks. To handle dynamically incoming training data samples for edge ML, CEFL [23] designed an online control system with admission control, load balancing, data scheduling, and accuracy adjustment. But model convergence and the aggregation frequency are not analyzed in these efforts. A few other works build online algorithms that can cope with dynamic network settings for DML to satisfy the resource and accuracy restrictions, [24]–[26], but they do not consider data offloading and its effects on improving the model accuracy.

III. System Model

In this section, we formalize the problem setting of our cloud-edge collaborative continuous learning framework. To simplify our definitions, we define $[X]$ to be the set $\{1, \cdots, X\}$.

A. System architecture

We consider a set of N devices denoted by $\mathcal{N}$, a set of M edge servers denoted by $\mathcal{M}$, and a remote cloud. Data streams are generated and processed in T evenly distributed timesteps,

indexed by $t \in [T]$. Each device, e.g., a smartphone or smart vehicle, can both collect data and perform training. These data can be offloaded to any one of the edge servers or the remote cloud to collaboratively finish the ML model training. We consider a parameter-server (PS) architecture where multiple nodes are coordinated by a central PS, which is deployed in the cloud for reliability. For clarity, we use i and j to index a pair of devices, m to index an arbitrary edge server, and c to denote the remote cloud. The full set of our network *nodes* is $\mathcal{U} = \mathcal{M} \cup \mathcal{N} \cup \{c\}$, which includes the devices, edge servers, and the cloud. We then define its size as $U \triangleq M + N + 1$.

1) Data generation and offloading: In each timestep t, device $i \in \mathcal{N}$ dynamically collects a dataset denoted by $\mathcal{D}_i(t)$ with a size $D_i(t)$. We allow each device $i \in \mathcal{N}$ to split its data and offload them to different accessible nodes $u \in \mathcal{U}$.[1] Let $s_{iu} \in [0, 1]$ represent the fraction of data collected by device $i \in \mathcal{N}$ that is offloaded to node $u \in \mathcal{U}$ at time t. The number of data samples offloaded from i to u is $D_i(t)s_{iu}(t)$.[2] The size of data processed by device $i \in \mathcal{N}$ at t is then:

$$H_i(t) = D_i(t)s_{ii}(t) + \sum_{j \neq i} D_j(t)s_{ji}(t). \forall i, j \in \mathcal{N}. \quad (1)$$

The edge servers and cloud instead do not generate any data. Thus, the size of data processed by each of them at time t is:

$$H_k(t) = \sum_i D_i(t)s_{ik}(t). \forall i \in \mathcal{N}, \forall k \in \mathcal{M} \cup \{c\} \quad (2)$$

In addition, each device i can discard a $r_i(t) \in [0, 1]$ fraction of data in t, as processing entire datasets may incur very large computation cost but not improve model accuracy much.

2) Cost components and the long-term constraint: Let $\phi_u(t)$ and $\Phi_u(t)$ represent the cost of processing a data point and the maximum number of data points that can be processed at node u at t, respectively. These factors are heterogeneous and time-varying possibly due to co-existing (background) applications or (re-)charging functions activated at each device. In contrast, we assume that the cloud has abundant computing resources and thus remove the constraint of its capacity. We then define $\psi_{iu}(t)$ as the transmission cost of device i in transmitting a single data point to node u, and it has a prevailing budget $\Psi_{iu}(t)$. Intuitively, $\psi_{iu}(t)$ and $\Psi_{iu}(t)$ depend on factors such as available bandwidth and channel interference conditions. In addition, we define $\varpi_u(t)$ to be the amount of data (parameters) sent from u to the PS in the cloud for model aggregation at time t. The total budget[3] of our consumed cost over T timesteps is C. **Balancing the cost of offloading, local computation, and model synchronization under a long-term cost budget** is therefore critical to achieve efficient continuous training. We will elaborate the challenges of designing performance-guaranteed algorithms in Sec. IV-C.

[1]Our work still works if only a subset of nodes is accessible.

[2]When $u = i$, device i processes corresponding data on the device locally.

[3]The budget C is pre-determined by the task owner and possibly increased with the service time T For example, performing a video analytics training application in 1 week should have a larger C than running it in 3 hours.

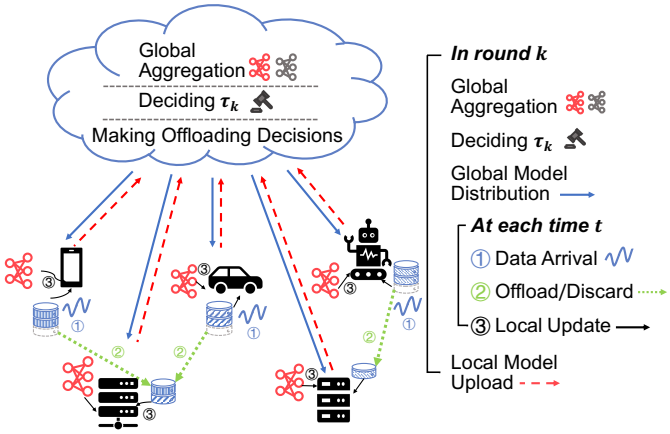


Fig. 1: Overview of EdgeC3 architecture.

B. Collaborative continuous training with dynamic datasets

Let $\mathbf{w}_u(t)$ represent the local model updated in time t using stochastic gradient descent (SGD) at node u and $\mathbf{w}$ denote the global model parameters aggregated in the cloud. Let $\mathcal{H}_u(t)$ represent the *prevailing dataset* that is used for local training at node u in t, which includes the data offloaded to and generated in u (see (1), (2)). For each local dataset $\mathcal{H}_u(t)$ with a size $H_u(t)$ at node u in t, the local loss function is:

$$F_u(\mathbf{w}_u(t), \mathcal{H}_u(t)) \triangleq \frac{1}{H_u(t)} \sum_{d \in \mathcal{H}_u(t)} f(\mathbf{w}_u(t), x_d, y_d), \quad (3)$$

where $f(\mathbf{w}_u(t), x_d, y_d)$ is the loss of each data point $d \triangleq (x_d, y_d)$ in $\mathcal{H}_u(t)$[4]. The objective function is to minimize the total loss of the global model, i.e., $F(\cdot) = \frac{\sum_{u \in \mathcal{U}} \mathcal{H}_u}{\sum_u H_u} F_u(\cdot)$. Each node u updates its parameters as follows:

$$\mathbf{w}_u(t) = \mathbf{w}_u(t-1) - \eta \nabla F_u(\mathbf{w}_u(t-1)), \quad (4)$$

where $\eta > 0$ is the learning rate and $\nabla F_u(\mathbf{w}_u(t-1))$ is the average gradient over the local dataset $H_u(t)$. Each kth *aggregation round* includes a number of $\tau_k \in \mathbb{Z}^+$ *local updates* and an *aggregation step* in timestep $t = \sum_k \tau_k, k \in \mathbb{Z}^+$, after which the local models are aggregated at the PS:

$$\mathbf{w}(t) = \mathbf{w}(t-1) - \eta \frac{\sum_u H_u(t-1) \nabla F_u(\mathbf{w}_u(t-1))}{\sum_u H_u(t-1)}. \quad (5)$$

Then, the PS broadcasts $\mathbf{w}(t)$ to each computation node, once the set of computation nodes for the next aggregation round is determined. Figure 1 depicts our EdgeC3 architecture.

Impact of aggregation frequency. Along with the offloading decisions, each τ_k needs to be carefully chosen. It plays a key role in: 1) balancing the cost of computation and communication for offloading datasets and transmitting model parameters, and 2) affecting the global model accuracy. A smaller τ_k can lead to a higher accuracy but also larger communication cost, leaving little remaining cost for data of-

[4]For simplicity of notation, we use $F_u(\mathbf{w}_u(t))$ interchangeably with $F_u(\mathbf{w}_u(t), \mathcal{H}_u(t))$ in the rest of this paper.

floading. The node will then be forced to do local computation and discard data, which impedes the accuracy finally.

IV. Problem Formulation and analysis

In this section, we present our joint optimization of aggregation frequency and data offloading with time-varying network characteristics and data generation. Our goal is to maximize the long-term training accuracy, subject to the constraints of the accumulated consumed cost and prevailing limits of communication and computation capabilities. We answer the question of how model accuracy is affected by the time-varying training data and aggregation frequency through an optimization problem shown in Section IV-B and a new convergence bound derived in Section IV-A.

A. Upper Bound on Convergence

Intuitively, more data processed and a higher aggregation frequency for model synchronization are both beneficial to improve the final accuracy. However, it is unclear how these two factors are coupled with the data movements and distribution shifts within each local dataset. To ensure tractable convergence analysis, we make the following assumptions:

Assumption 1. *The loss function $F_u(\mathbf{w})$ of each node u is convex, i.e., $F_u(\mathbf{w}) \geq F_u(\mathbf{v}) + (\mathbf{w}-\mathbf{v})^T \nabla F_u(\mathbf{v})$, ρ-Lipschitz, i.e., $||F_u(\mathbf{w}) - F_u(\mathbf{v})|| \leq \rho ||\mathbf{w}-\mathbf{v}||$, and β-smooth, i.e., $||\nabla F_u(\mathbf{w}) - \nabla F_u(\mathbf{v})|| \leq \beta ||\mathbf{w}-\mathbf{v}||$ for any $\mathbf{w}$ and $\mathbf{v}$.*

Assumption 2. *For any u and $\mathbf{w}$ at time t, there exists an upper bound of $\|\nabla F_u(\mathbf{w}(t)) - \nabla F(\mathbf{w}(t))\|$, which quantifies the degree of data shift, i.e., $\|\nabla F_u(\mathbf{w}(t)) - \nabla F(\mathbf{w}(t))\| \leq \delta_u(t)$, where $F(\cdot)$ is our global loss function.*

Assumption 1 is widely used in DML convergence studies [26], [27]. Nevertheless, our experimental results shown in Section VI demonstrate that our method works well for non-convex loss functions. Assumption 2 is customized for our continuous collaborative learning, since it captures the time-varying divergence between the local and global models, which is affected by the size and distribution dynamics of the generated data and our offloading decisions.

Lemma 1. *For each t that lies in any given aggregation round k with τ_k ($k \geq 1$) local updates, the gap between distributed parameters and the centralized parameter satisfy:*

$$||\mathbf{w}(t) - \mathbf{v}_k(t)|| \leq \frac{\eta}{2} \sum_{y=\sum_{l=1}^{k-1}\tau_l}^{t-1} \left((\delta^{(y)_k})^2 + h\big(y - \sum_{l=1}^{k-1}\tau_l\big)^2 \right).$$

Here, the auxiliary parameter $\mathbf{v}_k(t)$ equals $\mathbf{w}(t)$, if t is the time that local models are aggregated, and satisfies $\mathbf{v}_k(t) = \mathbf{v}_k(t-1) - \eta \nabla F(\mathbf{v}_k(t-1) | \cup_{u\in\mathcal{U}} \mathcal{H}_u(t))$ otherwise. Besides, we have $h(x) \triangleq ((\eta\beta+1)^x - 1)$, $\delta^{(t)_k} \triangleq \frac{\sum_{u=1}^{U} H_u(t)\delta_u^{(t)_k}}{\sum_{u=1}^{U} H_u(t)}$, $\delta_u^{(t)_k} \triangleq \max_{(\sum_{l=1}^{k-1}\tau_l) \leq y \leq t} \delta_u(y)$, and

$$\delta_u(t) = \|\nabla F_u(\mathbf{w}(t)|H_u(t)) - \nabla F(\mathbf{w}(t))\| \leq \frac{\gamma_u}{\sqrt{H_u(t)}} + \Lambda_u,$$

where $\Lambda_u = ||\nabla F_u(\mathbf{w}|\mathcal{H}_u) - \nabla F(\mathbf{w}| \cup_{u\in\mathcal{U}} \mathcal{H}_u(t))||$.

Lemma 1 relates the difference between local and global models to $\mathcal{H}_u(t)$, the amounts of data that each node actually processes. Based on this, we present our derived an upper bound of the training error between using collaborative learning where each node n computes gradients using $\mathcal{H}_u(t)$ for τ_k steps in each round k, denoted by $F(\mathbf{w}_t)$ for simplicity, and the optimal loss $F(\mathbf{w}^*)$ by processing these data centrally.

Theorem 1. *(Training error bound of EdgeC3). Under Assumptions 1 and 2, $\eta \leq \frac{1}{\beta}$, and $\epsilon = O(H_u(t)^{\frac{1}{2}})$, for any time t' in aggregation round k', the training error in any $t \geq t'$ in round k, denoted by $F(\mathbf{w}(t)) - F(\mathbf{w}^*)$, is at most:*

$$\frac{\eta\rho}{2\epsilon^2}\frac{1}{U^2}\sum_{t'=0}^{t-1}\Big(\sum_{u\in U}\big(\frac{\gamma_u}{\sqrt{H_u(t')}} + \Lambda_u\big)^2_{\hat{max}_{t'}} + h\big(t' - \sum_{k=1}^{k'-1}\tau_k\big)^2\Big) - t\omega\eta(1 - \frac{\beta\eta}{2}), \tag{6}$$

where $(X)_{\hat{max}_{t'}}$ is the maximum of X over each time in the round of t' until time t.

We provide the proof and method to select ϵ in [28].

B. Problem Formulation

Our goal is to jointly optimize (i) data offloading decisions $s_{ij}(t)$ and $r_i(t)$ and (ii) the aggregation frequency τ_k at each training round k so as to minimize the long-term error bound. Our optimization problem is then formulated as:

$$\underset{s_{iu}(t), r_i(t), \tau_k}{\text{minimize}} \quad \sum_{t=0}^{T-1} p(t) \tag{7}$$

$$\text{subject to:} \quad H_u(t) \leq \Phi_u(t),\ \forall u, t \tag{7a}$$

$$D_i(t) s_{iu}(t) \leq \Psi_{iu}(t),\ \forall i, u, t \tag{7b}$$

$$\sum_{t=0}^{T-1} C_{ca}(t) + \sum_{t\in\{\tau^{(k)}\}} C_{co}(t) \leq C \tag{7c}$$

$$r_i(t) + \sum_u s_{iu}(t) = 1,\ \forall i, u, t \tag{7d}$$

$$s_{iu}(t), r_i(t) \in [0,1],\ \forall i, u, t \tag{7e}$$

$$\tau_k \in [1, \tau_{max}],\ \forall k \tag{7f}$$

where the definitions of $p(t)$, $C_{ca}(t)$, and $C_{co}(t)$ are:

$$p(t) \triangleq \frac{\eta\rho}{2\epsilon^2}\frac{1}{U^2}\left(\sum_{u\in U}\left(\frac{\gamma_u}{\sqrt{H_u(t)}} + \Lambda_u\right)^2_{\hat{\max}_t} + h\big(t - \sum_{l=1}^{k-1}\tau_l\big)^2\right) - t\omega\eta(1 - \frac{\beta\eta}{2}),$$

$$C_{ca}(t) \triangleq \sum_u (H_u(t)\phi_u(t)) + \sum_i\sum_u D_i(t) s_{iu}(t)\psi_{iu}(t),$$

$$C_{co}(t) \triangleq \sum_{u\neq c} \varpi_u(t)\psi_{uc}(t).$$

Here, $H_u(t)$ represents the amount of data processed by each node u according to the definitions (1) and (2); $C_{ca}(t)$

is the total cost of local computation and data offloading; and $C_{co}(t)$ represents the cost due to model synchronization that happens only at time $\tau^{(k)} \triangleq \sum_{l=1}^{k-1} \tau_l + 1$, for each k until exceeding T. Constraints (7a) and (7b) ensure that the amounts of data processed at each node and transmitted through each link cannot exceed the node computation capacity and link bandwidth capacity, respectively. Constraint (7c) is our long-term cost constraint, which requires that the total cost spent on computation, offloading, and model synchronization cannot exceed the budget C. Equation (7d) ensures that all data collected by device i at t must either be processed or discarded. Finally, we consider that τ_k has an upper-limit $\tau_{max} \leq T$ decided according to the task owner, and we will theoretically analyze its impact on our algorithm performance.

C. Algorithmic Challenges

To reach the optimum of (7)–(7f) in an online manner, we encounter the following **technical challenges**:

1) Online uncertainty: All data arrive dynamically, and the cost of computation ($\phi_u(t)$) and communication ($\psi_{iu}(t)$) associated with every node changes over time. These can only be observed when time t starts, which means the problem needs to be solved online with the inputs are given on the fly at every time. Besides, since the aggregation frequency τ_k should be updated at the beginning of each round, but choosing τ_k should account for our offloading decisions to balance between the cost and model accuracy, further escalating the difficulty.

2) Long-term budget: Although dynamic and uncertain data generation and network qualities force us to make real-time decisions, we cannot perform data offloading and model synchronization greedily. The reason is that the long-term cost budget may require conservative expenditure to save cost for future use. However, with so many uncertain input parameters, splitting the cost into each time before the training task starts is not realistic. Thus, we need to handle the dependencies between our decisions in different times in an adaptive way.

3) High complexity: Even for the offline setting, where all the parameters over all times are revealed in advance, we still face a mixed integer programming problem with a non-linear objective function. It also requires co-optimization of two decision variables, resulting in high optimization complexity.

V. Two-timescale Online Algorithm Design

In this section, we propose a two-timescale online algorithm, where offloading is delay-sensitive and thus decided in a finer timescale while the aggregation frequency is decided once each aggregation round completes, as shown in Fig. 2. Our idea is to first decouple the joint-optimization problem into two sub-problems, as they interact with each other in only one constraint, the global budget constraint, and independently affect other constraints. We then integrate a Multi-Armed Bandits (MAB) [29] algorithm, which learns to optimize τ_k, into the Lyapunov Optimization framework for deciding the offloading in real-time. *The difficulty is to address and analyze the dependency between the decision making for the two variables and bounding the overall sub-optimality (i.e., regret).*

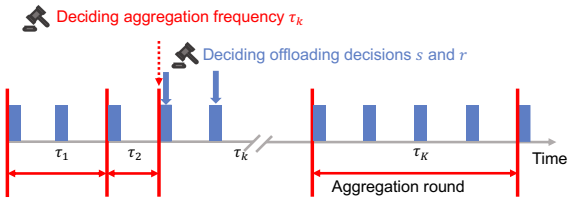


Fig. 2: Design of the online two-timescale schema.

To achieve this, we extend Lyapunov Optimization theories and the performance analysis into **an unconventional setting where the constraint can abruptly change due to the longer-timescale updates**, which drives our idea of using a window-based Lyapunov optimization algorithm.

A. Longer-timescale control: deciding aggregation frequency

Due to the fact that the future offloading decisions and network dynamics including the cost coefficients (e.g., $\psi_{iu}(t)$ and $\phi_v(t)$) and streaming datasets may change over time and haven't revealed when we need to decide τ_k in the longer-timescale. Therefore, we first decompose the longer timescale learning problem from (7)-(7f) and then define a fixed set of feasible arms $\{1, 2, \cdots, \tau_{max}\}$. The sub-problem for learning the best τ_k before each round starts is:

$$[\mathbb{P}1]: \ \text{minimize}_{\tau_k \in \{1,2,\cdots,\tau_{max}\}} \quad p(t) \tag{8}$$

We apply the upper confidence bound (UCB) [30] method for learning to optimize τ_k based on historical feedback of $p(t)$. As shown in line 9 of Algorithm 1, we choose the number of local updates τ_k, based on the UCB of each feasible choice τ, which is defined as $-\bar{p}(\tau) + \sqrt{2logK/T_\tau}$, where $\bar{p}(\tau)$ denotes the average value of $p(t)$ using τ local updates in history, and T_τ denotes the number of times that any $\tau \in [\tau_{max}]$ is selected. The first term in the UCB indicates that we tend to choose τ with the best performance so far, i.e., *exploitation*. Considering our objective function to be minimized, we take the opposite of $\bar{p}(t)$. The second is to increase the possibility of choosing arms that are selected less often, i.e., *exploration*. Algorithm 1 then greedily chooses the most suitable global aggregation frequency as:

$$\tau = \text{argmax}\{-\bar{p}(\tau) + \sqrt{2logK/T_\tau}\}, \tau \in [1, \tau_{max}] \tag{9}$$

Note that UCB has been widely used for online decision making, and we refer interested readers to [30] for more details. The merit of our method is beyond applying UCB and instead in i) effectively decomposing the two-timescale problem with a global constraint **associating the bandit actions and finer-timescale offloading decisions**, and ii) integrating online learning with extended Lyapunov Optimization techniques with performance analysis (see Section V-C).

B. Finer-timescale control: deciding data offloading

Given τ_k when round k starts, we need to solve $s_{iu}(t)$ and $r_i(t)$ at each t. We adopt the Lyapunov drift-plus-penalty

framework to construct a virtual queue $Q(t)$ as follows:

$$Q(t+1) = \max\{Q(t) + y(t), 0\} \tag{10}$$

$$y(t) = C_{ca}(t) + C_{co}(t)|_{t\in\{\tau^{(k)}\}} - \frac{C}{T}. \tag{11}$$

Through (11), we transform the global constraint (7c) to $\sum_t y(t) \leq 0$. The values of $Q(t)$ indicate how well a time-averaged cost constraint is satisfied, i.e., a larger violation of $y(t-1) \leq 0$ results in a larger $Q(t)$. The corresponding Lyapunov function is then defined as $L(\Theta(t)) = \frac{1}{2}Q(t)^2$. Next, the Lyapunov penalty drift can be formulated as:

$$\triangle L(\Theta(t)) = \mathbb{E}\left\{L(\Theta(t+1)) - L(\Theta(t))|\Theta(t)\right\}. \tag{12}$$

Finally, we wish to minimize a Lyapunov drift-plus-penalty:

$$\triangle L(\Theta(t)) + V\mathbb{E}\left\{p(t)|\Theta(t)\right\}, \tag{13}$$

where $V \geq 0$ is a coefficient that can be tuned by the owner of the training task to make a trade-off between the optimality and queue stability. A larger V means that we care more about the cost constraint, while a smaller V implies that we are willing to violate the constraint to some extend to achieve better model accuracy. We quantify this trade-off in Lemma 2. We then adopt the upper-bound of drift-plus-penalty [31] to transform the long-term constrained problem into a problem that the decisions are independent across times.

However, conventional Lyapunov optimization requires static constraints to guarantee bounded performance gap, but our global cost constraint may have an abruptly changed $y(t)$ when $t = \tau^{(k)}$. To address this, our idea is to use the current τ_k to estimate the communication cost per time for all future times to guarantee a *stable estimate* of $y(t)$. Formally, we modify (7c) as $\sum_{t=0}^{T-1} C_{ca}(t) + \frac{T}{\tau_k}C_{co}(t) \leq C$ and define the estimate of $y(t)$ for all t within round k as:

$$y(t) \triangleq C_{ca}(t) + \frac{1}{\tau_k}C_{co}(t) - \frac{C}{T}, \tag{14}$$

The offloading subproblem can be expressed as:

$$\begin{aligned} \min \quad & Vp(t) + Q(t)y(t) \qquad [\mathbb{P}2] \\ \text{s.t.} \quad & \text{constraint(7a)-(7b)} \\ \text{var.} \quad & s_{iu}(t), r_i(t) \in [0,1] \end{aligned} \tag{15}$$

This simple but efficient algorithm design is driven by our performance analysis. The key insight (a preview of our analysis in Section V-C) is that the upper-bound of the performance gap between our algorithm and the optimum can be decomposed into two gaps: 1) the gap in $p(t)$ between using our decisions, the sequence of τ_k and our offloading decisions $(\mathbf{s}, \mathbf{r})$, and that using optimal τ and our $(\mathbf{s}, \mathbf{r})$; 2) the gap in $p(t)$ between using optimal τ (denoted by τ^*) and our $(\mathbf{s}, \mathbf{r})$ against the final optimum. The first gap is bounded by UCB algorithm under reactive feedbacks of using τ_k. The second gap can be bounded by Lyapunov optimization theories since τ^* is static and thus $y(t) \leq 0$ **using** (14) **is the necessary and sufficient condition of the original global constraint** (7c) **under** τ^*.

Note that solving $\mathbb{P}2$ is then easy by using existing standard optimization solvers, although parameters related to the model (e.g., ρ, β) need to be estimated in practice. We provide the pseudocode of the routine running at the PS in Algorithm 1 and that performed by each computation node in Algorithm 2.

Algorithm 1: Procedure in the cloud

1: **Input**: Budget C, control parameter φ,γ_n,$\nabla F_u(\mathbf{w}|\mathcal{D}_u)$, maximum τ value τ_{max}, lower bound of global loss ϵ, gradient descent step size η, online step size η_o, number of nodes U, trade-off parameter V;
2: Initialize $\tau \leftarrow 1$, $\hat{t} \leftarrow t \leftarrow 0$, $k \leftarrow 1$, $\mathbf{w}(0) \leftarrow 0$;
3: Send $\mathbf{w}(0)$ and τ to all nodes;
4: **repeat**
5: Receive $\phi_u(t), \Phi_u(t)$ and estimate $\psi_{iu}(t), \Psi_{iu}(t)$;
6: **if** $t - \hat{t} = \tau$ **then**
7: $\hat{t} \leftarrow t$; $k \leftarrow k+1$; update τ by (9);
8: Receive $\nabla F_u(\mathbf{w}_u(t))$;
9: Estimate $\nabla F(\mathbf{w}(t)) \leftarrow \max_u\{\nabla F_u(\mathbf{w}_u(t))\}$
10: Execute global update according to (5);
11: Calculate $\Lambda_u \leftarrow ||\nabla F_u(\mathbf{w}|\mathcal{D}_u) - \nabla F(\mathbf{w}(t))||$
12: Broadcast $\mathbf{w}(t)$ and τ and receive ρ_u, β_u;
13: Estimate $\rho \leftarrow max\{\rho_u\}, \beta \leftarrow max\{\beta_u\}$;
14: **end if**
15: **if** $t > 0$ **then**
16: $s^*, r^* \leftarrow$ **Solver**$(V, U, C, \rho, \eta, \epsilon, k, \tau, \varphi, D_u(t), \psi_{iu}(t), \phi_u(t), \Psi_{iu}(t), \Phi_u(t), \Lambda_u, \gamma_u)$;
17: Send s^*, r^* to all nodes;
18: **end if**
19: $t \leftarrow t+1$;
20: **until**

Algorithm 2: Procedure at each node u

1: Initialize $\hat{t} \leftarrow t \leftarrow 0$;
2: Receive $\mathbf{w}(0)$ and τ from the cloud;
3: **repeat**
4: Receive $D_u(t)$ data samples if $n \in \mathcal{N}$;
5: Send $\phi_u(t), \Phi_u(t)$ to the cloud;
6: **if** $t - \hat{t} = \tau$ **then**
7: $\hat{t} \leftarrow t$; send $\nabla F_u(\mathbf{w}_u(t))$ to the cloud;
8: Receive $\mathbf{w}(t)$ and τ from the cloud;
9: $\rho_u = ||F_u(\mathbf{w}_u(t)) - F_u(\mathbf{w}(t))||/||\mathbf{w}_u(t) - \mathbf{w}(t)||$;
10: $\beta_u = ||\nabla F_u(\mathbf{w}_u(t)) - \nabla F_u(\mathbf{w}(t))||/||\mathbf{w}_u(t) - \mathbf{w}(t)||$;
11: Send ρ_u, β_u to the cloud; $\mathbf{w}_u(t) \leftarrow \mathbf{w}(t)$;
12: **end if**
13: **if** $t > 0$ **then**
14: Receive s^*, r^* from the cloud and offload data;
15: **end if**
16: $t \leftarrow t+1$; execute local update according to (4);
17: **until**

Algorithm workflow. When each round k starts, τ_k is updated by our UCB algorithm, as shown in line 5-7 in 1. Similar to [12], we obtain $\nabla F_u(\mathbf{w}|\mathcal{D}_u)$ and $\nabla F(\mathbf{w}(t))$ through local computation and central tests, respectively. The values of ρ_u

and β_u at each node are calculated based on the local loss computed at $\mathbf{w}(t)$ and $\mathbf{w}_u(t)$. They are needed in each t to derive the offloading decision (lines 9-10 of Algorithm 2), so we use the global model parameter $\mathbf{w}(t)$ obtained in the latest aggregation step instead. When $\mathbf{w}_u(t) = \mathbf{w}(t)$, we update ρ_u and β_u to be zero. Similar to [12], we define that $h(\tau) = 0$ for all $\tau \geq 1$. This situation only occurs when different nodes have extremely similar datasets so that the value of τ will not have any impact on the accuracy of the model. We use the maximum value of each node as the estimates of ρ, β and $\nabla F(\mathbf{w}(t))$ (line 13 of Algorithm 1), which still yields a valid convergence rate based on our analysis. Given the estimated parameters, we can adopt a classic solver for convex optimization to find the optimal offloading decisions (line 16 of Algorithm 1). In each t, the computation nodes offload data according to s^* and r^* received from cloud and then update their local models.

C. Theoretical analysis

We now theoretically bound the performance gap (a.k.a. *regret*) between our two-timescale algorithm and the expected offline optimum, who knows all the input parameters and obtains the optimal solutions τ^*, $\mathbf{s}_t^*$, and $\mathbf{r}_t^*$. We extend the analysis of Lyapunov optimization since we use modified per-time constraint and periodically change the constraints due to modified τ_k each round, and we nicely embed the regret of UCB into the performance bound of our offloading decisions.

Lemma 2. *The gap of the objective value $\sum_{t=1}^{T} p(t)$ between using our offloading decisions $(\mathbf{s}_t, \mathbf{r}_t)$ and the optimum $(\mathbf{s}_t^*, \mathbf{r}_t^*)$, under the optimal number of local updates τ^* is:*

$$Regret_1 = \mathbb{E}\left[\sum_{t=0}^{T-1}\left(p(\mathbf{s}_t, \mathbf{r}_t, \tau^*) - \sum_{t=0}^{T-1} p(\mathbf{s}_t^*, \mathbf{r}_t^*, \tau^*)\right)\right] \leq \frac{T}{2V}[C_{ca} + \frac{C_{co}}{\tau_{max}} - \frac{C}{T}]_{max}^2 + \frac{\mathbb{E}[L(\Theta(0))]}{V}$$

Theorem 2. *The total performance gap between using our $(\mathbf{s}_t, \mathbf{r}_t, \tau_{k(t)})$ ($k(t)$ means the round of t) and the optima, defined as $Regret_{total} = \mathbb{E}[\sum_{t=0}^{T-1} p(\tau_{k(t)}, \mathbf{s}_t, \mathbf{r}_t) - \sum_{t=0}^{T-1} p(\tau^*, \mathbf{s}_t^*, \mathbf{r}_t^*)]$ is at most $\leq \mathcal{O}(\sqrt{K \log K} + \varsigma T)$, where $\varsigma < 1$ is a tunable parameter determined by τ_{max} and other parameters. (Details in computing ς are deferred into [28].)*

It shows that our regret grows sub-linearly with time, implying that the time-average gap approaches zero as T increases. Note that we can always find $\tau_{max} = TC_{co}/(C - TC_{co})$, so that the coefficient ς equals zero. But τ_{max} also affects the first term of the regret (the full form of regret is shown in [28]), and we leave the optimization of τ_{max} to our future work. The detailed proof can be found in our report [28].

VI. Experimental Validation

A. Experiment Setup

1) Training models: We evaluate the performance of the collaborative training task using a Convolutional Neural Network (CNN), with 2 convolutional layers, 2 max pooling layers, 2 local response normalization layers, 1 fully connection layer, and 1 softmax layer. It verifies that our method works for distributed training under non-convex objective functions. We use two types of data distributions: *1) i.i.d*, where each node has an entire dataset and *2) non-i.i.d*, where all the data points in each node have the same label. Both scenarios are constructed using two datasets: MNIST and CIFAR-10. We set the learning rate as $\eta = 0.01$, which is the same as [12] .

TABLE I: Experiment parameters.

$\Psi_{ij}(t)$	$\Psi_{ic}(t)$	$D_i(t)$	$\Phi_i(t)$	$\phi_u(t)$	$\varpi_u(t)$
[50, 150]	10^4	[20, 500]	[20, 200]	[1, 30]	200

2) Network characteristics: We utilize the 4G LTE dataset [32] to simulate the network dynamics. We map the "ULbitrate" in $[20, 300]$ to our time-varying transmission cost $\psi_{ij}(t)$. We randomly sample $\psi_{ic}(t)$ from $[10^7, 2 \times 10^7]$ to reflect the difference between the cloud and edge and show other parameters in Table I. The experiments are conducted on an Intel NUC desktop which serves as the cloud and a 12th Gen Intel(R) Core(TM) i9-12900K server to deploy 5 devices and 2 servers. Results in other experimental settings can be found in [28]. We adopt Sockets to realize data transmission and solve our problem $\mathbb{P}2$ using a standard CVX solver.

3) Baselines: We compare each of our two sub-algorithms with different baselines. First, we fix the aggregation frequency and compare different offloading algorithms.

- **Not offloading** does not offload any data and meets the local computation constraints.
- **Random offloading** uniformly at random selects the target node and offloading fractions in each timestep, with all the constraints satisfied.
- **Offloading** refers to our proposed offloading method.

We then compare with the following baselines to evaluate our chosen decisions of τ_k for model aggregation.

- **Fixed value** uses a fixed τ (= 1 or 10) and the same offloading as EdgeC3 throughout the training.
- **Dynamic** uses a state-of-the-art [12] strategy to choose τ_k and the same offloading as EdgeC3.

We finally compare with different algorithm combinations.

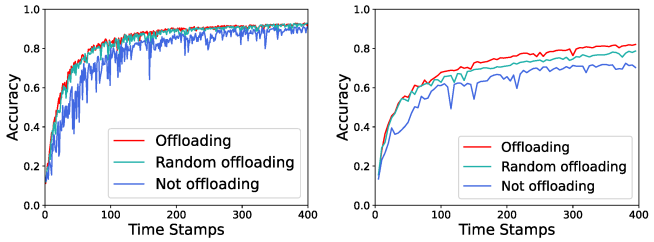
- **RandomAdaptive** uses the same τ_k as EdgeC3.
- **OffloadingDynamic** adopts our offloading decisions and combines [12] for choosing τ_k.
- **RandomDynamic** combines random offloading and [12].

In addition, in each time, there are three different data shift degrees: 1) **a**, each node has data points of all 10 classes, 2) **b**, the data points of each node are only 6 classes, 3) **c**, the data points of each node are only 1 class. For both **b** and **c**, the dataset is selected randomly from the 10 classes.

B. Evaluation Results

1) Performance of offloading: Fig. 3 shows the test accuracy of our approach under MNIST. Fig. 3(a) shows that using i.i.d. datasets, our offloading scheme has significant advantages over Not offloading. The overall difference between

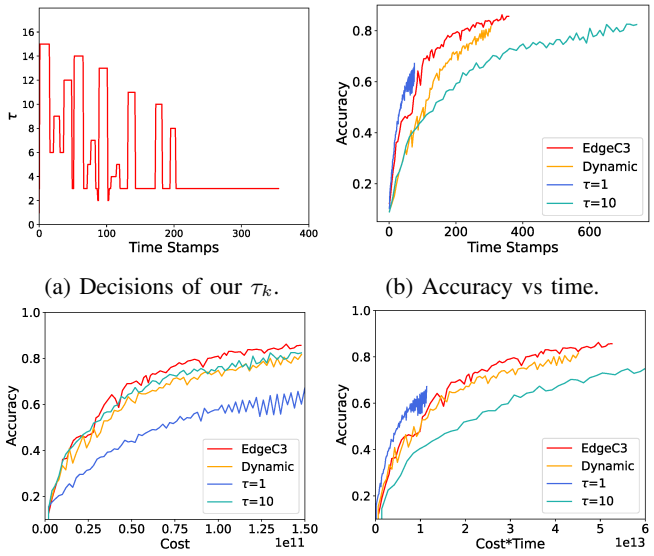
the random scheme and using our offloading solutions only in MNIST is not large. In contrast, Fig. 3(b) reveals that our algorithm has obvious advantages over the two baselines for non-i.i.d distributions. This is because the CNN model can easily achieve a good performance on the MNIST dataset, especially under i.i.d case. This implies that the model has little dependence on the size of data. But in the case of non-i.i.d distributions or more complicated models, data sizes play a more critical role in speeding up the model convergence. Thus, our algorithm can show more obvious advantages.



(a) i.i.d distribution. (b) Non-i.i.d distribution.

Fig. 3: Impact of offloading on MNIST (CNN).

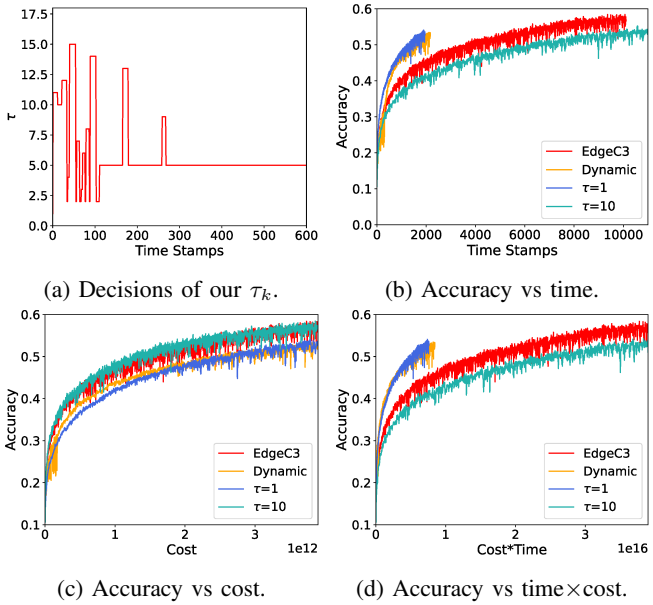
2) Advantages of model aggregation frequency: Fig. 4 and Fig. 5 exhibit the empirical superiority of our approach over other approaches. Fig. 4(a) and Fig. 5(a) show that the UCB algorithm largely explores different choices of τ and makes fewer mistakes when time goes on. At the beginning, the algorithm explores through trials and errors, and the results of $\tau = 3$ and $\tau = 5$ have been learned under the MNIST and CIFAR10 datasets, respectively. Our baselines either consume the budget too aggressively because of a high aggregation frequency, or it cannot converge within the same time as ours because of setting the aggregation frequency to be too low.



(a) Decisions of our τ_k. (b) Accuracy vs time.

(c) Accuracy vs cost. (d) Accuracy vs time×cost.

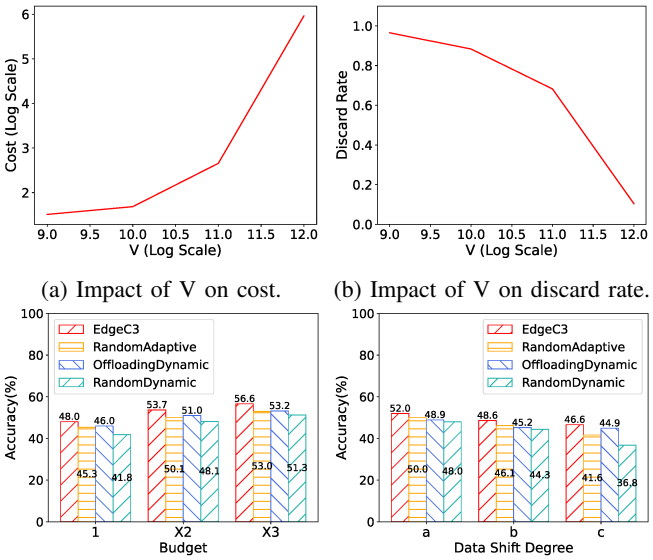
Fig. 4: Evaluating choices of τ under MNIST.



(a) Decisions of our τ_k. (b) Accuracy vs time.

(c) Accuracy vs cost. (d) Accuracy vs time×cost.

Fig. 5: Evaluating choices of τ under CIFAR10.



(a) Impact of V on cost. (b) Impact of V on discard rate.

(c) Impact of budget. (d) Impact of data shift degree.

Fig. 6: Impact of various configurations.

Although **Dynamic** can also adjust the τ according to the cost, it is not designed for continuous learning under uncertain network dynamics and the long-term budget. Our method can strike a better balance between time and cost budget to achieve higher accuracy. Fig. 4(c) and Fig. 5(c) show the achieved accuracy per unit of cost. In Fig. 5(c), EdgeC3 (Adaptive) may seem inferior to using a larger τ. However, under the same cost, using $\tau = 10$ takes a longer period of time than our online algorithm, and Fig. 4(d) and Fig. 5(d) verifies our advantage in terms of accuracy for each unit of time×cost.

3) Impact of V and data shifts: Fig. 6(a) and Fig. 6(b) show that a larger V (see (15)) leads to a higher accuracy

but a larger cost since more data points are needed to be processed; and thus the discard rate will drop. Fig. 6(c) demonstrates the impact of different cost budgets and the labels "X2" and "X3" represent doubled and tripled cost budgets respectively, under which the accuracy increases by 7.2% to 9.5%, and EdgeC3 can achieve higher accuracy over the baselines. Finally, Fig. 6(d) shows the accuracy under different data shift degrees with the same cost budget. With the increase of data shift, the accuracy decreases; and EdgeC3 improves the accuracy by 6.41% againt **OffloadingDynamic**, 12.3% against **RandomAdaptive**, and 14.9% against **RandomDynamic**, respectively. This emphasizes the value of co-optimized aggregation frequency and dynamic offloading.

VII. Concluding Remarks and Future Work

In this paper, we propose EdgeC3 that jointly optimizes the frequency of model synchronization and dynamic data offloading, navigating the trade-off between model accuracy and the long-term cost. We derive an upper-bound of the training error with respect to adjustable aggregation frequencies and data shifts. To learn the best aggregation frequency under uncertain future offloading decisions, we integrate online learning into an extended Lyapunov Optimization algorithm. We theoretically and experimentally demonstrate the efficacy of our method. For future work, we will further integrate data selection to enhance its robustness in edge ML systems.

References

[1] M. Ranzato, G. E. Hinton, and Y. LeCun, "Guest editorial: Deep learning," *Int. J. Comput. Vis.*, vol. 113, no. 1, pp. 1–2, 2015.

[2] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surv. Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[3] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A survey on distributed machine learning," *CoRR*, vol. abs/1912.09789, 2019. [Online]. Available: http://arxiv.org/abs/1912.09789

[4] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 44:1–44:37, 2014.

[5] S. Wang, S. Hosseinalipour, V. Aggarwal, C. G. Brinton, D. J. Love, W. Su, and M. Chiang, "Towards cooperative federated learning over heterogeneous edge/fog networks," 2023. [Online]. Available: https://arxiv.org/abs/2303.08361

[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, ser. Proceedings of Machine Learning Research, A. Singh and X. J. Zhu, Eds., vol. 54. PMLR, 2017, pp. 1273–1282. [Online]. Available: http://proceedings.mlr.press/v54/mcmahan17a.html

[7] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.

[8] H. T. Nguyen, V. Sehwag, S. Hosseinalipour, C. G. Brinton, M. Chiang, and H. V. Poor, "Fast-convergent federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 201–218, 2020.

[9] S. Hosseinalipour, S. S. Azam, C. G. Brinton, N. Michelusi, V. Aggarwal, D. J. Love, and H. Dai, "Multi-stage hybrid federated learning over large-scale d2d-enabled fog networks," *IEEE/ACM Transactions on Networking*, 2022.

[10] S. Wang, Y. Ruan, Y. Tu, S. Wagle, C. G. Brinton, and C. Joe-Wong, "Network-aware optimization of distributed learning for fog computing," *IEEE/ACM Trans. Netw.*, vol. 29, no. 5, pp. 2019–2032, 2021.

[11] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *IEEE INFOCOM*. IEEE, 2018, pp. 63–71.

[12] ——, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, 2019.

[13] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *2017 IEEE 37th international conference on distributed computing systems (ICDCS)*. IEEE, 2017, pp. 328–339.

[14] C. Hu, W. Bao, D. Wang, and F. Liu, "Dynamic adaptive dnn surgery for inference acceleration on the edge," in *IEEE INFOCOM*. IEEE, 2019, pp. 1423–1431.

[15] L. Liu, M. Zhao, M. Yu, M. A. Jan, D. Lan, and A. Taherkordi, "Mobility-aware multi-hop task offloading for autonomous driving in vehicular edge computing and networks," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[16] I. Alghamdi, C. Anagnostopoulos, and D. P. Pezaros, "Data quality-aware task offloading in mobile edge computing: an optimal stopping theory approach," *Future Generation Computer Systems*, vol. 117, pp. 462–479, 2021.

[17] F. Farahbakhsh, A. Shahidinejad, and M. Ghobaei-Arani, "Multiuser context-aware computation offloading in mobile edge computing based on bayesian learning automata," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, p. e4127, 2021.

[18] N. Le Roux, M. Schmidt, and F. Bach, "A stochastic gradient method with an exponential convergence rate for finite training sets," *Pereira et al*, 2013.

[19] S. Shalev-Shwartz and T. Zhang, "Stochastic dual coordinate ascent methods for regularized loss minimization," *arXiv preprint arXiv:1209.1873*, 2012.

[20] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," *Advances in neural information processing systems*, vol. 26, 2013.

[21] A. Defazio, F. Bach, and S. Lacoste-Julien, "Saga: A fast incremental gradient method with support for non-strongly convex composite objectives," *Advances in neural information processing systems*, vol. 27, 2014.

[22] B. Veloso, J. Gama, and B. Malheiro, "Self hyper-parameter tuning for data streams," in *International Conference on Discovery Science*. Springer, 2018, pp. 241–255.

[23] Z. Zhou, S. Yang, L. Pu, and S. Yu, "Cefl: Online admission control, data scheduling, and accuracy tuning for cost-efficient federated learning across edge nodes," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9341–9356, 2020.

[24] Y. Jin, L. Jiao, Z. Qian, S. Zhang, and S. Lu, "Budget-aware online control of edge federated learning on streaming data with stochastic inputs," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3704–3722, 2021.

[25] ——, "Learning for learning: Predictive online control of federated learning with edge provisioning," in *IEEE INFOCOM, Vancouver, BC, Canada, May 10-13, 2021*. IEEE, 2021, pp. 1–10.

[26] X. Zhang, J. Wang, G. Joshi, and C. Joe-Wong, "Machine learning on volatile instances," 2020.

[27] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[28] "Edgec3: Online management for edge-cloud collaborative continuous learning," Tech. Rep. [Online]. Available: https://www.dropbox.com/s/x5l7w8i9x1qi4d3/EdgeC3.pdf?dl=0

[29] E. Hazan *et al.*, "Introduction to online convex optimization," *Foundations and Trends® in Optimization*, vol. 2, no. 3-4, pp. 157–325, 2016.

[30] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, pp. 235–256, 2002.

[31] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[32] D. Raca, J. J. Quinlan, A. H. Zahran, and C. J. Sreenan, "Beyond throughput: A 4g lte dataset with channel and context metrics," in *Proceedings of the 9th ACM multimedia systems conference*, 2018, pp. 460–465.