Chapter 3
Heterogeneity Aware Distributed Machine Learning at the Wireless Edge for Health IoT Applications: An EEG Data Case Study

Umair Mohammad and Fahad Saeed

3.1 Introduction and Motivation

The culminated amount of electronic data generated each day is equal to three hundred billion gigabytes. A significant chunk of this data comes from user equipment (UE) including but not limited to smartphones, wellness devices, traffic cameras, autonomous connected vehicles, unmanned aerial vehicles (UAVs), and augmented reality equipment – collectively categorized under the umbrella of internet of things (IoT). Increasingly, to enable effective services, health devices – including clinical and wellness devices – form part of the IoT networks which are expected to exceed 25 billion devices connected to the internet by 2030 [1]. Forecasts by Statista show that the industrial IoT market which was worth $285 billion USD in 2017 will almost double in 5 years to $540 billion USD. Figure 3.1 charts this exponential growth and provides statistics on how various industries grew over a 5-year period from 2017 to 2022. The healthcare devices market share will reach $200 billion USD as shown in Fig. 3.1.

With increasing number of devices, it is estimated that more than 73.1 zettabytes ($\sim 10^{21}$) of data will be generated from these devices [1] by 2025. To enable useful scientific, clinical, or commercial value, much of the data analytics depend on machine learning (ML) techniques. These techniques include traditional ML such as regression (linear and logistic), support vector machines (SVM), neural networks (NN) and deep learning (DL) including the deep NN (DNN), convolutional NN (CNN), the residual neural net (ResNet), and Transformers, among others. Most of these techniques solve an underlying unconstrained optimization problem using iterative approaches such as the gradient descent (GD). Because these

U. Mohammad · F. Saeed (✉)
Knight Foundation School of Computing and Information Sciences, Florida International University, Miami, FL, USA
e-mail: umohamma@fiu.edu; fsaeed@fiu.edu

M. H. Amini (ed.), *Distributed Machine Learning and Computing*, Big and Integrated Artificial Intelligence 2, https://doi.org/10.1007/978-3-031-57567-9_3

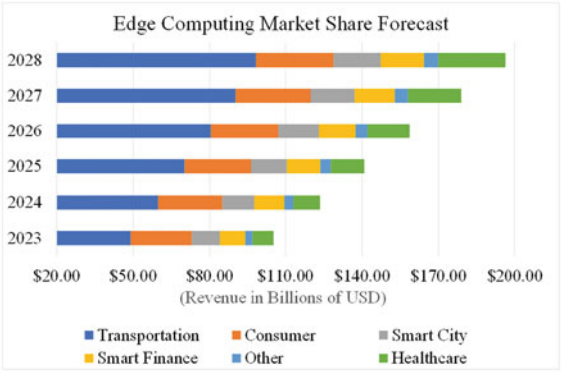


Fig. 3.1 Edge computing market share forecast of various industries from 2023 to 2028. (Data was taken from [3] for healthcare and [4] for other applications). As clearly observable, two of the largest industries include smart transportation and consumer with healthcare a distant fifth. However, over 5 years, healthcare clearly takes over other sectors to become the joint second largest revenue generator

iterative approaches can be computationally exhaustive for single processor devices, distributed machine learning (DML) approaches have been used. Mostly, offline training with DML is done using performance computing (HPC) with clusters of graphical processing units (GPUs) and CPUs. Even for inference at the edge, cloud-based approaches are used where the collected data from multiple nodes is transferred to a central cloud for inference or re-training. However, due to the load on backhaul links and rising privacy concerns, the trend is moving toward inference at the edge along with training and model updates. This has led to the development of the paradigm of federated learning (FL) [2] which supports the cooperative training of ML models at the edge without the involvement of cloud servers.

The goal of this chapter is to promote a generic framework for centrally optimized DML at the edge called *mobile edge learning* (MEL) with a focus on healthcare applications. The MEL paradigm fully encompasses FL and is also alternatively known in the literature as "Edge Artificial Intelligence" (Edge AI) or "federated learning (FL) at the resource-constrained edge." MEL supports both parallelized learning (PL) and FL which separates it from other approaches. In PL, edge devices (hereafter: "learners") distribute their data to trusted devices within the edge network to update the ML model for a specific task. In contrast, each learner owns their own dataset in FL which is more suitable for similar applications but under different conditions and requirements.

While edge analytics and edge AI (or edge intelligence, EI for short) may not completely replace cloud, it has significantly changed how intelligent IoT systems operate [5]. Advantages of EI include reduced communication costs, low storage requirements as most data can be discarded, and enhanced privacy and security

for the customers. It is expected that MEL will play a critical role in providing some of these features. Though there are several applications such as self-driving vehicles [6] and smart cities [7], one of the most interesting applications is health-IoT (H-IoT) or the internet of medical things (IoMT) [8]. Recent works cite FL as having a key role to play in IoMT/H-IoT analytics [9, 10]. There are several applications where real-time decision-making can either significantly enhance the quality of life or save lives. Model training with MEL can significantly improve the performance of such technologies. For example, MEL can be useful when predicting cardiac events by applying ML to data from a wearable electrocardiogram (ECG) or for seizure detection or prediction from a wearable electroencephalograph (EEG). While techniques presented in the next sections apply to MEL broadly, we will refer to H-IoT applications wherever applicable.

The rest of the chapter is organized as follows. We begin by introducing the general MEL model in Sect. 3.2. Section 3.3 then discusses synchronous MEL with only timeliness requirements. Section 3.4 introduces additional energy consumption limits. Detailed implementation details follow in Sect. 3.5 followed by results and discussions in Sect. 3.6. Lastly, Sect. 3.7 expands more on the H-IoT applications and provides a mathematical roadmap on how MEL can be tailored for H-IoT.

3.2 System Model and Parameters

Machine learning is the ability of machines (computers) to make decisions using prior data without being explicitly programmed. ML can be supervised when labeled data is available, or unsupervised in the absence of the data labels. In either case, the ML model must be trained. For most ML techniques, the model parameters are updated using an iterative optimization procedure based on a predefined loss/cost function. In supervised learning, the trained model is then evaluated in the validation/testing phase by generating an output without training on the validation/test dataset. The final output can be continuous or discrete depending upon the type of the task.

Let us consider a dataset $\mathcal{D}$ comprising a total of d samples. Each data sample $\mathcal{D}_n$ for $n = 1, \ldots, d$ that has $\mathcal{F}$ features is represented by a feature that can be denoted by x_j where $j = 1, \ldots, \mathcal{F}$. The set of features belonging to data sample number n can be denoted by $\mathbf{x}_n = \{x_1, \ldots, x_j, \ldots, x_{\mathcal{F}}\}$. These features serve as the ML model input and there may be a predefined output or target given by y_n. The objective is to find a set of model parameters $\mathbf{w}$ that minimize a loss function $F(\mathbf{x}_n, \mathbf{y}_n, \mathbf{w})$. If we represent the loss as $F_n(\mathbf{w})$ for short because $\mathbf{x}_n$ and y_n are known, the total loss is given by:

$$F(\mathbf{w}) = \frac{1}{d} \sum_{n=1}^{d} F_n(\mathbf{w}) \tag{3.1}$$

This optimization is typically done using iterative approaches such as the gradient descent (GD) because an analytical solution is not available. Then, at any discrete time-step l, for $l = 1, \ldots, L$, the model is updated as follows:

$$\mathbf{w}[l] = \mathbf{w}[l-1] - \eta \nabla F\left(\mathbf{w}[l-1]\right) \tag{3.2}$$

The learning rate represented by η is usually set on the interval (0, 1) and influences the convergence rate and the final accuracy. This process is iteratively applied sample by sample or batch by batch as in stochastic GD (SGD) [11] until the whole dataset is covered. Multiple epochs are performed until a stopping criterion is reached.

3.2.1 General Distributed Machine Learning

Many ML and DL techniques, including regression, support vector machine (SVM), and neural networks (NN), are built on iterative gradient-based learning. Distributed ML (DML) has been proposed to reduce the load on one processor due to intense compute requirements of such iterative approaches. One approach with data parallelism (DP) is where a central node distributes a large dataset to multiple other nodes to train local models. The central node maintains a global model and performs frequent global updates to maintain an optimal global model. Assume there is a central parameter server (referred to as the orchestrator hereafter) that initiates the DML process on a set of $\mathcal{K} = \{1, \ldots, k, \ldots, K\}$ learners. Each learner k updates the local model using a batch of the data $\mathcal{D}_k$ of size d_k (which may be locally owned or received by the orchestrator). After initializing a global model and from then on, after every global cycle, the orchestrator will send the global model $\mathbf{w}$ and possibly d_k samples to each learner $k \in \mathcal{K}$ which performs multiple τ_k updates of the local model $\mathbf{w}_k$ in parallel. Then, the orchestrator collects all local models for global aggregation. One such cycle can be called the global update cycle. Figure 3.3 illustrates one such cycle. These cycles repeat until the orchestrator reaches a stopping criteria such as a achieving desired validation loss, depletion of resources, or exceeding the time limit dedicated to the learning task.

$$\mathbf{w}_k[l] = \mathbf{w}_k[l-1] - \eta \nabla F_k(\mathbf{w}_k[l-1]) \tag{3.3}$$

The local model parameter set at learner k is given by $\mathbf{w}_k$, the local loss is given by $F_k(.)$, η is the learning rate, and the gradient operation is denoted by ∇. The local loss $F_k \; \forall \; k \in \mathcal{K}$ can be calculated using the local dataset $\mathcal{D}_k$ of size d_k in the following way [12]:

$$F_k(\mathbf{w}_k) = \frac{1}{d_k} \sum_{n=1}^{d_k} F_n(\mathbf{w}_k) \tag{3.4}$$

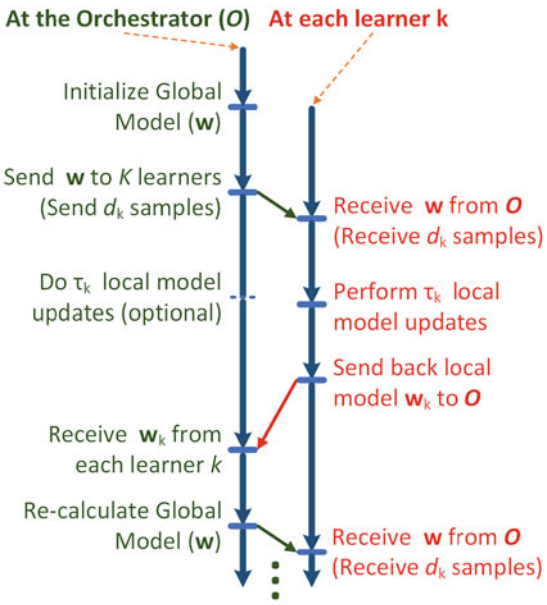


Fig. 3.2 Illustration of the DML process with DP and synchronous global updates

The global model can be obtained by aggregating the local models by applying an aggregation mechanism. One such method is the following [12]:

$$\mathbf{w} = \frac{1}{d} \sum_{k=1}^{K} d_k \mathbf{w}_k \tag{3.5}$$

If the local updates are synchronized, then the global aggregation occurs after $\tau_k = \tau \ \forall k \in \mathcal{K}$ time-steps, whereas in the asynchronous case, τ_k may differ for each learner. The orchestrator may perform multiple global cycles until a stopping criterion is reached (e.g., good performance or resource depletion). This process is summarized in Fig. 3.2.

3.2.2 Transition to Wireless MEL

Let us transition the above described DML system to wireless MEL where the learners have heterogeneous communication capabilities. For example, the devices can range from power laptops to smartphones with medium-sized processors to smart watches with limited capabilities. Consider an MEL system with a set of learners $\mathcal{K} = \{1, \ldots, k, \ldots, K\}$ as depicted in Fig. 3.3. Each learner $k \in \mathcal{K}$ updates the local model τ_k times on a batch of size d_k in every global update. The data/model communication occurs on wireless channel defined by gain h_{kO}

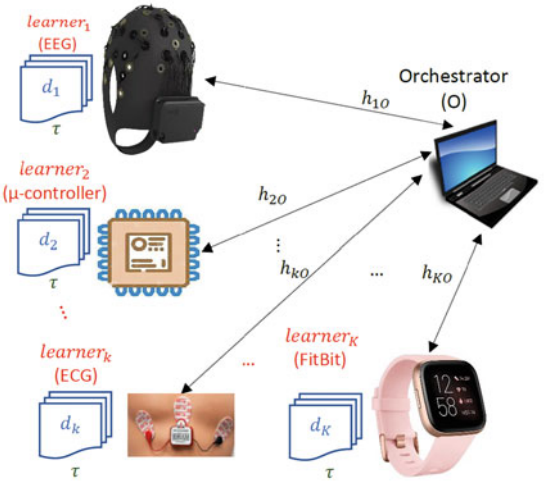


Fig. 3.3 System model of an MEL setting. Learners can be of different types including a wireless wearable EEG (e.g., learner 1), a microcontroller (e.g., learner 2), wearable ECG (learner 3), etc. Each learner performs τ_k local model updates in every global cycle on a dataset of size $d_k \; \forall \; k \in \mathcal{K}$. It has a local processor with clock speed f_k and a channel to the orchestrator represented by gain $h_{kO} \; \forall \; k \in \mathcal{K}$. Both respectively define the computation and communication capability of each learner $k \in \mathcal{K}$ which are heterogeneous

which can drastically vary spatially (among devices) and temporally (over time across multiple global updates). In contrast, in DML with HPC or wired nodes, resources are more homogeneous and less limited such that $\tau_k = 1$ which converges to the centralized case. To quantify this heterogeneity, we will relate the communication/computation parameters as well as the ML model specifics to each learner's time and energy consumption.

Let us recall the two closely related but distinct scenarios for MEL: federated learning (FL) and parallelized learning (PL). In FL, the learner generates its own data but cannot transfer it to a central cloud or other peers due to certain constraints (mainly privacy but also potentially bandwidth limitation [13]). On the other hand, the PL scenario usually involves a main node, which may be the edge server or one of the end devices. This edge server/end device parallelizes the learning process over its local dataset on multiple cores/nodes due to one or more reasons (e.g., limited main node resources, faster processing, lower energy consumption) [14]. In either case, the learning process cycles between a central orchestrator distributing the global model at the beginning of each cycle, learners updating the local model on their individual datasets, the orchestrator collecting the local models, and finally, performing the global aggregation until a stopping criterion is reached. The only difference is that in PL, the orchestrator also distributes the data subsets along with the global model at the start of each global cycle. Figure 3.4 illustrates the differences between both approaches. Clearly, PL fully encompasses FL, which will be reflected in the mathematical derivations later, but only adds to it the batch

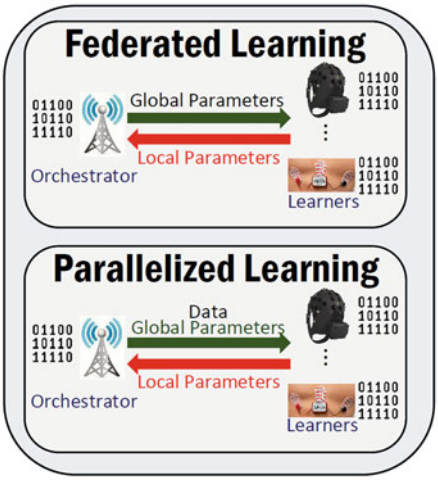


Fig. 3.4 Illustration of the differences between FL and PL, both paradigms that are part of MEL

transfer component from the orchestrator to the learners. We will therefore mainly formulate the problems with respect to PL but indicate the minor changes applicable to FL whenever necessary.

The goal is to minimize the local loss function [15]. The total size of all batches is denoted by $d = \sum_{k=1}^{K} d_k$, which is usually preset by the orchestrator O given its computational capabilities, the desired accuracy, and the time/energy constraints of the training/learning process. The number of local iterations or local updates run by learners on their allocated batch is denoted by τ_k. For the synchronous case, $\tau_k = \tau \; \forall \; k \in \mathcal{K}$, whereas in asynchronous task allocation, each learner can perform a different number of τ_k local iterations.

In addition to the two approaches, two key aspects of the MEL process are the expected task completion time and the energy consumption per learner $k \in \mathcal{K}$. In this chapter, we will consider both time and energy constraints in our model since both constraints are useful in the context of health and wellness IoT. There are multiple variables that can impact the time and energy consumption in MEL, whereas some of these may also impact the accuracy. For example, the number of local updates will directly impact the execution time and the dataset size will impact both the execution time and the transmission time. If MP is applied, dataset size may impact the completion time in both FL and PL, whereas with DP, it will only impact in PL. A smaller batch size may allow for more local updates which may improve accuracy because typically, in SGD, the loss decreases as the number of iterations are increased. However, if the dataset size is too small, that may also adversely affect the accuracy. Other variables that may impact the local completion time and energy consumption include the transmission power, local computational power, and the complexity of the ML model. Whereas these components may not be of significant influence when DL is executed over controlled wired and infrastructural servers, their high heterogeneity can tremendously impact the performance of DL when applied in wireless and mobile edge environments.

As in health and wellness devices specifically that require some sort of mitigation steps, the orchestrator will demand the results within preset duration within which all of these four steps should be completed. In previous works, it has been assumed that only devices that are charging or fully charged will take part in MEL. However, for most health devices that require real-time continuous monitoring, these devices or learners in MEL may not be fully charged or on direct power and may have a limit on the amount of battery power they are willing to drain. To this end, in the following two subsections, we define the time taken and the energy consumed by one learner $k \in \mathcal{K}$, respectively, to complete the MEL process.

3.2.2.1 Relationship to Completion Time and Energy Consumption

In the following paragraphs, we will relate these parameters for user k to both its local time and energy consumption for one global cycle. The orchestrator performs the aggregation of the parameters only once after all learners send back their result within the global update cycle after doing $\tau_k \ \forall \ k$ local updates. To summarize, the global update process in MEL occurs in periodic cycles that we will refer to as the global update cycles. This process should include the following phases:

1. The orchestrator transmits global model $\mathbf{w}$ and dataset of size d_k to each learner $k \in \mathcal{K}$ (in the common FL, only $\mathbf{w}$ is transmitted).
2. Each learner k computes τ_k local model updates.
3. Each learner returns the local model $\mathbf{w}_k \ \forall \ k \in \mathcal{K}$ to the orchestrator.
4. The orchestrator performs global aggregation is defined in (3.5).

Consider that d_k data samples can be expressed in B_k^{data} bits as follows:

$$B_k^{data} = d_k \mathcal{F} \mathcal{P}_d \tag{3.6}$$

where $\mathcal{F}$ is the feature vector size and $\mathcal{P}_d$ is a factor that accounts for the precision and compression ratio. The size of the local model $\mathbf{w}_k \ \forall \ k$ in bits is denoted by B_k^{model} can be expressed as:

$$B_k^{model} = \mathcal{P}_m \left(d_k S_d + S_m \right) \tag{3.7}$$

where $\mathcal{P}_m$ is the model bit precision or compression ratio. This size consists of two parts, the constant part specific to the model architecture described by S_m and a dynamic part S_d dependent upon the dataset size which can be used to support model parallelism (MP). Please note that the aggregation mechanism described in (3.5) cannot be employed with MP.

At the start of each global cycle, the orchestrator sends the optimal global model of size B_k^{model} bits and dataset of size B_k^{data} bits (in FL, $B_k^{data} = 0$). We assume communication occurs over ideal binary symmetric orthogonal channels without interference with channel bandwidth W, gain h_{kO}, and noise spectral density N_0.

Assuming a transmission power of P_{kO}, the time $t_k^S \; \forall \; k \; \in \; \mathcal{K}$ taken to complete the first step can be given by:

$$t_k^S = \frac{d_k \mathcal{F} \mathcal{P}_d + \mathcal{P}_m \left(d_k \mathcal{S}_d + \mathcal{S}_m\right)}{W \log_2 \left(1 + \frac{P_{ko} h_{ko}}{N_0}\right)} \tag{3.8}$$

The total computations X_k required for one local update by each learner $k \in \mathcal{K}$ is given by:

$$X_k = d_k C_m \tag{3.9}$$

where C_m is the computational complexity of the model. The time $t_k^C \; \forall \; k \; \in \; \mathcal{K}$ needed to perform one local update is:

$$t_k^C = \frac{X_k}{f_k} = \frac{d_k C_m}{f_k} \tag{3.10}$$

where f_k is each learner k's local processor frequency dedicated to the DL task. The time for the second step will be $\tau_k \times t_k^C \; \forall \; k \in \mathcal{K}$. Next, the time taken for the third step $t_k^R \; \forall \; k \in \mathcal{K}$ to send the updated local model to the orchestrator can be described as:

$$t_k^R = \frac{\mathcal{P}_m \left(d_k \mathcal{S}_d + \mathcal{S}_m\right)}{W \log_2 \left(1 + \frac{P_{ko} h_{ko}}{N_0}\right)} \tag{3.11}$$

The time for the last stage is negligible compared to the first three stages due to it being simple aggregation and use of efficient over-the-air approaches. Thus, the global cycle time, which is the total time $t_k \; \forall \; k \in \mathcal{K}$ taken by learner k to complete the first three processes of MEL, is equal to:

$$\begin{aligned} t_k &= t_k^S + \tau_k t_k^C + t_k^R \\ &= \frac{d_k \mathcal{F} \mathcal{P}_d + 2\mathcal{P}_m \left(d_k \mathcal{S}_d + \mathcal{S}_m\right)}{W \log_2 \left(1 + \frac{P_{ko} h_{ko}}{N_0}\right)} + \tau_k \frac{d_k C_m}{f_k} \end{aligned} \tag{3.12}$$

Given the above-described MEL model, each learner k consumes energy when performing the τ_k local model updates and when transmitting the local model $\mathbf{w}_k$ to the orchestrator. For the time being, we do not consider the energy consumed by the orchestrator because it will have a negligible impact on learner k, especially if the orchestrator is an edge server connected to a main supply. Given the processor speed of f_k in GHz, the energy $e_k^C \; \forall \; k \in \mathcal{K}$ consumed to perform one local update on a dataset of size d_k is given by [16]:

$$e_k^C = \mu X_k {f_k}^{\nu - 1} = \mu d_k C_m {f_k}^{\zeta - 1}, \; k \in \mathcal{K} \tag{3.13}$$

where μ is the onboard chip capacitance (typically $10^{-9} \sim 10^{-12}$ F) and $\nu = 2$ [16]. The energy $e_k^R \ \forall \ k \in \mathcal{K}$ consumed by learner k to send the most up-to-date local model is:

$$e_k^R = \frac{P_{kO} B_k^{model}}{R_k} = \frac{\mathcal{P}_m \left(d_k \mathcal{S}_d + \mathcal{S}_m\right)}{W \log_2 \left(1 + \frac{P_{ko} h_{ko}}{N_0}\right)}, \ k \in \mathcal{K} \tag{3.14}$$

The total energy $e_k \ \forall \ k \in \mathcal{K}$ consumed in one global update cycle can be given by:

$$\begin{aligned} e_k &= \tau_k e_k^C + e_k^R \\ &= \frac{P_{kO} \mathcal{P}_m \left(d_k \mathcal{S}_d + \mathcal{S}_m\right)}{W \log_2 \left(1 + \frac{P_{ko} h_{ko}}{N_0}\right)} + \tau_k d_k \mu C_m {f_k}^{\zeta - 1} \end{aligned} \tag{3.15}$$

3.2.3 Problem Formulation

It is clear that both t_k and $e_k \ \forall \ k \in \mathcal{K}$ depend upon a number of parameters including the number of local updates τ_k, the local dataset size d_k, transmission power P_{ko}, model size S_m, model computational complexity C_m, etc. Some of these will be preset by the orchestrator with DL model selection such as S_m and C_m, whereas others may depend upon the wireless communication protocol (P_{kO}, H_{kO}, W) or the device type (e.g., f_k). However, $\tau_k, d_k, P_{ko} \ \forall \ k \in \mathcal{K}$ can be optimized or controlled for best use of the resources. We limit our discussion to the optimization of these variables though other works [17–19] focus on optimizing the wireless communication. As discussed earlier, previous works have focused on optimizing τ_k's [12, 15, 20, 21].

However, the impact of dataset size $d_k \ \forall \ k \in \mathcal{K}$ allocation and the PL scenario have never been studied, which are covered in this chapter. To this end, we will study the joint impact of τ_k and d_k on resource consumption in the form of time and energy and try to optimize them such that it enhances ML model performance.

To make things more compact, we rewrite the expressions of $t_k \ \forall \ k \in \mathcal{K}$ in (3.12) as a function of τ_k and d_k as follows:

$$t_k = C_k^2 \tau_k d_k + C_k^1 d_k + C_k^0 \tag{3.16}$$

where C_k^2, C_k^1, and C_k^1 represent the quadratic, linear, and constant coefficients as follows:

$$C_k^2 = \frac{C_m}{f_k} \tag{3.17}$$

$$C_k^1 = \frac{\mathcal{F}\mathcal{P}_d + 2\mathcal{P}_m\mathcal{S}_d}{W \log_2\left(1 + \frac{P_{ko}h_{ko}}{N_0}\right)} \tag{3.18}$$

$$C_k^0 = \frac{2\mathcal{P}_m\mathcal{S}_m}{W \log_2\left(1 + \frac{P_{ko}h_{ko}}{N_0}\right)} \tag{3.19}$$

We can also rewrite the expression for $e_k \; \forall \; k \in \mathcal{K}$ in (3.15) as follows:

$$e_k = G_k^2 \tau_k d_k + G_k^1 d_k + G_k^0 \tag{3.20}$$

The quadratic, linear, and constant coefficients are denoted by G_k^2, G_k^1, and G_k^1, respectively, in the following way:

$$G_k^2 = \mu C_m {f_k}^{\zeta - 1} \tag{3.21}$$

$$G_k^1 = \frac{P_{k0}\mathcal{P}_m\mathcal{S}_d}{W \log_2\left(1 + \frac{P_{ko}h_{ko}}{N_0}\right)} \tag{3.22}$$

$$G_k^0 = \frac{P_{k0}\mathcal{P}_m\mathcal{S}_m}{W \log_2\left(1 + \frac{P_{ko}h_{ko}}{N_0}\right)} \tag{3.23}$$

It is clear that the expressions in (3.16) and (3.21) are quadratic with respect to τ_k and $d_k \forall \; k \in \mathcal{K}$. Though the significance of this will be outlined later, it is worth noting the complete heterogeneity aware (HA) MEL system model can be described by these equations. With respect to optimization, we can either go for the synchronous implementation $\tau_k = \tau \; \forall \; k \in \mathcal{K}$ (HA-Sync) or the semi-synchronous with different values for τ_k (HA-Asyn). Further, we can either consider only time constraints or dual-time and energy constraints. This gives rise to four possible scenarios:

1. HA-Sync with time constraints only
2. HA-Asyn with time constraints only
3. HA-Sync with dual-time and energy constraints
4. HA-Asyn with dual-time and energy constraints

In this chapter, we will only focus on HA-Sync (scenarios 1 and 3) and refer readers to [22] for more details on HA-Asyn. We will begin with HA-Sync with only time constraints and then move on to dual-time and energy constraints.

3.3 Synchronous MEL with Only Time Constraints

In this section, we discuss the first scenario where all learners perform a synchronous number of local updates in every global update.[1] In Sect. 3.2.3, we introduced the MEL system model parameters and how they relate to the time consumed in each global cycle by each learner $t_k \ \forall k \in \mathcal{K}$. In synchronous MEL, the orchestrator sets an upper limit T such that $t_k \leq T \ \forall k \in \mathcal{K}$ and the number of local updates are synchronized such that $\tau_k = \tau \ \forall k \in \mathcal{K}$.

3.3.1 Formulation

The goal is to minimize the loss of the MEL model. In general, the loss of GD-based optimization methods approaches a minimum as the learning iterations are increased. For synchronous DML, this is equivalent to maximizing the number of local updates τ given a fixed number of global updates as demonstrated by the following convergence proof.

Lemma 1 *Let* $\mathbf{w}[L]$ *denote the global model at update step* L *and* $\mathbf{w}^*$ *denote the optimal global model. For simplicity, assume that* τ *local updates are performed in each global update for a fixed number of global updates* G *such that* $L = G\tau$. *Then, the difference between the loss at update step* L *and the global optimal loss* $\left[F(\mathbf{w}[L]) - F(\mathbf{w}^*)\right] \rightarrow 0$ *as* $\tau \rightarrow \infty$.

This is a factual error as initially, the authors considered presenting both, the synchronous and asynchronous models. However, due to space limitations, the authors only presented the synchronous model. Unfortunately, while making the appropriate changes, the Lemma related to the synchronous version was removed erroneously whereas it is the Lemma related to the asynchronous version which should have actually been removed. In contrast, the proof related to Lemma of the synchronous version was retained in Appendix 1 while the proof related to the Lemma of the asynchronous version was correctly removed via the elimination of Appendix 4.

Thus, maximizing the MEL accuracy is achieved by maximizing τ. Therefore, the problem can be expressed as the following optimization program:

$$\max_{\tau, d_k \ \forall \ k \in \mathcal{K}} \quad \tau \tag{3.24}$$

$$\text{s.t.} \qquad C_k^2 \tau d_k + C_k^1 d_k + C_k^0 \leq T, \quad \forall k \in \mathcal{K} \tag{3.24a}$$

[1] This section is part of two papers: "Adaptive Task Allocation for Mobile Edge Learning" published in proceedings of the IEEE WCNCW 2019 [23] and "Dynamic Task Allocation for Mobile Edge Learning" published in IEEE Transactions on Mobile Computing.

$$\sum_{k=1}^{K} d_k = d \tag{3.24b}$$

$$\tau \in \mathcal{Z}_+ \tag{3.24c}$$

$$d_k \in \mathcal{Z}_+, \quad k \in \mathcal{K} \tag{3.24d}$$

Constraint (3.24a) guarantees that $t_k \leq T \; \forall \; k \in \mathcal{K}$. Constraint (3.24b) ensures that the whole dataset is covered across the set of all learners. Constraints (3.24c) and (3.24d) are simply nonnegative integer constraints on the optimization variables τ and d_k which, recall, are quadratically related in constraint (3.24a). Consequently, the problem can be expressed as a quadratically constrained integer linear program (QCILP). This formulation applies to both FL and PL, with the exception of the value of constant C_k^1 which does not impact the solution approach.

3.3.2 Solution

Because QCILP are NP-hard [24] in general and heuristic polynomial time approaches incur a high computational cost, we can at least relax the integer constraints in (3.24) and (3.24a). The relaxed problem is given by:

$$\max_{\tau, d_k \; \forall \; k \in \mathcal{K}} \quad \tau \tag{3.25a}$$

$$\text{s.t.} \quad C_k^2 \tau d_k + C_k^1 d_k + C_k^0 \leq T, \quad k \in \mathcal{K} \tag{3.25b}$$

$$\sum_{k=1}^{K} d_k = d \tag{3.25c}$$

$$\tau \geq 0 \tag{3.25d}$$

$$d_k \geq 0, \quad k \in \mathcal{K} \tag{3.25e}$$

The resulting program in (3.25) is quadratically constrained linear program (QCLP) which can be efficiently solved using research/commercially available solvers such as OPTI [25]. To obtain the suboptimal integer τ^* and $d_k^* \; \forall \; k \in \mathcal{K}$, we can floor the obtained real values and cases where either $\tau = 0$ or one more of $d_k = 0$ for any $k \in \mathcal{K}$ represent the infeasibility of MEL.

However, these solvers are still too computationally complex. Notice that the associated matrix of each quadratic constraint in (3.25b) is symmetric. These matrices will have one positive and one negative eigenvalue each which results in the problem being non-convex. This leads to the inefficiency and the inability to derive analytical solutions. However, we can derive a more efficient solution using

Lagrangian analysis and the Karush-Kuhn-Tucker (KKT) conditions to obtain upper bounds on the optimal optimization variables τ and $d_k \ \forall \ k \in \mathcal{K}$.

The Lagrangian function of (3.25) can be written as:

$$L\left(\mathbf{x}, \lambda, \nu, \alpha\right) = -\tau + \sum_{k=1}^{K} \lambda_k \left(C_k^2 \tau d_k + C_k^1 d_k + C_k^0 - T\right) + \nu \left(\sum_{k=1}^{K} d_k - d\right) - \alpha_0 \tau - \sum_{k=1}^{K} \alpha_k d_k \tag{3.26}$$

where $\lambda_k \ \forall \ k \in \mathcal{K}$, ν, and $\alpha_0/\alpha_k \ \forall \ k \in \mathcal{K}$ are the Lagrangian multipliers associated with constraints (3.25b), (3.25c), (3.25d), and (3.25e), respectively. Using the KKT conditions, Theorem 1 introduces bounds on τ^* and $d_k^* \ \forall \ k \in \mathcal{K}$.

Theorem 1 *The optimal value of the batch size $d_k^* \ \forall$ allocated to each learner $k \in \mathcal{K}$ satisfies the following bound:*

$$d_k^* \leq \frac{T - C_k^0}{\tau^* C_k^2 + C_k^1} \qquad \forall \ k \in \mathcal{K} \tag{3.27}$$

Further, the analytical upper bound on the optimization variable τ belongs to the solution set of the polynomial equation:

$$d \prod_{k=1}^{K} \left(\tau^* + b_k\right) - \sum_{k=1}^{K} a_k \prod_{\substack{l=1 \\ l \neq k}}^{K} \left(\tau^* + b_l\right) = 0 \tag{3.28}$$

where $r_k^0 = C_k^0 - T$, $a_k = -\dfrac{r_k^0}{C_k^2}$, and $b_k = \dfrac{C_k^1}{C_k^2}, \ \forall \ k \in \mathcal{K}$.

Proof Please refer to "Appendix 2" for the proof. □

3.4 Synchronous MEL with Dual-Time and Energy Constraints

The solutions in the previous section apply to MEL when optimal task allocation needs to be done only under time constraints. However, edge devices, especially UEs, are usually battery operated which means energy is a premium resource. Though most FL literature assumes only devices being charged or under direct power supply will participate in MEL, this may not always be feasible. Therefore, we study the same HA-Sync model but this time under dual-time and energy constraints.

3.4.1 Formulation

Recall the MEL model described in Sect. 3.3 where the time taken t_k and the energy consumed $e_k \ \forall \ k \in \mathcal{K}$ for one global update cycle is given by (3.16) and (3.21), respectively. We have established that local model updates per global update can improve validation performance. However, now the objective is to perform batch size (d_k) allocation such that the number of local updates τ per global update is maximized without violating the global cycle time T *and* for each learner $k \in \mathcal{K}$, without exceeding the local energy consumption limit defined as E_k^0 J of energy per global cycle. (Notice that for synchronous MEL, $\tau_k = \tau$, and the optimization variables are $\tau, d_k \ \forall \ k \in \mathcal{K}$.) We can rewrite the time and energy constraints as respectively shown in (3.29) and (3.30).

$$t_k = C_k^2 d_k \tau + C_k^1 d_k + C_k^0 \leq T \qquad \forall \ k \in \mathcal{K} \tag{3.29}$$

$$e_k = G_k^2 d_k \tau + G_k^1 d_k + G_k^0 \leq E_k^0 \qquad \forall \ k \in \mathcal{K} \tag{3.30}$$

The coefficients C_k^2, C_k^1, and C_k^0, related to the completion time t_k, and the coefficients G_k^2, G_k^1, and G_k^0, related to the energy consumption $e_k \ \forall \ k \in \mathcal{K}$, have been described in Section 3.2.3.

Since the relationship between the optimization variables is quadratic in both the time and energy constraints, and the variables are integers, the problem will still be an NP-hard QCILP. Once again, we can relax the integer constraints on the variables. While the problem still presents as a non-convex QCLP but allows for obtaining suboptimal solutions in polynomial time, the relaxed problem can be expressed as the following optimization program:

$$\max_{\tau, d_k \ \forall \ k} \quad \tau \tag{3.31}$$

$$\text{s.t.} \quad C_k^2 d_k \tau + C_k^1 d_k + C_k^0 \leq T, \quad \forall k \in \mathcal{K} \tag{3.31a}$$

$$G_k^2 d_k \tau + G_k^1 d_k + G_k^0 \leq E_k^0, \quad \forall k \in \mathcal{K} \tag{3.31b}$$

$$\sum_{k=1}^{K} d_k = d \tag{3.31c}$$

$$\tau \geq 0 \tag{3.31d}$$

$$d_k \geq d_l, \quad \forall k \in \mathcal{K} \tag{3.31e}$$

Constraints (3.31) and (3.31a) guarantee that the MEL process does not violate the limits on the global cycle time and local energy consumption, respectively. Constraint (3.31c) ensures the utilization of the complete dataset of size d.

Constraints (3.31d) and (3.31e) ensure that τ is nonnegative and the batch sizes d_k's are nonnegative integers $\forall k$. This ensures that some learners are not completely eliminated or have too few data samples which may also affect MEL performance.

3.4.2 Proposed Solution

The problem in (3.31) can be solved numerically using commercial solvers that may employ approaches such as interior point methods, branch and bound techniques, heuristics, etc. However, we propose a more efficient analytical-numerical solution based on a relaxation approach. Based on the suggest-and-improve (SAI) method [26], we will derive upper bounds on the optimal variables using Lagrangian analysis and then use a local optimizer (coordinate descent) to reach the optimal solution.

The equality constraint in (3.31c) can be written as the following two inequality constraints: $\sum_{k=1}^{K} d_k - d \leq 0$ and $-\sum_{k=1}^{K} d_k + d \leq 0$. In that case, the Lagrangian function of the relaxed problem is given by:

$$L\left(\mathbf{x}, \boldsymbol{\lambda}, \Gamma, \alpha, \bar{\alpha}, \omega, \boldsymbol{\nu}\right) = -\tau + \sum_{k=1}^{K} \lambda_k \left(C_k^2 \tau d_k + C_k^1 d_k + C_k^0 - T\right) + \sum_{k=1}^{K} \gamma_k \left(G_k^2 \tau d_k + G_k^1 d_k + G_k^0 - E_k^0\right) + \alpha \left(\sum_{k=1}^{K} d_k - d\right) - \bar{\alpha} \left(\sum_{k=1}^{K} d_k - d\right) - \omega\tau - \sum_{k=1}^{K} \nu_k d_k \tag{3.32}$$

The Lagrange multipliers associated with the global cycle time and local energy constraints are given by λ_k and γ_k, respectively, $\forall\, k \in \mathcal{K}$. The Lagrange multipliers related to the two total task size constraint inequalities are given by $\alpha/\bar{\alpha}$, and ω/ν_k $k \in \mathcal{K}$ are the Lagrangian multipliers associated with the nonnegative constraints of both sets of optimization variables τ and d_k, respectively.

Let us denote the set of optimization variables by $\mathbf{x} = [\tau\ d_1\ d_2\ \ldots\ d_k\ \ldots\ d_K]^T$ and the set of Lagrange multipliers by $\Gamma = [\boldsymbol{\lambda}, \Gamma, \alpha, \bar{\alpha}, \omega, \boldsymbol{\nu}]^T$, where $\lambda = [\lambda_1 \ldots \lambda_k \ldots \lambda_K]^T$, $\gamma = [\gamma_1 \ldots \gamma_k \ldots \gamma_K]^T$, and $\nu = [\nu_1 \ldots \nu_k \ldots \nu_K]^T$.

Theorem 2 *The set of optimal Lagrange multipliers Γ^* can be obtained by solving the dual problem in the following semi-definite program (SDP):*

$$\max_{\Gamma} \quad \zeta \tag{3.33}$$

$$s.t. \quad \begin{bmatrix} \mathbf{F}^2(\Gamma) & \frac{1}{2}\mathbf{f}^1(\Gamma) \\ \frac{1}{2}\mathbf{f}^1(\Gamma) & f_0(\Gamma) - \zeta \end{bmatrix} \succcurlyeq 0$$

$$\Gamma \succcurlyeq 0$$

The functions of the Lagrange multipliers $\mathbf{F}^2(\Gamma)$, $\mathbf{f}^1(\Gamma)$, *and* $f_0(\Gamma)$ *are defined in the proof.*

Proof Please refer to "Appendix 3" for the proof. □

A candidate solution is given by:

$$\hat{\mathbf{x}} = -\frac{1}{4}\mathbf{F}^2(\Gamma)^{-1}\mathbf{f}^1(\Gamma) \tag{3.34}$$

In the case of a convex QCQP, the resulting solution will be optimal with zero duality gap, i.e., $\hat{\mathbf{x}} = \mathbf{x}$. In our case, because of the problem being non-convex, we have to use a simple local optimizer called the coordinate descent method to improve the candidate solution where the optimal solution $\mathbf{x}^*$ is given by [26]:

$$\mathbf{x}^* = \text{coordinate-descent}(\hat{\mathbf{x}}) \tag{3.35}$$

3.5 Heterogeneous Simulation Setup and MEL Algorithm

In this section, we will setup the simulation environment and describe the important parameters based on *real-world settings*. Unless otherwise specified, these parameters will be used throughout the next section to study the performance of MEL in terms of achievable local updates τ and ML model validation performance. Typically, a wireless MEL environment will include a set of end devices (learners) with heterogeneous capabilities connected via heterogeneous wireless communication links. We will first study the important parameters influencing both computation and communication, describe the modelling strategy, and quantify its effect on heterogeneity. Then we will present the complete simulation environment including the underlying ML model parameters. We will then demonstrate the superiority of the proposed HA schemes and illustrate how they can be employed in the real world with algorithmic steps.

3.5.1 Heterogeneity Analysis

The main variable that represents the computational capabilities of the learners is the processor clock speed $f_k \ \forall \ k \ \in \ \mathcal{K}$. After taking into account the speedups offered by modern multicore processing, we must consider the range of speeds offered by different devices such as laptops (6 GHz), mobile phones (2.4 GHz), various microcontroller types (0.5–1.5 GHz), etc. Therefore, the modeling strategy will include selecting a fixed lower end reference speed, to which we will add an additional amount to represent the variations in resources dedicated to the MEL

process by each learner $k \in \mathcal{K}$. At the start of each global cycle, the clock speed $f_k \; \forall \; k \in \mathcal{K}$ will be drawn from:

$$f_k \sim f_{ref} + f_\sigma U, \quad k \in \mathcal{K} \tag{3.36}$$

The reference clock is given by f_{ref}, whereas f_σ represents the maximum additional clock speed. $U \sim \mathcal{U}\{0, 1\}$ is the uniformly distributed random variable on $[0, 1]$ which ensures additional amount drawn from $[0, f_\sigma]$. To emulate different devices, we will use four references: $f_{ref,1} = 6000\,\text{MHz}$, $f_{ref,2} = 2000\,\text{MHz}$, $f_{ref,3} = 1000\,\text{MHz}$, and $f_{ref,2} = 500\,\text{MHz}$. The associated additional maximum amounts are $f_{\sigma,1}$, $f_{\sigma,2}$, $f_{\sigma,3}$, and $f_{\sigma,4}$, respectively.

The communication capability can be measured by the achievable rate $\rho_k \; \forall \; k \in \mathcal{K}$ which is defined as:

$$\rho_k = W \log_2 \left(1 + \frac{P_{ko} h_{ko}}{N_0}\right), \quad k \in \mathcal{K} \tag{3.37}$$

The rate is influenced by the bandwidth W, each learner's transmission power $P_{ko} \; \forall \; k$, and the channel gain $h_{ko} \; \forall \; k$. The gain is mainly determined by the path loss which is inversely proportional to the distance of the learner $R_{kO} \; \forall \; k$ from the orchestrator. We will simulate two different types of environments: Wi-Fi and cellular channels. Assuming a log-normal shadowing model with flat-fading, the gain h_{kO} can be calculated as the inverse of the signal attenuation as follows:

$$h_{kO} = \begin{cases} \left[\textbf{Linear}\left\{7 + 2.1 \log(\frac{R_{kO}}{R_0}) + \mathcal{N}\{0, 10\}\right\}\right]^{-1} & \text{Wi-Fi} \\ \left[\textbf{Linear}\left\{128.1 + 37.5 \log(\frac{R_{kO}}{R_0}) + \mathcal{N}\{0, 10\}\right\}\right]^{-1} & \text{Cellular} \end{cases} \tag{3.38}$$

Conversion from decibel scale to linear is represented by the **Linear** operation, R_{kO} is the distance of learner k to the orchestrator, and $\mathcal{N}$ is a zero-mean Gaussian random variable with standard deviation 10 dB. As we are not doing resource allocation (studied in other works [19, 27–29]), the bandwidth will be constant. Further, small-scale fading will be much lower compared to path loss and the impact of noise will be similar across multiple learners. Therefore, the most impactful variables on the rate heterogeneity will be the transmission power P_{kO} and the distance R_{kO}. Based on wireless communication standards, the maximum transmission power limit P_k^{max} is 23 dBm (or 0.1995 W in linear scale). We further assume that each learner-orchestrator pair use a lower power level per global cycle instead of the maximum allowed. Then, the transmission power $P_k \; \forall \; k \in \mathcal{K}$ is drawn from the following distribution:

$$P_k \sim P_k^{max} - P_\sigma U, \quad k \in \mathcal{K} \tag{3.39}$$

The variable P_σ represents the maximum value by which the power can be reduced and $U \sim \mathcal{U}\{0, 1\}$. To simulate the heterogeneous co-location of learners, the distances are simply drawn from $R_k \sim R_\sigma U$. R_σ is the maximum possible distance of the learner from the orchestrator. Note that the maximum radius of the environment R_0 can be different depending upon the environment.

Quantifying the heterogeneity exactly is difficult due to the complex nonlinear relations among the different important parameters. However, note that the maximum deviation among the most impactful variables f_k, P_{kO} and R_{kO} is bound by f_σ, P_σ, and R_σ, respectively. Hence, we define a new metric called the heterogeneity factor $\mathcal{H}_{fac}$ as follows:

$$\mathcal{H}_{fac} = \frac{P_\sigma}{P_k^{max}} + \frac{R_\sigma}{R_0} + \sum_{i=1}^{i=4} \frac{f_{\sigma,i}}{f_{ref,i}} \tag{3.40}$$

$\mathcal{H}_{fac}$ is the sum of ratios of the maximum deviations to the reference values. Taking the ratios accounts for the different units of measurements for the physical quantities making $\mathcal{H}_{fac}$ unitless. The heterogeneity factor is also not calibrated, (e.g., limited to range [0,1]) and a higher $\mathcal{H}_{fac}$ simply implies a more heterogeneous environment.

3.5.2 Simulation Environment

We use four different reference clock speeds to emulate the capacity of different types of devices such as laptops and roadside units ($f_{ref,1} = 6$ GHz), smartphones and tablets ($f_{ref,2} = 2$ GHz), commercial microcontrollers such as the Raspberry Pi ($f_{ref,3} =$ ‘GHz), and lower grade microcontrollers such as the Arduino ($f_{ref,4} = 0.5$ GHz). These may be attached to IoT devices such as traffic controllers, onboard units, cameras, industrial sensors, etc. We examine two different types of environments: 802.11-type environment (e.g., Wi-Fi) and a cellular-type environment. The maximum distance R_0 is set to 50 m for the former and 500m for the latter environment. To test the MEL with real-world ML tasks, we consider the classification task using two different datasets: the pedestrian [31] and MNIST [32] datasets. For both classification tasks, the measure of MEL performance is the validation accuracy. The pedestrian dataset has 8,000 training images from pedestrian crossing traffic light cameras. Each image has 684 features (18 × 36 pixels). The task is to predict whether a pedestrian is present in the image, a binary classification task for which we use a single-layer neural network with 300 neurons in the hidden layer. The set of model parameters includes two matrices $\mathbf{w} = [w_1, w_2]$, where w_1 is 300 × 648 and w_2 is 300 × 1. Thus, the model size B_k^{model} is 6,240,000 bits, whereas the forward and backward passes will require $C_m = 781{,}208$ floating point operations [33]. The MNIST dataset is also used for a classification task, but it has slightly larger images (28 × 28 pixels) leading to

Table 3.1 List of simulation parameters

Parameter	Value
Wi-Fi attenuation model	$7 + 2.1 \log(R)$ dB [30]
Cell attenuation model	$128 + 37.1 \log(R)$ dB [16]
Learner bandwidth (W)	5 MHz
Maximum distance indoor (R_0)	50 m
Maximum distance outdoor (R_0)	500 m
Max. tran. power (P_k^{max})	23 dBm
Noise power density (N_0)	-174 dBm/Hz
Reference clock speeds ($f_{ref,[1-4]}$)	{6, 2, 1, 0.5} GHz
Pedestrian dataset size (d)	9,000 images
Pedestrian dataset features ($\mathcal{F}$)	648 (18×36) pixels
MNIST dataset size (d)	60,000 images
MNIST dataset features ($\mathcal{F}$)	784 (28×28) pixels

$\mathcal{F} = 784$. A more complex three-layer DNN is used which has a size $B_k^{model} =$ 8,974,080 bits and complexity $C_m = 67{,}424{,}160$ floating point operations [33]. All of these parameters including the channel attenuation models are given in Table 3.1.

To perform the heterogeneity analysis, we vary P_σ from 0.01 to 0.10 in steps of 0.01 W and R_σ from 5 m to $R_0 = 50$ m in steps of 5 m for an 802.11-type environment. The clock speed heterogeneity is emulated by varying $f_{\sigma,1}$ from 60 to 600 MHz, $f_{\sigma,2}$ from 40 to 400 MHz, $f_{\sigma,3}$ from 30 to 300 MHz, and $f_{\sigma,4}$ from 20 to 200 MHz in steps of 60, 40, 30, and 20 MHz, respectively. For the remaining simulations, we fix R_σ equal to R_0 (i.e., 50 m for 802.11-type and 500 m for cellular-type environments), whereas P_σ is set to 0.05 W. The processor clock speed variations are emulated by setting $f_{\sigma,1} = 600$ MHz, $f_{\sigma,2} = 400$ MHz, $f_{\sigma,1} = 300$ MHz, and $f_{\sigma,2} = 200$ MHz. These parameters are selected to have a $\mathcal{H}_{fac} \in [1.45, 1.65]$ which represents a high heterogeneity level where the HA schemes provide noticeable gains. Moreover, this selection strategy truly shows the robustness of the HA schemes compared to HU because it supports more varying clock speeds, transmission powers, and channel gains.

3.5.3 MEL Algorithm

Though these are simulations, in the real world, it is assumed that the orchestrator and all K learners will exchange this information at the start of each global cycle for the orchestrator to run the centralized optimization. For example, each learner $k \in \mathcal{K}$ sends the information related to the clock speed f_k and transmission power P_{kO} it can dedicate for the next cycle. The communication time of parameter exchange is negligible compared to model/data transmission times. Further, channel gains and

Algorithm 1 Process at the Orchestrator

Input: $T, d, d_0, \mathcal{K}$
Output: $\mathbf{w}$

Initialize $\mathbf{w}$ and set the flag $STOP \leftarrow$ **FALSE**
while not *STOP* **do**
  In Parallel: Send $\mathbf{w}$ to each learner $k \in \mathcal{K}$
  In Parallel: Receive P_{kO}, h_{kO}, f_k, and e_k^0 from $k \in \mathcal{K}$
  if *Only Time Constraints* **then**
    Solve (3.28) to obtain τ^* and (3.27) to get d_k^*
  else
    Solve (3.34) to obtain the candidate $\hat{\tau}, \hat{d}_k$
    Use candidates to obtain τ^* and d_k^* by solving (3.35)
  end if
  In Parallel: SEND $\tau = \lfloor \tau^* \rfloor, d_k = \lfloor d_k^* \rfloor$ to each learner $k \in \mathcal{K}$
  if *PL* **then**
    In Parallel: SEND d_k data samples to each learner $k \in \mathcal{K}$
  end if
  WAIT for Ts to let each learner perform τ local updates
  In Parallel: RECEIVE $w_k \; \forall \; k \in \mathcal{K}$
  Obtain $\mathbf{w}$ using (3.5)
  if *STOPPING CRITERIA REACHED* **then**
    Set $STOP \leftarrow$ **TRUE**
  end if
end while
return $\mathbf{w}$

distances are known to the orchestrator via standard wireless channel estimation and triangulation algorithms. The complete steps followed by the orchestrator over multiple global cycles are summarized in Algorithm 1.

3.6 Results and Discussions

We plot the achievable number of local updates τ versus the heterogeneity factor $\mathcal{H}_{fac}$ and compare the performance of the HA and HU schemes. It can be seen that the HA-PL and HA-FL both offer a gain on the number of achievable local updates per global cycle as $\mathcal{H}_{fac}$ increases. This gain can be observed specifically for values of $\mathcal{H}_{fac} \geq 1$. On the other hand, the achievable τ remains constant for the HU schemes with increasing $\mathcal{H}_{fac}$ implying that these are truly HU. Our HA schemes are able to perform flexible local dataset size allocations per global cycle to maximize local updates. In the following subsections, we will show that for fixed numbers of global cycles, this also translates to higher final validation accuracies and lower convergence times. Figure 3.5 plots the achievable number of local updates τ versus the heterogeneity factor $\mathcal{H}_{fac}$ and compares the performance of the HA and HU schemes.

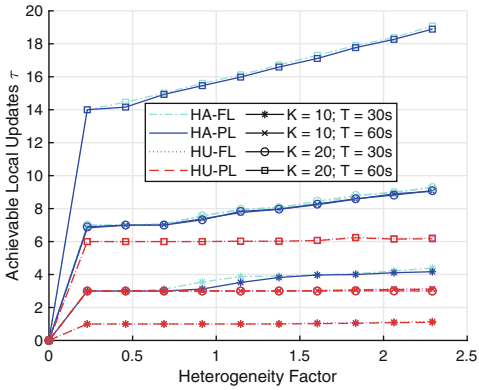


Fig. 3.5 Achievable local updates τ for the MNIST dataset versus the heterogeneity factor for different sets of learners and global cycle times

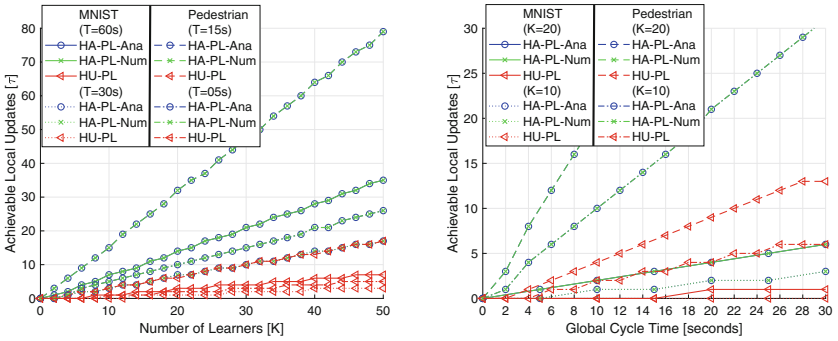


Fig. 3.6 Achievable local updates τ per global cycle for the MNIST and Pedestrian datasets for all schemes for (**a**) different values of T versus K and (**b**) for $K = 10$ and 20 versus T

3.6.1 Impact of Time Constraints on Local Model Updates

Figure 3.6 presents the simulation results for the PL scenario in an 802.11-type environment for both MNIST and pedestrian classification tasks. The top two subfigures demonstrate the achievable local updates τ when performing MNIST and pedestrian classification tasks with the above-described simulation parameters in Figs. 3.6a (versus increasing number of learners K) and 3.6b (versus increasing global cycle times T). We compare the results for τ for the HA scheme (HA-Sync) from the analytical upper bounds proposed in Theorem 1 (denoted by HP-PL-Ana) against numerical results from a commercially available solver (denoted by HA-PL-Num). As observable, there is no optimality gap as witnessed by how well the lines representing HA-PL-Ana stack up with HA-PL-Num. We also plot the achievable τ for the HU scheme (denoted by HU-PL).

In general, it can be observed that the HA-Sync approach allows for significantly more local updates τ compared to HU-Sync with gains in the region of 200% to 600%. For instance, Fig. 3.6a shows that both HA-PL-Ana and HA-PL-Num make it possible to perform $\tau = 6$ updates for MNIST classification with a system of $K = 20$ learners and $T = 30$ s, a gain of 600% against the HU-PL which allows only 1 local update. Further, HA-PL provides better performance with (less than) half the resources. For example, for a system of $K = 20$ learners, HU-PL performs worse with $T = 60$ s than HA-PL with $T = 30$ s for MNIST classification. Further, for pedestrian classification as shown in Fig. 3.6b, HA-Sync with $K = 10$ performs better than HU-PL with $K = 20$ for fixed global cycle time $T = 05$s with a gain of up to 500%. These performance gaps are reflected in the validation performance superiority of HA-Sync.

3.6.1.1 Improvements in Validation Accuracy

Figure 3.7 compares the progression of the validation accuracy after each global cycle for the MNIST and pedestrian datasets using both HA-Sync and HU-Sync schemes. Observably, HA-Sync offers a validation accuracy improvement up to 0.6% for the MNIST dataset and 8% for the pedestrian dataset. Furthermore, the HA schemes can offer up to 56% reduction in conversion times to certain accuracy thresholds. For example, performing MNIST classification with a system of $K = 10$ learners with a $T = 30$ s global cycle time constraint can be achieved with 96.50% accuracy in five global cycles with the proposed HA schemes in this section. In contrast, the HU schemes require nine global cycles which represent a reduction of 44% or 2 minutes. The designed HA scheme in this section can allow more local updates per global cycle which result in higher validation performance and lower convergence times.

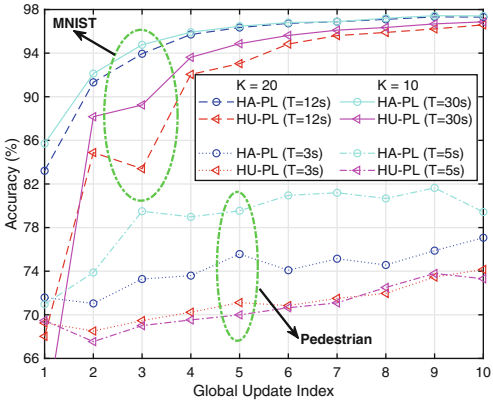


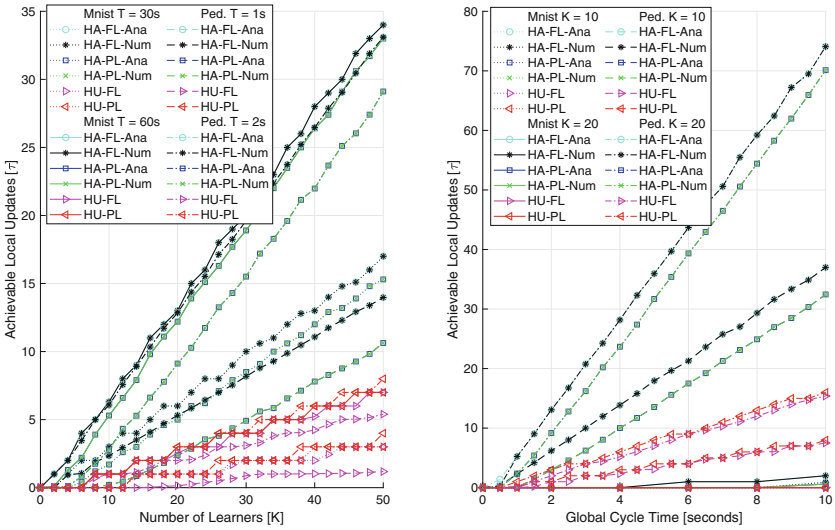
Fig. 3.7 Validation accuracy progression comparison between HA and HU for the MNIST dataset for up to 10 global cycles and for the pedestrian dataset up to 20 global cycles

3.6.2 Comparing FL versus PL

To really test the strength of the proposed HA schemes, we further evaluate them in a more challenging cellular type of environment for the same tasks with the all other simulation parameters being the same. In addition to HA-Sync with HA-PL-Ana/Num, we also compare against FL by adding results for HA-FL-Ana/Num as shown in Fig. 3.8. Once again, Figs. 3.8a,b present the achievable local updates for the MNIST and pedestrian classification tasks versus K and T, respectively. It is clear again that there is no optimal gap between the upper bounds in Theorem 1 and the numerical solutions from the solver.

In general, we observe that HA-FL/PL provide more achievable local updates compared to HU-FL/PL with gains ranging from 100% to 600%, and the FL approaches provide nominally more local updates than PL. Recall that the only difference between FL and PL is that there is an additional component of data transfer from the orchestrator to each learner $k \in \mathcal{K}$ in PL. However, since the ML models sizes usually dominate the dataset sizes (especially batches or subsets of data), the difference between the achievable local updates τ is nominal. It is more pronounced for the pedestrian dataset in Fig. 3.9 because the model used is much smaller, making its size comparable to the dataset size.

In terms of the validation accuracy performance, both HA-FL/PL provide gains up to 0.7% for MNIST classification and 8% for pedestrian classification with



(a) (b)

Fig. 3.8 Achievable local updates τ for: (**a**) the MNIST dataset versus K for $T = 30$s and $T = 60$s and versus T for $K = 10$ and 20 (**b**) the Pedestrian dataset versus K for $T = 1$s and $T = 2$s and versus T for $K = 10$ and 20

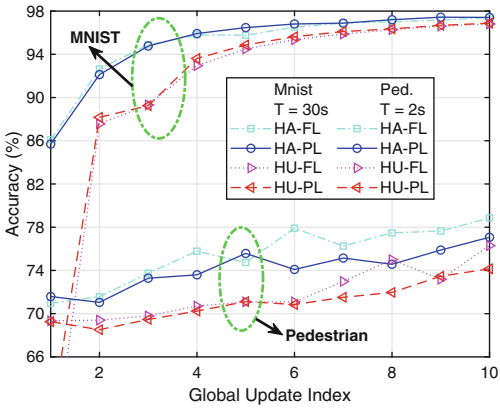


Fig. 3.9 Validation accuracy results HA/HU-Sync for both PL and FL approaches for MNIST classification with $K = 10$ and $T = 60$s and Pedestrian classification with $K = 10$ and $T = 5$s

convergence time reductions up to 53% as illustrated in Fig. 3.9. While the goal of FL and PL is different for various applications, we compare the performance of both to establish scientific rigor. In general, the performance of both HA-FL/PL is similar, especially for the MNIST datasets where both schemes have similar number of local updates. However, FL does reach many accuracy milestones faster than PL such as having a lower convergence time by 50% to reach a validation accuracy of 77% for the pedestrian dataset as shown in Fig. 3.9. Nevertheless, both approaches reach a similar final validation accuracy for both tasks. The initial speedup achieved by FL may directly result from performing more local updates per global update. However, recall that FL updates the local model using only a subset of the same local dataset which makes it more "deterministic" compared to PL where SGD may be applied by shuffling the dataset in between global updates. This implies that the improvements offered by more local updates τ per global cycle may be overshadowed by the more "stochastic" nature of HA-PL compared to FL.

3.6.3 Comparison to Centralized Approaches

To demonstrate the superiority of MEL in general for resource-constrained wireless edge environments, we compare both PL and FL against centralized approaches in terms of achievable validation accuracy and communication overhead. Centralized approaches imply that the learning task is executed at a single central entity which assumes to be an edge server such as a base station. For PL, it will simply mean the orchestrator sends all data to the server distributing among its peers. For FL, it will involve all K learners transmitting their local dataset to the server. Thus, the training phase will comprise three steps: transmission of the data from one or more learners to the edge, ML training within a time limit, followed by return of the trained model to each learner. Assuming a global cycle time T, the total training time is set to $10T$ for MNIST classification and $20T$ for pedestrian. The

Table 3.2 Final accuracy and communication overhead of the HA schemes compared to centralized learning

Environment	Dataset	K	T (s)	PL acc	FL acc	Cen acc	PL over	FL over	Cen over
802.11	MNIST	20	12	**97.31 %**	-	94.63 %	39.60 %	-	1.25 %
802.11	MNIST	10	30	**97.41 %**	-	95.35 %	31.46 %	-	0.38 %
802.11	Pedestrian	20	03	**81.86 %**	-	50.00 %	22.78 %	-	2.73 %
802.11	Pedestrian	10	05	**80.76 %**	-	50.00 %	33.05 %	-	6.83 %
Cellular	MNIST	10	60	**97.41 %**	97.40 %	95.27 %	14.69 %	1.47 %	0.02 %
Cellular	Pedestrian	10	05	**80.76 %**	80.34 %	50.00 %	19.62 %	2.30 %	0.01 %
Cellular	Pedestrian	10	02	80.15 %	**80.83 %**	50.00 %	42.70 %	4.52 %	0.89 %
Cellular	Pedestrian	20	02	79.38 %	**81.98 %**	50.00 %	46.03 %	4.74 %	0.79 %

communication overhead for each learner is $V_k = (t_k^S + t_k^R)/t_k \ \forall \ k \in \mathcal{K}$ which is the ratio of the communication to total time. The average overhead is given by $1/K \sum_{k=1}^{K} V_k$. While the experiments are done with the same parameters described in Section 3.5.2, the edge server for the cellular environment is assumed to have far superior compute capabilities represented by a clock speed of 10 GHz. Table 3.2 shows that the centralized approaches either completely fail or at least provide an inferior validation accuracy performance. In contrast, MEL with either FL or PL does incur a higher communication overhead cost with frequent model updates. As expected, PL has the highest communication overhead due to the additional data transfer component.

3.6.4 Complexity Analysis and Execution Time

In addition to the time needed for ML model training, the HA-Sync schemes will require an additional amount of time for optimization which must be considered. Recall that the problem of interest is a non-convex QCLP after relaxation which is typically solved using interior point methods whereas the proposed solution for HA-Sync relies upon solving a Kth degree-polynomial. Most polynomial solvers use the companion matrix method where the complexity mainly arises from the QR factorization needed for diagonalization and can be expressed as $\mathbf{O}\left(4/3K^3 + K^2\right)$ [34]. In contrast, for a convex QCLP with n quadratic matrices and m quadratic constraints, the computational complexity is $\mathbf{O}\left(n^{1/2}\left[m+n\right]n^2\right)$ [35] which can be expressed as $\mathbf{O}\left(\left[K+1\right]^{1/2}\left[2K^3 + 5K^2 + 4K + 1\right]\right)$ for our problem. However, because the problem is non-convex, the complexity is actually much higher.

Figure 3.10 better illustrates the complexity in terms of execution time of HA-PL-Ana, HA-PL-Num, and HU-PL. The optimizations are done on an 8-core Intel i7 2.4 GHz processor; numerical optimizations were done using the OPTI toolbox [25] used for the numerical optimization, whereas MATLAB's polynomial solver were used for HA-PL-Ana. The experiments were repeated 100 times for each

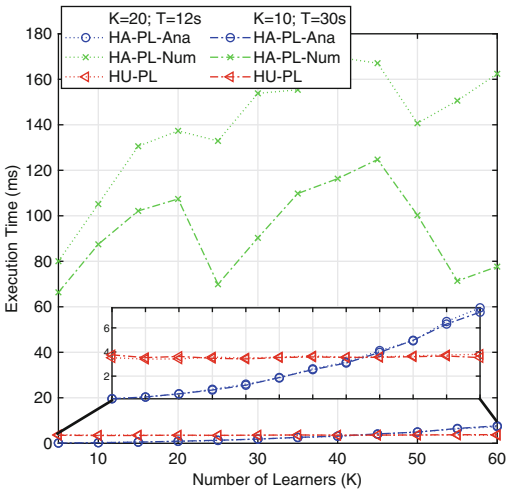


Fig. 3.10 HA optimization algorithm execution time comparisons using the MNIST dataset for the PL scheme versus K for $T = 30$ and 60 s

configuration of K learners with two different values of $T = 30$ and 60 s. As expected, the numerical solution with HA-PL-Num incurred the highest execution time more than 500% in most cases. Further, decreasing T seemed to have a high impact which can be explained by the tightening of the solution space. In contrast, the box plot demonstrates doing HA-Sync with the proposed analytical upper bounds requires an optimization time comparable to the HU approach. In fact, for up to a system of 40 learners, the execution time required is much lower compared to HU-PL which has a constant-time complexity as it requires solving a simple linear program. Moreover, even though the subsequent increase is exponential, it is on the order of milliseconds which is negligible compared to the substantial improvements offered in convergence time reductions and validation accuracy improvement.

3.6.5 Performance with Energy Constraints

The subsection's major contribution is to quantify the impact of local energy consumption limits $E_k^0 \ \forall \ k \ \in \ \mathcal{K}$ joules (J or watt-seconds) per global cycle in addition to the time constraint T. This is equivalent to $\frac{E_k^0}{3.6}$ milliwatt-hours (mWh). For example, a learner that uses up 20 J per global update cycle for 12 cycles will consume a total of 66.67 mWh. This is handy for real-world comparison as battery capacities are rated by their amperage (mAh) and voltage (V) which can easily be converted to mWh. Now, consider a smartphone battery rated at 4000 mAh and 3.6 V for a capacity of 14,400 mWh. For a learner consuming 20 J per global cycle, after 12 cycles, $\frac{66.67}{14{,}400} * 100 = 0.46\%$ of the battery would have been drained by the complete learning process.

To simulate the impact of different levels of energy availability, we define an average energy constraint EJ for all K learners across all global cycles. In each global cycle, the local energy consumption limit E_k^0 will be drawn from $[E-2.5, E+2.5]$ J to represent the differences in the available/dedicated energy to the learning task by each learner $k \in \mathcal{K}$. We use the set of values $\{10,\ 15,\ 20,\ 25\}$ which corresponds to a battery drainage of $\{0.23,\ 0.35,\ 0.46,\ 0.57\}$%. With respect to the other aspects of the simulation setup, we consider a cellular environment and the MNIST classification task to test the HA/HU Sync with dual-time/energy constraints. All of the remaining relevant parameters are as described in Table 3.1 in Sect. 3.5.

Figure 3.11a compares the HA scheme against the HU scheme in terms of the achievable number of local updates τ per global cycle. Clearly, the HA schemes provide a higher number of local updates compared to HU with gains up to 300%. Further, as energy consumption limits are increased, the HU schemes fail to show any improvement in the number of local updates. In contrast, the HA schemes can offer increasing gains in achievable local updates as T is increased as clearly shown by Fig. 3.11b. One reason may be that a higher T gives more flexibility in performing the batch size allocation. This behavior is reflected directly in the validation accuracy performance as depicted in Figs. 3.11d–g. As the allowed average energy consumption is increased from 10 to 15 J per global cycle, there is a bump in the performance of HU schemes but then it saturates. In contrast, the HA schemes are able to offer about a 0.1% gain in validation accuracy at each global

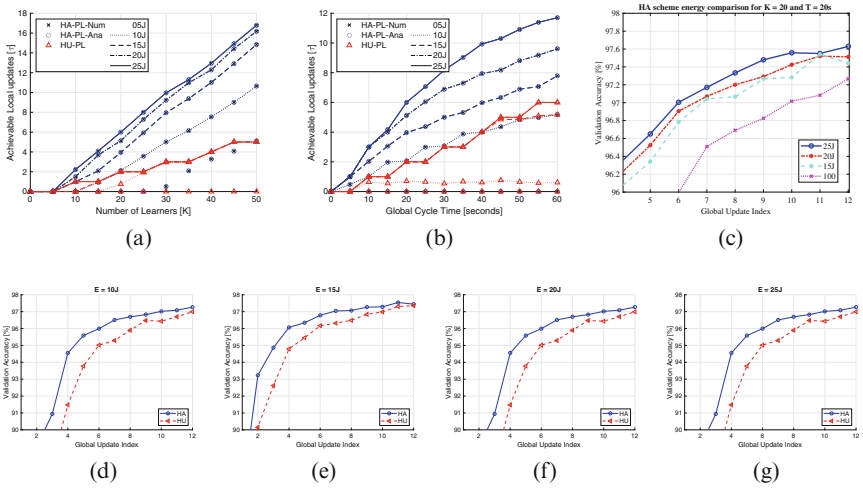


Fig. 3.11 Achievable number of local updates τ for average energies of 5 to 25 J in steps of 5 J by all schemes (**a**) vs K for $T = 20$s and (**b**) vs T for $K = 20$. (**c**) Validation accuracy progression for up-to 12 global updates for $K = 20$ for all HA schemes at different levels of energy constraint. HA vs HU validation comparison for energy constraint levels of (**d**) E =10J, (**e**) E =15J, (**d**) E =20J and (**e**) E =25J

update step each time the average energy consumption is increased by 5 J. With the exception of global update 11 which is an outlier, this behavior is corroborated by Fig. 3.11c. The major takeaway is that the HA schemes offer the best performance timeliness and energy constraints are the most stringent; they also offer the highest gains when the constraints are relaxed.

3.7 Extension of IoMT/H-IoT to EEG Data

With the increasing acceptance of ML techniques by the medical community for image-based diagnostics, attention has now turned to real-time ML-based event prediction with H-IoT/IoMT. Almost invariably, most applications of H-IoT rely upon multichannel time series data from various sensors. For example, one application is fall detection for people with dementia, Alzheimer's disease (AD), or Parkinson's disease (PD) using accelerometers, motion sensors, and gait sensors [36]. In case an inertial measurement unit is used, data comes from three channels (pitch, roll, and yaw) as a time series. Another possible application is cardiac event detection such as atrial fibrillation [37] from multi-lead electrocardiogram (ECG).

We will focus our discussion on another exciting application which is real-time seizure detection and/or early prediction using wearable electroencephalography (EEG) sensors. However, the mathematical formulations and methods discussed apply equally to any real-time multichannel time series data. Though offline training from benchmark datasets may be suitable for design and development of new predictive models and architectures, real-time inference with partial re-training for personalization is more suitable for real-world applicability. In practice, offline learning from large datasets is suitable for coming up with new models and techniques. However, the purpose of using ML for automatic diagnosis of cardiac events is better served by applying these models directly or with transfer learning at the edge for real-time inference as shown in for atrial fibrillation detection and potentially re-training for personalization [38]. Because of their superiority to other form of sensors for seizure detection/prediction and better accessibility compared to implant-based technologies, epileptic seizure detection/prediction using wearable EEG with end-to-end deep learning is an active area of research. Consider a set of users wearing long-term wearable wireless EEG monitoring devices co-located in a dense urban environment comprising multiple edge servers. This scenario can happen in a clinical setting such as a neurology ward with multiple patients of epilepsy. Alternatively, it can occur in home care settings where facilities cater to specific populations that may have epileptic seizures as a symptom of their disease such as cerebral palsy or tuberous sclerosis (TSC). If the people who suffer from scenarios are equipped with wearable EEG, one large predictive model can be deployed initially but then re-trained partially to personalize for a specific patient. Similar strategies are followed at an individual level for implant-based seizure control technologies such as responsive neuro-stimulation (RNS) where it takes up to 6 months for the device to become effective.

If we apply this strategy to seizure prediction with wearables, collecting all the data from multiple users at the edge, transmitting to the cloud, re-training, and deploying the model may not be feasible. It will put an unnecessary load on the backhaul networks where upload speeds may be limited and give rise to privacy concerns. Recently, leveraging the power of ES and fog nodes training with federated learning at the edge with the concept of "trusted edge" in the healthcare and biomedical sectors has been proposed [9]. Instead of cloud-based training, the predictive model can be co-cooperatively re-trained either fully or partially using MEL with associated capable fog nodes. The data can remain locally stored for sharing with medical professionals later or discarded depending upon the users' choices. Encryption or distortion techniques may be used to ass another layer of privacy [39].

3.7.1 Mathematical Formulation for EEG data

Consider a set of users who have epilepsy and are using wearable EEG along with a companion app running on a smartphone that executes the inference model and generates a seizure alert. The smartphone serves as the edge device or "learner" and uses a 4G base station as the edge server or orchestrator in this scenario. The learners will execute the inference model on segments of EEG data and the presence or lack of a seizure alert will be the class label for this data. The orchestrator can improve the general model by running FL on a set of multiple learners. In contrast, if an individual user wants to "personalize" the model to its own data, it can enlist the help of trusted edge devices (e.g., laptops, wireless routers, smart home equipment, other smartphones, etc.) in the vicinity to act as learners.

In this scenario, the expression for the model size in (3.7) will change as follows:

$$B_k^{model} = \mathcal{P}_m \left(d_k S_d + S_m\right) \quad \forall\, k \in \mathcal{K} \tag{3.41}$$

where $\mathcal{P}_m$ accounts for the compression ratio and bit precision, S_m represents the size of the non-trainable part/layers, and the term $d_k S_d$ is replaced S_r, which represents the size of the part/layers of the model being re-trained. In addition, the time series for each learner $k \in \mathcal{K}$ will be collected from a frame of duration T_k^D seconds sampled at F_k^S Hz via N_k^C channels. This will mainly impact the PL scenario where the expression for the data subset size in (3.6) changes as follows:

$$B_k^{data} = d_k F_k^S T_k^D N_k^C \mathcal{P}_d \quad \forall\, k \in \mathcal{K} \tag{3.42}$$

where $\mathcal{P}_d$ is accounts for the precision/compression ratios. The rest of the model, solutions, and algorithm in Sects. 3.2, 3.3, 3.4, and 3.5 can be applied as discussed prior.

This is a very brief description without a detailed discussion of the results. However, there are several other innovations possible which though beyond the

scope of the chapter, we mention here briefly. For example, specific to PL, the physical parameters related to the sensors including sampling rates, frame durations, and number of channels can also be optimization parameters. Further, the size of the model used for re-training can be part of the problem statement. All of this will require extensive solutions to the resulting optimization problem and analysis of the complexity to guarantee the performance gains. Further, the optimization problem itself can be part of the MEL solution with techniques such as distributed deep reinforcement learning (DDRL). This will be more in line with the trend of moving away from central optimization to distributed optimization. The benefits include avoiding a single point of failure, more robustness to injection attacks, and more flexibility in ML model training.

Acknowledgments This material is based upon work supported by the National Science Foundation (NSF) under Grant No. TI-2213951. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Appendix 1

The global model' loss is denoted by $F(\mathbf{w})$. Let us assume that this function has the following three properties:

1. $F(\mathbf{w})$ is convex.
2. $F(\mathbf{w})$ is ρ-Lipschitz: $\|F(\mathbf{w}) - F(\bar{\mathbf{w}})| \leq \rho|\mathbf{w} - \bar{\mathbf{w}}|$
3. $F(\mathbf{w})$ is β-smooth: $\|\nabla F(\mathbf{w}) - \nabla F(\bar{\mathbf{w}})| \leq \beta|\mathbf{w} - \bar{\mathbf{w}}|$

Each learner $k \in \mathcal{K}$ in HA-Sync performs a total of L updates. Then, the difference between the loss at update L and the optimal global model denoted by $\mathbf{w}^*$ is bounded by:

$$F(\mathbf{w}[L] - F(\mathbf{w}^*) \leq \frac{1}{G\tau\,[A + B(1 - C)]} \tag{3.43}$$

The learning rate is given by η and we can define a control parameter $\phi = 1 - \frac{\eta\beta}{2}$. The local losses are bound by the parameter ϵ whereas the function $h(\tau) = \frac{\delta}{\beta}[(\eta\beta + 1)^{\tau} - 1] - \eta\delta\tau$. For more details on δ and ϵ, the reader is referred to [12]. In general, η is selected such that $0 < \eta\beta < 1$, $\eta\phi - \frac{\rho h(\tau)}{\tau\epsilon^2} \geq 0$, and $(\eta\beta + 1)^{\tau} \geq \eta\beta\tau + 1$. Consider the case where MEL is not optimized and each global cycle allows each learner $k \in \mathcal{K}$ to perform the same integer number of local updates τ. Then, in a fixed number of global updates G, MEL will allow for a total of $L = G\tau$ updates. Let us define the constants $A = \eta\phi + \frac{\rho\delta}{\epsilon^2}$, $B = \frac{\rho\delta}{\beta\epsilon^2}$, and $C = \eta\beta + 1$. Based on these definitions and assumptions, we can define the upper bound on the loss as follows:

$$F(\mathbf{w}[L] - F(\mathbf{w}^*)) \leq \frac{1}{G\tau\,[A + B(1-C)]} \tag{3.44}$$

It is observable that $A + B(1 - C) \geq 0$, and further, the number of global updates G are fixed. Hence, the bound on the loss will converge to zero as $\tau \to \infty$.

Appendix 2

Let us write the KKT optimality conditions for (3.25) as shown in (3.45) and (3.51). The conditions (3.45) ensures that dataset size of any learner $k \in \mathcal{K}$ satisfies (3.27) and (3.47) ensures that the bound in (3.27) holds with equality for any learner $k \in \mathcal{K}$ if $\lambda_k^* \geq 0$. This is significant because strong duality holds for some feasible τ^* when strictly speaking, $\lambda_k^* > 0\ \forall\ k \in \mathcal{K}$. This means the upper bound will be the optimal solution.

$$C_k^2\tau^* d_k^* + C_k^1 d_k^* + C_k^0 - T \leq 0, \quad k \in \mathcal{K} \tag{3.45}$$

$$\alpha_0^*, \alpha_k^*, \text{ and } \lambda_k^* \geq 0 \quad k \in \mathcal{K} \tag{3.46}$$

$$\lambda_k^* \left(C_k^2\tau^* d_k^* + C_k^1 d_k^* + C_k^0 - T\right) = 0, \quad k \in \mathcal{K} \tag{3.47}$$

$$-\alpha_0^*\tau^* = 0 \tag{3.48}$$

$$-\alpha_k^* d_k^* = 0 \quad k \in \mathcal{K} \tag{3.49}$$

$$\sum_{k=1}^{K} d_k^* - d = 0 \tag{3.50}$$

$$-\nabla\tau^* + \sum_{k=1}^{K} \lambda_k^* \nabla \left(C_k^2\tau^* d_k^* + C_k^1 d_k^* + C_k^0 - T\right) + \nu^* \nabla \left(\sum_{k=1}^{K} d_k^* - d\right) - \alpha_0^* \nabla\tau^* - \nabla \left(\sum_{k=1}^{K} \alpha_k^* d_k^*\right) = 0 \tag{3.51}$$

We can then rewrite the bound on d_k^* in (3.27) as an equality and substitute it back in (3.50) to obtain the following relation:

$$d = \sum_{k=1}^{K} d_k^* = \sum_{k=1}^{K} \left[\frac{T - C_k^0}{\tau^* C_k^2 + C_k^1}\right] = \sum_{k=1}^{K} \left[\frac{a_k}{\tau^* + b_k}\right] \tag{3.52}$$

The expression on the rightmost hand-side has the form of a partial fraction expansion of a rational polynomial function of τ^* where $a_k, b_k \in \mathcal{R}^{++}$. Therefore, we can expand (3.52) to the form shown in (3.53).

$$\frac{a_1}{\tau^*+b_1}+\frac{a_2}{\tau^*+b_2}+\cdots+\frac{a_k}{\tau^*+b_k}+\cdots+\frac{a_K}{\tau^*+b_K}=$$
$$\frac{1}{(\tau^*+b_1)(\tau^*+b_2)\ldots(\tau^*+b_k)\ldots(\tau^*+b_K)}\times$$
$$\Big[a_1(\tau^*+b_2)(\tau^*+b_3)\ldots(\tau^*+b_k)\ldots(\tau^*+b_K)+$$
$$a_2(\tau^*+b_1)(\tau^*+b_3)\ldots(\tau^*+b_k)\ldots(\tau^*+b_K)+\ldots+$$
$$a_k(\tau^*+b_1)(\tau^*+b_2)\ldots(\tau^*+b_{k-1})(\tau^*+b_{k+1})\ldots(\tau^*+b_K)$$
$$+\cdots+a_K(\tau^*+b_1)(\tau^*+b_2)\ldots(\tau^*+b_k)\ldots(\tau^*+b_{K-1})\Big] \tag{3.53}$$

Finally, the expanded form can be cleaned up in the form of a rational function with respect to τ^*, which is equal to the total dataset size d as shown in (3.54). Please note that the degrees of the numerator and denominator will be $K-1$ and K, respectively. Furthermore, the poles of the system will be $-b_k$, and since $b_k \geq 0$, the system will be stable. Furthermore, $\tau^* = -b_k$ is not a feasible solution for the problem because it is eliminated by the $\tau \geq 0$ constraint. Therefore, we can rewrite (3.54) as shown in (3.28). By solving this polynomial, we obtain a set of solutions for τ^*, where one of them is feasible. The problem being non-convex, this feasible solution τ^* will constitute the upper bound to the solution of the relaxed problem.

$$d = \frac{\sum_{k=1}^{K} a_k \prod_{\substack{l=1 \\ l\neq k}}^{K} (\tau^*+b_l)}{\prod_{k=1}^{K}(\tau^*+b_k)} \tag{3.54}$$

As a last step, to ensure that the solution set is feasible, it must be noted that according to (3.48) and (3.49), α_0^* and α_k^* $\forall$ k must be equal to 0. Expanding the vanishing gradient condition in (3.51), it can be shown that the following two relations can be derived (representing $K+1$ equations):

$$\lambda_k^* C_k^2 \tau^* + \lambda_k^* C_k^1 + \nu^* = \alpha_k^*,\ k \in \mathcal{K} \tag{3.55}$$

$$-1 + \sum_{k=1}^{K} \lambda_k^* C_k^2 d_k^* = \alpha_0^* \tag{3.56}$$

By setting $\alpha_0^* = 0$ and $\alpha_k^* = 0$ for $k \in \mathcal{K}$, we can write λ_k^* in terms of ν^* as shown in (3.57) and substitute the resulting expression in (3.56) to find ν^* using the values of d_k^* and τ^* obtained from (3.27) and (3.28), respectively.

$$\lambda_k^* = -\frac{\nu^*}{C_k^2\tau^* + C_k^1},\ k \in \mathcal{K} \tag{3.57}$$

$$\nu^* = -\frac{1}{\sum_{k=1}^{K} \frac{C_k^2 d_k^*}{C_k^2 \tau^* + C_k^1}} \tag{3.58}$$

The values of λ_k^* for $k \in \mathcal{K}$ can be obtained by back-substitution of ν^* in (3.57). As one can observe, as long as there exists a τ^* greater than zero, ν^* will be negative and hence, λ_k^* for $k \in \mathcal{K}$ will be strictly greater than zero. Hence, as long as there exists a $\tau^* > 0$ in the feasible set such that $d_k^* > 0$, there will exist a set of $\lambda_k^* > 0$ for $k \in \mathcal{K}$. This fact can be used to verify the feasibility of the solution. This step is also helpful when there may exist multiple values of τ greater than zero for choosing the optimal τ^*. Extensive simulations presented in Sect. 3.5 demonstrated that there was *no* optimality gap between the analytical upper bounds and the numerical solution.

Appendix 3

Recall that the optimization variables are given by $\mathbf{x} = [\tau \; d_1 \; d_2 \; \ldots \; d_k \; \ldots \; d_K]^T$. Then, the relaxed problem in (3.25) can be rewritten in the standard form of a QCQP as follows:

$$\min_{\mathbf{x}} \quad \mathbf{x}^T \mathbf{F} \mathbf{x} + \mathbf{f}^T \mathbf{x} + f_0 \tag{3.59}$$

$$\text{s.t.} \quad \mathbf{x}^T \mathbf{P}_k \mathbf{x} + \mathbf{p}_k^T \mathbf{x} + p_k^0 \leq 0, \quad \forall k \in \mathcal{K} \tag{3.59a}$$

$$\mathbf{x}^T \mathbf{Q}_k \mathbf{x} + \mathbf{q}_k^T \mathbf{x} + q_k^0 \leq 0, \quad \forall k \in \mathcal{K} \tag{3.59b}$$

$$\mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{a}^T \mathbf{x} + a_0 \leq 0 \tag{3.59c}$$

$$\mathbf{x}^T \bar{\mathbf{A}} \mathbf{x} + \bar{\mathbf{a}}^T \mathbf{x} + \bar{a}_0 \leq 0 \tag{3.59d}$$

$$\mathbf{x}^T \mathbf{U} \mathbf{x} + \mathbf{U}^T \mathbf{x} + u_0 \leq 0 \tag{3.59e}$$

$$\mathbf{x}^T \mathbf{V}_k \mathbf{x} + \mathbf{v}_k^T \mathbf{x} + v_k^0 \leq 0, \quad \forall k \in \mathcal{K} \tag{3.59f}$$

The time and energy constraints are defined by (3.59a) and (3.59b), respectively. Constraints (3.59c) and (3.59d) are two inequality constraints used to simplify the equality constraint of total dataset size allocation. The nonnegative constraints on τ and d_k are given in (3.59e) and (3.59f), respectively. The expressions for problem definition and each constraint have three terms each: a quadratic term, a linear term, and a constant term. The constant terms $p_k^0 = C_k^0 - T$ and $q_k^0 = G_k^0 - E_k^0 \; \forall \; k \in \mathcal{K}$ and are associated with the time and energy constraints, respectively. The remaining constant terms $a_0 = -d$, $\bar{a}_o = d$ and $v_k^0 = d_l, \forall \; k$ whereas $u_0 = 0$ and $f_0 = 0$. In contrast, the coefficients associated with the linear terms in the objective ($\mathbf{f}$) and

constraints ($\mathbf{p}_k$, $\mathbf{q}_k$, $\mathbf{a}$, $\bar{\mathbf{a}}$, $\mathbf{u}$, and $\mathbf{v}_k$) can be represented by the following set of vectors:

$$\begin{aligned}
\mathbf{f} &= \begin{bmatrix} -1\,0\,0\,\ldots\,C_k^1\,\ldots\,0 \end{bmatrix}^T \\
\mathbf{p}_k &= \begin{bmatrix} 0\,0\,0\,\ldots\,C_k^1\,\ldots\,0 \end{bmatrix}^T, \forall\, k \\
\mathbf{q}_k &= \begin{bmatrix} 0\,0\,0\,\ldots\,G_k^1\,\ldots\,0 \end{bmatrix}^T, \forall\, k \\
\mathbf{a} &= [0\,1\,1\,\ldots\,1\,\ldots\,1]^T \\
\bar{\mathbf{a}} &= [0\,-1\,-1\,\ldots\,-1\,\ldots\,-1]^T \\
\mathbf{u} &= [-1\,0\,0\,\ldots\,0\,\ldots\,0]^T \\
\mathbf{v}_k &= [0\,0\,0\,\ldots\,-1\,\ldots\,0]^T, \forall\, k
\end{aligned} \tag{3.60}$$

In general, the coefficients associated with the quadratic terms are $(K+1) \times (K+1)$ matrices. Because this is a QCLP, the quadratic term in the objective is $\mathbf{0}_{(K+1)\times(K+1)}$, a $(K+1)\times(K+1)$ zero matrix. The coefficients associated with the time and energy constraints, $\mathbf{P}_k$ and $\mathbf{Q}_k$, respectively, can be described as follows:

$$\mathbf{P}_k(i,j) = \begin{cases} 0.5C_k^2, & \text{if } \begin{array}{l} i=1\ \&\ j=k+1 \\ i=k+1\ \&\ j=1 \end{array} \\ 0, & \text{otherwise} \end{cases} \tag{3.61}$$

$$\mathbf{Q}_k(i,j) = \begin{cases} 0.5G_k^2, & \text{if } \begin{array}{l} i=1\ \&\ j=k+1 \\ i=k+1\ \&\ j=1 \end{array} \\ 0, & \text{otherwise} \end{cases} \tag{3.62}$$

The quadratic coefficients of the remaining constraints $\mathbf{A}$, $\bar{\mathbf{A}}$, $\mathbf{U}$ and $\mathbf{V}_k$ are all $\mathbf{0}_{(K+1)\times(K+1)}$. We can now define the functions $\mathbf{F}^2(\mathbf{\Gamma})$, $\mathbf{f}^1(\mathbf{\Gamma})$ and $f_0(\mathbf{\Gamma})$ as [26]:

$$\mathbf{F}^2(\mathbf{\Gamma}) = \sum_{k=1}^{K} \lambda_k \mathbf{P}_k + \gamma_k \mathbf{Q}_k \tag{3.63}$$

$$\mathbf{f}^1(\mathbf{\Gamma}) = \sum_{k=1}^{K} (\lambda_k \mathbf{p}_k + \gamma_k \mathbf{q}_k + \nu_k \mathbf{v}_k) + \alpha \mathbf{a} + \bar{\alpha}\bar{\mathbf{a}} + \omega \mathbf{u} \tag{3.64}$$

$$f_0(\mathbf{\Gamma}) = \sum_{k=1}^{K} \left(\lambda_k p_k^0 + \gamma_k q_k^0 + \nu_k v_k^0 \right) + \alpha a_0 + \bar{\alpha}\bar{a}_0 \tag{3.65}$$

References

1. B. Jovanovic, Internet of things statistics for 2023 – taking things apart Online (2023). https://dataprot.net/statistics/iot-statistics/
2. B. McMahan, D. Ramage, Federated learning: collaborative machine learning without centralized training data Online (2017). https://ai.googleblog.com/2017/04/federated-learning-collaborative.html
3. Markets and markets, Edge computing in healthcare market | Revenue trends and growth drivers Online (2023). https://www.marketsandmarkets.com/Market-Reports/edge-computing-in-healthcare-market-133588379.html#::text=Theglobaledgecomputingin,26.1%25from2022to2028
4. Statista, Internet of Things – US | Statista market forecast Online (2023). https://www.statista.com/outlook/tmo/internet-of-things/united-states
5. Sciforce, Can Edge Analytics Become a Game Changer? – Sciforce – Medium Online (2019). https://medium.com/sciforce/can-edge-analytics-become-a-game-changer-9cc9395d2727
6. S. Samarakoon, M. Bennis, W. Saad, M. Debbah, Federated learning for ultra-reliable low-latency V2V communications, in *2018 IEEE Global Communications Conference, GLOBECOM 2018 – Proceedings* (Institute of Electrical and Electronics Engineers Inc., Dubai, UAE, 2018). Online. https://ieeexplore.ieee.org/document/8647927
7. B. Hu, Y. Gao, L. Liu, H. Ma, Federated region-learning: an edge computing based framework for urban environment sensing, in *2018 IEEE Global Communications Conference, GLOBECOM 2018 – Proceedings*. (Institute of Electrical and Electronics Engineers Inc., Dubai, UAE, 2018). Online. https://ieeexplore.ieee.org/document/8647649
8. J. Jeon, J. Kim, J. Huh, H. Kim, S. Cho, Overview of distributed federated learning: research issues, challenges, and biomedical applications, in *2019 International Conference on Information and Communication Technology Convergence (ICTC)* (IEEE, Jeju Island, South Korea, 2019), pp. 1426–1427. Online. https://ieeexplore.ieee.org/document/8939954/
9. N. Rieke, J. Hancox, W. Li, F. Milletarì, H.R. Roth, S. Albarqouni, S. Bakas, M.N. Galtier, B.A. Landman, K. Maier-Hein, S. Ourselin, M. Sheller, R.M. Summers, A. Trask, D. Xu, M. Baust, M.J. Cardoso, The future of digital health with federated learning. NPJ Digital Med. **3**(1), 1–7 (2020). Online. http://dx.doi.org/10.1038/s41746-020-00323-1
10. W. Yang, B. Lim, N.C. Luong, D.T. Hoang, Federated learning in mobile edge networks : a comprehensive survey. IEEE Commun. Surv. Tutorials (Early Access), 1–33 (2020). Online. https://ieeexplore.ieee.org/document/9060868
11. L. Bottou, O. Bousquet, The tradeoffs of large scale learning, in *Advances in Neural Information Processing Systems*, ed. by J.C. Platt, D. Koller, Y. Singer, S. Roweis. NIPS Foundation (http://books.nips.cc), vol. 20, (2008), pp. 161–168. Online. http://leon.bottou.org/papers/bottou-bousquet-2008
12. S. Wang, T. Tuor, T. Salonidis, K.K. Leung, C. Makaya, T. He, K. Chan, Adaptive federated learning in resource constrained edge computing systems. IEEE J. Sel. Areas Commun. Early Access, 1–1 (2019). Online. https://ieeexplore.ieee.org/document/8664630/
13. S. Teerapittayanon, B. McDanel, H.T. Kung, Distributed deep neural networks over the cloud, the edge and end devices, in *Proceedings – International Conference on Distributed Computing Systems*, pp. 328–339
14. J. Dean, G.S. Corrado, R. Monga, K. Chen, M. Devin, Q.V. Le, M.Z. Mao, M.A. Ranzato, A. Senior, P. Tucker, K. Yang, A.Y. Ng, Large scale distributed deep networks, in *Advances in Neural Information Processing Systems*, vol. 25 (2012), pp. 1223–1231. Online. https://papers.nips.cc/paper/4687-large-scale-distributed-deep-networks
15. S. Wang, T. Tuor, T. Salonidis, K.K. Leung, C. Makaya, T. He, K. Chan, When edge meets learning : adaptive control for resource-constrained distributed machine learning, in *INFOCOM* (2018). Online. https://ieeexplore.ieee.org/document/8486403

16. U.Y. Mohammad, S. Sorour, Multi-objective resource optimization for hierarchical mobile edge computing, in *2018 IEEE Global Communications Conference: Mobile and Wireless Networks (Globecom2018 MWN)*, Abu Dhabi, United Arab Emirates (2018), pp. 1–6. Online. https://ieeexplore.ieee.org/document/8648109
17. H.H. Yang, Z. Liu, T.Q.S. Quek, H.V. Poor, Scheduling policies for federated learning in wireless networks. IEEE Trans Commun **68**(1), 317–333 (2019). Online. https://ieeexplore.ieee.org/document/8851249
18. Z. Yang, M. Chen, W. Saad, C.S. Hong, M. Shikh-Bahaei, Energy efficient federated learning over wireless communication networks. IEEE Trans Wirel Commun **20**(3), 1935–1949 (2020). Online. https://ieeexplore.ieee.org/document/9264742
19. M. Chen, Z. Yang, W. Saad, C. Yin, H.V. Poor, S. Cui, A joint learning and communications framework for federated learning over wireless networks. IEEE Trans. Wirel. Commun. **20**(1), 269–283 (2021). Online. https://ieeexplore.ieee.org/document/9210812, https://arxiv.org/abs/1909.07972
20. T. Tuor, S. Wang, T. Salonidis, B.J. Ko, K.K. Leung, Demo abstract: distributed machine learning at resource-limited edge nodes, in *INFOCOM 2018 – IEEE Conference on Computer Communications Workshops* (2018), pp. 1–2
21. D. Conway-Jones, T. Tuor, S. Wang, K.K. Leung, Demonstration of federated learning in a resource-constrained networked environment, in *2019 IEEE International Conference on Smart Computing (SMARTCOMP)* (2019). Online. https://ieeexplore.ieee.org/abstract/document/8784064
22. U. Mohammad, S. Sorour, Adaptive task allocation for asynchronous federated and parallelized mobile edge learning (2020) arXiv preprint, arXiv:1905.01656, Online. https://arxiv.org/abs/1905.01656
23. U. Mohammad, S. Sorour, Adaptive task allocation for mobile edge learning, in *2019 IEEE Wireless Communications and Networking Conference Workshop (WCNCW)* (IEEE, 2019), pp. 1–6. Online. https://ieeexplore.ieee.org/document/8902527/
24. A.D. Pia, S.S. Dey, M. Molinaro, Mixed-integer quadratic programming is in NP. Math. Program. **162**(1), 225–240 (2017)
25. J. Currie, D.I. Wilson, OPTI: lowering the barrier between open source optimizers and the industrial MATLAB user, in *Foundations of Computer-Aided Process Operations*, ed. by N. Sahinidis, J. Pinto (Savannah, Georgia, USA, 2012)
26. J. Park, S. Boyd, General heuristics for nonconvex quadratically constrained quadratic programming (2017) arXiv e-prints. Online. http://arxiv.org/abs/1703.07870
27. M. Chen, H.V. Poor, W. Saad, S. Cui, Convergence time minimization of federated learning over wireless networks, in *IEEE International Conference on Communications*. (Institute of Electrical and Electronics Engineers Inc., 2020), pp. 1–6. Online. https://ieeexplore.ieee.org/document/9148815
28. M. Chen, H.V. Poor, W. Saad, S. Cui, Convergence time optimization for federated learning over wireless networks. IEEE Trans Wirel Commun. **20**(4), 2457–2471 (2021). Online. https://ieeexplore.ieee.org/document/9148815
29. A. Abutuleb, S. Sorour, H.S. Hassanein, Joint task and resource allocation for mobile edge learning, in *2020 IEEE Global Communications Conference, GLOBECOM 2020 – Proceedings*. (Institute of Electrical and Electronics Engineers Inc., 2020), pp. 1–6. Online. https://ieeexplore.ieee.org/document/9322399
30. S. Cebula, A. Ahmad, J.M. Graham, C.V. Hinds, L.A. Wahsheh, A.T. Williams, S.J. DeLoatch, Empirical channel model for 2.4 GHz IEEE 802.11 WLAN, in *Proceedings of the 2011 International Conference on Wireless Networks* (2011)
31. S. Munder, D.M. Gavrila, An experimental study on pedestrian classification. IEEE Trans. Pattern Anal. Mach. Intell. **28**(11), 1863–1868 (2006). Online. https://ieeexplore.ieee.org/document/1704841
32. Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998). Online. https://ieeexplore.ieee.org/document/726791

33. B. Shillingford, What is the time complexity of backpropagation algorithm for training artificial neural networks? – Quora (2016). Online. https://www.quora.com/What-is-the-time-complexity-of-backpropagation-algorithm-for-training-artificial-neural-networks
34. F. Uhlig, The DQR algorithm, basic theory, convergence, and conditional stability. Numer. Math. **76**(4), 515–553 (1997)
35. A. Nemirovski, Interior point polynomial time methods in convex programming. Lect. Notes **42**(16), 3215–3224 (2004). Online. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.160.6909&rep=rep1&type=pdf
36. Y.A. Qadri, A. Nauman, Y.B. Zikria, A.V. Vasilakos, S.W. Kim, The future of healthcare internet of things: a survey of emerging technologies. IEEE Commun. Surv. Tutorials **22**(2), 1121–1167 (2020). Online. https://ieeexplore.ieee.org/document/8993839
37. J.M. Raja, C. Elsakr, S. Roman, B. Cave, I. Pour-Ghaz, A. Nanda, M. Maturana, R.N. Khouzam, Apple watch, wearables, and heart rhythm: where do we stand?. Ann. Transl. Med. **7**(17), 417–417 (2019). Online. http:///pmc/articles/PMC6787392/?report=abstract, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6787392/
38. S. Kiranyaz, T. Ince, M. Gabbouj, Real-time patient-specific ECG classification by 1-D convolutional neural networks. IEEE Trans. Biomed. Eng. **63**(3), 664–675 (2016)
39. O. Choudhury, A. Gkoulalas-Divanis, T. Salonidis, I. Sylla, Y. Park, G. Hsu, A. Das, Anonymizing data for privacy-preserving federated learning. Online. https://arxiv.org/abs/2002.09096