Chapter 6

A federated learning technique with heterogeneous devices and networking resources

This chapter proposes a fast federated learning technique to train a model for a given task at the participant devices in the presence of the heterogeneous device and networking resources. The proposed technique starts with the collection of available resource information of participant devices and selects a generic model which directly works on most of the devices. We next propose a knowledge distillation-based early-halting approach for devices, where the generic model does not fit directly. The early halting in the FFL technique speeds up the training of the model at the participant devices.

6.1 Introduction

A participant device in FL uses its resources, such as memory and processing power (which may not be the same for all devices), to load the model and train them locally. The availability of resources at the participant devices depends on their type and other installed services. Such heterogeneity in device resources requires unequal

time to train the model using local datasets. Moreover, a large-size model may not be successfully trained on a device with limited resources. Similarly, the time required to share the WPM between each participant device and the central server depends on the networking resources like bandwidth. The heterogeneity in the device and networking resources implies that all devices may not simultaneously transfer the WPM to the central server for the aggregation and hence slow down the FL. Figure 6.1 illustrates an example scenario of FL that trains a Locomotion Mode Recognition (LMR) model on the participant devices using local datasets. LMR model recognizes the locomotion mode used by the participant. The participant devices are the smartphones of different brands, and they have unequal memory and processing power. Since the participants are at different locations, the network bandwidth between the devices and central server may not be the same.

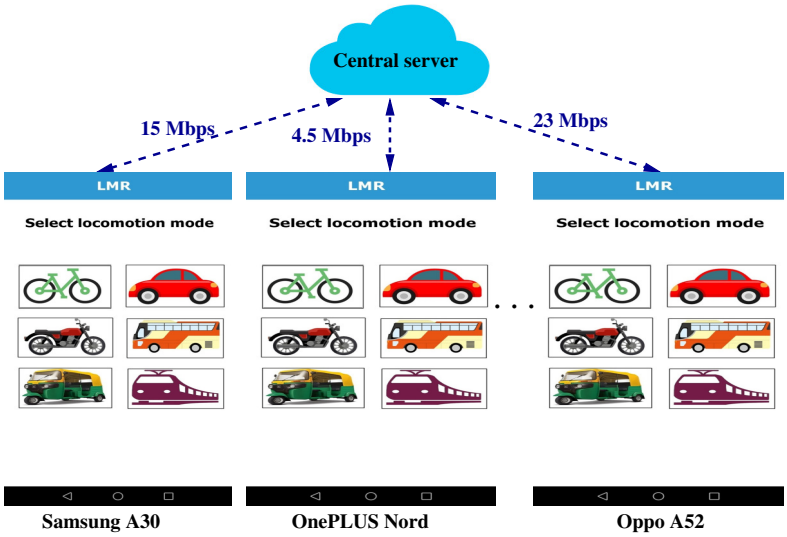


Figure 6.1: A scenario of FL with heterogeneous resources (unequal memory and processing power of smartphones, bandwidth).

To address the heterogeneity of resources among the participants in FL, prior studies proposed mechanisms that discard slow processing participants, called *stragglers*, from the federation [48–50]. However, the removal of stragglers hampers effective utilization of the local dataset (on stragglers) and prohibits performance improvement via FL.

Some existing work [51] have considered a fixed size model for all the devices of heterogeneous resources. The fixed size model may not fully utilize the colossal resources devices. The existing work [52,53] used Knowledge Distillation (KD) to resize and train the model that fit on the devices for FL. The KD is a student-teacher learning process. The training of the student model under the guidance of the teacher model requires multiple epochs in KD and delays the aggregation process at the central server. The existing work [54] considered unequal bandwidth issue in FL that faces weight staleness. Some devices update their WPM multiple times, while others may not participate.

While considering the heterogeneity of device and networking resources, we present a *Fast Federated Learning* (FFL) technique to train the deep learning models for prediction or classification tasks at the participant devices using the local datasets. Specifically, we address the following question: *how does FL successfully train a model on the participant devices with heterogeneous resources (unequal memory, processing power, and bandwidth) using local datasets?* To this end, the major contributions and novelty of this work are as follows:

• *Selection of a generic model*: The first contribution is to design an approach for selecting a generic model of a given task based on the available resources of participant devices. The central server collects the information of available resources of participant devices, divides the devices into categories using k-means clustering, and selects a model which supports the median category of devices. Different from the existing work [48, 54, 80], simultaneously considering the devices and networking resources makes the selected model not arbitrarily large or small and can successfully delay on insufficient and colossal resource devices.

• *Training of the generic model on participant devices with heterogeneous resources*: The next contribution is to design an approach for training of the generic model on participant devices using the local dataset with heterogeneous resources. Different from the existing work [81–83], the approach considers the scenarios where devices have suffi-

cient, colossal, and insufficient resources to train the model. The participant devices directly use the generic model in the scenario where available resources of the devices are sufficient to train the generic model. The approach uses knowledge distillation (student-teacher learning) to train resized generic models for insufficient and colossal resource devices. To speed up the training of the model at each participant device, the approach halts the teacher training after a certain halting epoch. We derive an expression to find the halting epoch for the given accuracy.

• *Aperiodic global update at central server*: We propose an aperiodic global update approach where the central server does not wait to receive the WPM from all the participants to estimate the updated WPM. The duration of two consecutive global updates divides into fixed time intervals. The central server aggregates the received WPM from the participants in each time interval and uses the aggregated WPM in the next time interval. Unlike existing work [51, 80], the aperiodic global update speedups the model's training in the presence of heterogeneous resources.

• *Experimental validation:* We perform a real-world study to evaluate the feasibility and performance of the FFL technique. In this study, we recruited 128 student volunteers. The volunteers' smartphones work as participant devices and the central server is located in the institute to execute the FL operations. The task of the real-world study was to recognize the locomotion modes used by the participants. The study considered six locomotion modes: bicycle, bike, car, auto-rickshaw, bus, and train. We also verify the FFL technique on the existing baseline techniques [52, 54], collected and existing datasets of locomotion modes [55], and three validation metrics. The results show that the FFL technique minimizes the training time with high accuracy.

• **Motivation:** We observed the following limitations in the existing work, which motivate our work. A model may not achieve adequate accuracy if its weights are discarded during global aggregation in FL [48]. Reducing the processing power of the device during training of the model slowdown the aggregation process [80, 81].

Suppressing the communication round for aggregation [21, 83] also increases the stale models at the participants. The parallel training and communication come with the cost of gradient-staleness [54]. Considering a fixed size of lightweight models is not suitable for unequal resources participant devices [86]. Sending WPM of the lightweight model to the central server increases the number of the round of global aggregation [52]. Moreover, using KD in FL slowdowns the training of models at the devices [51–53, 86]. *In summary, the existing FL technique in the presence of heterogeneous resources avoid the straggler devices during aggression at the central server, delay the aggression process, and/or reduce the number of aggregation round.* To overcome the above limitations, we propose a fast federated learning technique to train a model at the participant devices in the presence of heterogeneous resources. The technique early halts the training of the model and aperiodically updates the WPM of the participant devices.

The rest of the chapter is organized as follows. Section 6.2 presents the FFL technique to train a model on the participant devices with heterogeneous resources using local datasets. The real-world study and performance evaluation are discussed in Section 6.3 and Section 6.4, respectively. Finally, the chapter concludes in Section 6.5.

6.2 FFL technique

We propose a Fast Federated Learning (FFL) technique to train the deep learning models for prediction or classification tasks at the participant devices with heterogeneous resources. Figure 6.2 shows the framework of the FFL technique. The central server initiates the FFL with the collection of the available resource information from participant devices followed by the selection of an un-trained generic model, denoted as M_o. The server transfers the generic model M_o to all the participant devices. The generic model M_o is successfully trained on most of the participant devices. We next propose a KD-based early halting approach for insufficient or colossal resources devices to train generic model M_o. The halting approach speeds up the training process and

improves the performance of the model within available resources. Finally, we propose an aperiodic global update approach that helps the participant devices to share their updated WPM aperiodically. Algorithm 6.1 shows the steps of the FFL technique.

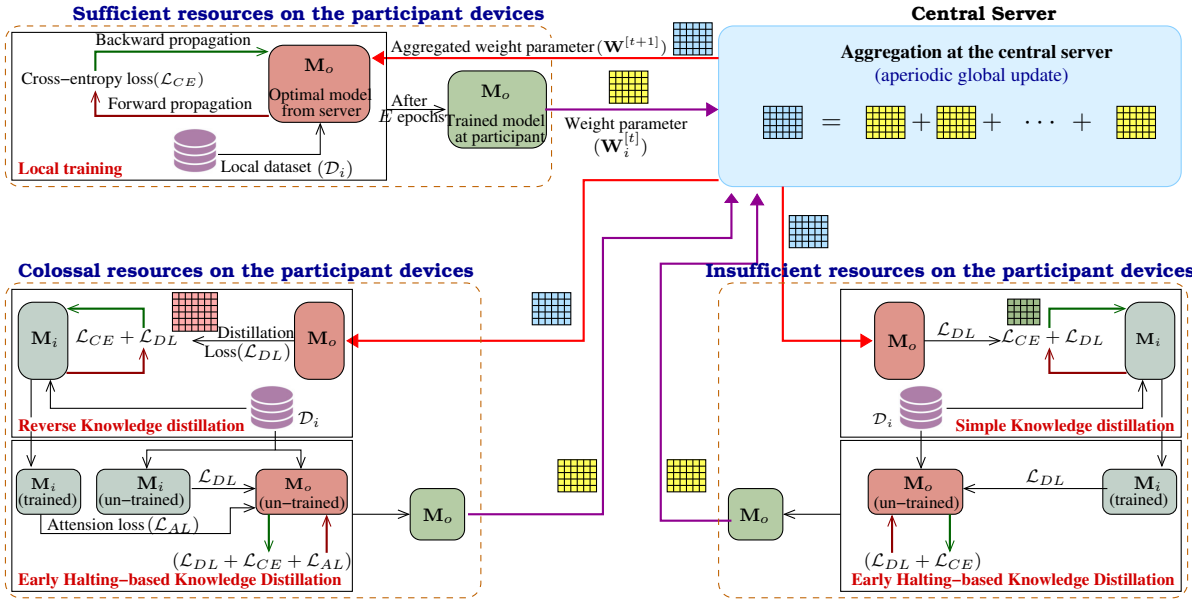


Figure 6.2: Illustration of the framework for fast federated learning technique.

6.2.1 Model selection on central server

This work assumes an FL scenario with a set $\mathcal{P}$ of N participants, $\mathcal{P} = \{\varrho_1, \varrho_2, \cdots, \varrho_N\}$ and a central server. Each participant initially sends a collaboration request to the server to show its willingness to participate in the federation. The server extracts the information of the available resources at participants, *i.e.*, memory, processing power, and network bandwidth, from the received request. Let α, β, and γ denote the weightage of processing, bandwidth, and memory, respectively, used to select the generic model for FFL technique, where $\alpha + \beta + \gamma = 1$ and $0 < \{\alpha, \beta, \gamma\} < 1$ [111, 121]. The model selection factor, denoted by **S**-factor, for a participant device is estimated as $\alpha\times$processing+ $\beta\times$bandwidth+ $\gamma\times$memory [111, 121]. The server uses unit-based normalization, which brings all resource values into the range [0,1] and estimates **S**-factor of all devices. The server uses k-means clustering on **S**-factor for dividing the

devices into categories, where $k \leq \sqrt{N}$ [122]. The server finally selects a model M_o which fit on the median category of the devices. The selected model is not arbitrarily large or small and suitable for more than half of the participant devices. The model with too large-size WPM may require a high compression ratio, which degrades the participants' accuracy with insufficient resources. The small model doesn't provide adequate accuracy on the devices. The model selection steps are shown in Procedure 6.1.

Procedure 6.1: Model selection on central server

Input: Resource information of $\mathcal{P}$ participants;
for *each participant* $\varrho_i \in \mathcal{P}$ **do**
    ϱ_i sends collaboration request to central server;
    Server extracts resources information of ϱ_i;
    Estimate **S**-factor as $\alpha\times$processing+ $\beta\times$bandwidth+ $\gamma\times$memory;
Apply k-means clustering on **S**-factors to categorized $\mathcal{P}$;
Select model which fit on the median category of the devices;
return Model M_o;

6.2.2 Model training on participant devices

Each participant $\varrho_i \in \mathcal{P}$ trains the received model M_o using local dataset $\mathcal{D}_i$. The duration of training and inference on the participant devices depend on their processing power. Similarly, the storage requirement relies upon the size of the model. In addition, the available resources on the participant devices may diverge during training and inference of the model. We consider three scenarios based on the heterogeneity of available resources on the participant devices. In the first scenario, the available resources are sufficient to train the model M_o. The other scenarios are possible when the participant device's resources are colossal or insufficient to train and perform inference on model M_o. The model training steps are shown in Procedure 6.2.

6.2.2.1 Participant devices with sufficient resources

In the first scenario, the available resources of participant devices match the requirement of resources to train and perform inference on model M_o. This scenario is illustrated

in Figure 6.2. Similarly to the FL technique discussed in [123], the training of M_o on participant ϱ_i incorporates forward and backward propagation for E local iterations. In addition, we incorporate cross-entropy loss, denoted as $\mathcal{L}_{CE}(\cdot)$, during training of M_o on ϱ_i. $\mathcal{L}_{CE}(\cdot)$ estimates the discrepancy between predicted and actual labels of instances in the local dataset $\mathcal{D}_i$.

6.2.2.2 Participant devices with colossal resources

A participant $\varrho_i \in \mathcal{P}$ with colossal resources can run a sophisticated model M_i that achieves better inference performance than small size model M_o. We use reverse KD technique with cross-entropy loss $\mathcal{L}_{CE}(\cdot)$ and distillation loss $\mathcal{L}_{DL}(\cdot)$ [124] to generate M_i from M_o, as shown in Figure 6.2. The training of M_i continued for E local epochs on $\mathcal{D}_i$. Later, the participant ϱ_i uses trained M_i for the given task.

The participant ϱ_i regenerates M_o from trained M_i for sharing the updated WPM to the central server for the next round of global aggregation. We use the KD technique [59] to transfer the knowledge from the trained M_i teacher model to the student M_o model. The logits of trained M_i become a hard target for M_o; therefore, the comparisons of their logits don't provide satisfactory performance [60]. Moreover, using un-trained M_i and trained M_i teachers for training student model M_o requires huge training parameters [61]. To overcome these issues, the FFL technique considers M_i as un-trained and trained teacher models and M_o as student model with layers sharing as shown in Figure 6.3. However, the training of M_o by using two teachers (trained M_i and un-trained M_i) requires enormous resources of the participant device.

The FFL technique *early halts* the training of the un-trained M_i model after epochs h_1 to overcome the requirement of enormous resources of the participant device, where $h_1 < E$ and E denotes the total required epochs for the training of model M_o. The early halting saves the device's resources during training of M_o and therefore fast the FL. Hereafter, the training of M_o will continue under the guidance of trained M_i.

Theorem 6.1 proves that the number of epochs h_1 to halt the training of un-trained M_i is sufficient to achieve the desired accuracy from M_o. The early halting technique uses cross-entropy loss $\mathcal{L}_{CE}(\cdot)$, attention loss $\mathcal{L}_{AL}(\cdot)$, and distillation loss $\mathcal{L}_{DL}(\cdot)$, as shown in Figure 6.3(a). The performance of M_o can be improved in the supervision of trained M_i that compares output at each epoch. The comparison is carried out using attention loss between M_o and trained M_i. The combined loss ($\mathcal{L}_{comb}(\cdot)$), which operates during the simultaneous training of M_o and un-trained M_i, is given by:

$$\mathcal{L}_{comb}(\cdot) = \begin{cases} \lambda_1\mathcal{L}^o_{CE}(\cdot) + \lambda_2\mathcal{L}_{AL}(\cdot) + \lambda_3\mathcal{L}_{DL}(\cdot) + \lambda_4\mathcal{L}^i_{CE}(\cdot), \\ \text{till training of un-trained } M_i, \\ \lambda_1\mathcal{L}^o_{CE}(\cdot) + \lambda_2\mathcal{L}_{AL}(\cdot) + \lambda_3\mathcal{L}_{DL}(\cdot). \end{cases} \tag{6.1}$$

where, λ_1, λ_2, λ_3, and λ_4 are the fractional contribution of different loss functions, $0 < \{\lambda_1, \lambda_2, \lambda_3, \lambda_4\} < 1$. We only optimize the combined loss associated with M_o, as the contribution of the loss of untrained M_i is uniform throughout the training of M_o. The early halting optimizes the following problem:

$$\min \ \mathcal{L}^o_{comb}(\cdot) \tag{6.2a}$$

$$s.t., \quad \lambda_1 + \lambda_2 + \lambda_3 = 1, \tag{6.2b}$$

$$0 < \{\lambda_1, \lambda_2, \lambda_3\} < 1. \tag{6.2c}$$

6.2.2.3 Participant devices with insufficient resources

A participant ϱ_i with insufficient resources can run a less complex model M_i, which provides inferior inference performance than M_o. We use KD approach [59] to generate small size model M_i from M_o with cross-entropy loss $\mathcal{L}_{CE}(\cdot)$ and distillation loss $\mathcal{L}_{DL}(\cdot)$. The participant uses trained M_i for the given task, as shown in Figure 6.2.

The participant ϱ_i regenerates M_o using trained M_i to share the updated WPM to

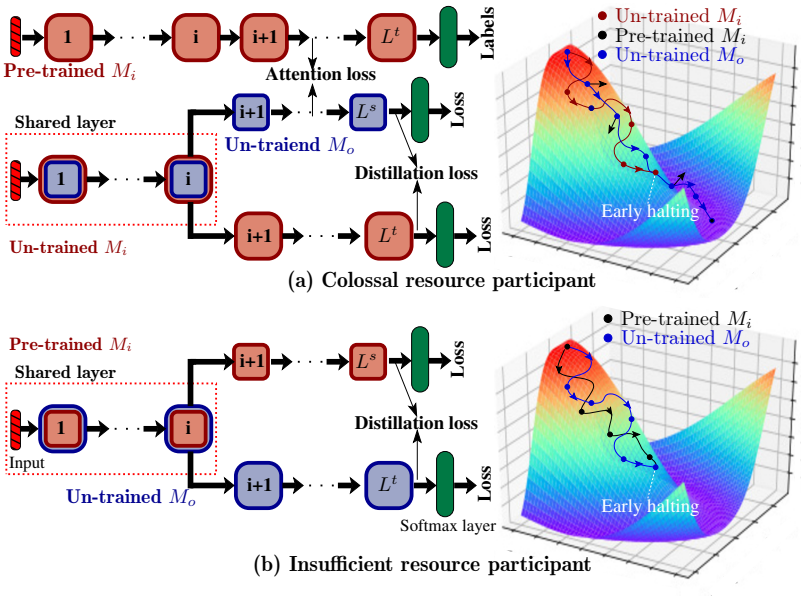


Figure 6.3: Early halting of the training of the generic model.

the central server for the next round of aggregation. We use KD to transfer the knowledge from the trained M_i teacher model to the student model M_o. Due to insufficient resources on participant ϱ_i, it is tedious to train M_o for E epochs in a limited time. We use the proposed early halting approach as shown in Figure 6.3(b). During the training of the M_o model, we do not use knowledge from the un-trained model M_i due to limited resources. The knowledge from trained M_i is used to guide the training of M_o, where training of M_o is halted at epoch h_2 ($h_2 < E$). We can obtain h_2 using Theorem 6.1.

6.2.3 Aperiodic global update

Each participant sends the WPM of the trained model to the central server for aggregation. The server may not receive the updated WPM simultaneously from all the participants if participant devices have unequal network bandwidth and processing power. The waiting for the updated WPM from all the participants at the server introduces unavoidable delays during aggregation. The FFL technique introduces aperiodic global updates at the server inspired from [125] to overcome the above problem. It allows each participant ϱ_i to aperiodically transfers its updated WPM $W_i^{[t]}$ to the server at global

Procedure 6.2: Model training on participant devices

Input: Generic model M_o, h_1 h_2, and E epochs;
for *each participant* $\varrho_i \in \mathcal{P}$ **do**
  if *(ϱ_i has sufficient resources)* **then**
    Train M_o on $\mathcal{D}_i$;
  else if *(ϱ_i has colossal resources)* **then**
    Train M_i from M_o using reverse KD and $\mathcal{D}_i$;
    /*Regenerate M_o using early halting for aggregation*/
    for *epoch* $e \leq E$ **do**
      if $e \leq h_1$ **then**
        Train M_o using pre-trained and un-trained M_i;
      else
        Train M_o using pre-trained M_i;
  else
    Train M_i from M_o using KD and $\mathcal{D}_i$;
    /*Regenerate M_o using early halting for aggregation*/
    Train M_o using pre-trained M_i for h_2 epochs;
return M_i for ϱ_i and WPM of M_o for central server;

iteration t of T time interval, where $t \leq R$ and R is the number of global iteration. The following steps are executed at iteration t:

• **Step 1:** Let us assume none of the participants has send the updated WPM before t, and the server has WPM $W^{[t-1]}$. Let k participants of set $\mathcal{P}$ have to send their WPM in threshold time $\mathcal{T}$. The η and Q_i denote the learning rate and the number of instances in dataset $\mathcal{D}_i$, respectively. The server performs aggregation at $t + \mathcal{T}$ and the updated WPM is given as:

$$W^{[t+\mathcal{T}]} = W^{[t-1]} - \eta\nabla(W') \text{ where}, W' = \sum_{i=1}^{k} \Big(\frac{Q_i}{Q_1 + Q_2 + \cdots + Q_k}\Big) W_i^{[t]}. \tag{6.3}$$

The sever sends back the updated WPM $W^{[t+\mathcal{T}]}$ to all k participants.

• **Step 2:** Let l ($l \in \{1, 2, \cdots, N-k\}$) denotes the number of participants, which have send their WPM in the interval $t+\mathcal{T}$ and $t+2\mathcal{T}$. Similar as Step 1, the server performs

aggregation at $t + 2\mathcal{T}$ to obtain updated WPM $W^{[t+2\mathcal{T}]}$ as:

$$W^{[t+2\mathcal{T}]} = W^{[t+\mathcal{T}]} - \eta\Big(\nabla(W') + \delta\nabla(W') \odot \nabla(W') \odot (W^{[t+\mathcal{T}]} - W^{[t-1]})\Big), \tag{6.4}$$

where, W' is evaluated using Equation 6.3 for l participants. δ is a variable lies in range $[0, 1]$ and symbol $\odot$ represent element-wise product. This step repeats until all the participants not send their WMP to the server. Aperiodic global update steps are shown in Procedure 6.3.

Procedure 6.3: Aperiodic global update

Input: Global aggregation interval T, time threshold $\mathcal{T}$;
Initialize: $q \leftarrow 1$;
for $j \leftarrow 1$ *to* T **do**
  if $j \leq \mathcal{T}$ **then**
    Collect WPM from the participants;
    /*Let k participants send WPM in first $\mathcal{T}$ interval*/
  Aggregate WPM from k participants using Equation 6.3;
  while $q < \frac{T}{\mathcal{T}}$ **do**
    if $q.\mathcal{T} < j \leq (q+1)\mathcal{T}$ **then**
      Collect WPM from the participants;
    Aggregate WPM using Equation 6.4;
    $q \leftarrow q + 1$;
return Aggregated WPM of M_o at central server;

Algorithm 6.1: Fast federated learning technique

Input: Set $\mathcal{P}$ of N participants with their local dataset, global iteration R;
Output: Trained model on each participant ϱ_i $(1 \leq i \leq N)$;
Call **Procedure 6.1** to select generic model at central server;
Central server shares model to the participants $\mathcal{P}$;
for R *global integration* **do**
  Call **Procedure 6.2** to train generic model using local dataset at each device;
  Call **Procedure 6.3** to send updated WPM from participant devices to the server;
Return Trained model at each participant device;

6.2.4 Deriving expression for halting epoch

This section derives the expression of halting epoch h in terms of allowable loss variance ϵ, where $h \leq E$ and E is the maximum epochs for training on the participant. Let $W_H(e)$ and $W_E(e)$ denote the WPM at epoch e ($e \leq E$) when the training of model at participant incorporates halting or non-halting mechanism, respectively. Similarly, $\mathcal{L}_{comb}(W_H(e))$ and $\mathcal{L}_{comb}(W_E(e))$ represent the combine loss (Equation 6.1) when the training of model at participant incorporate halting or non-halting mechanism, respectively. To derive the expression for ϵ, we use the following assumptions as given in [21]:
a) $\mathcal{L}_{comb}(\cdot)$ is *ρ-Lipschitz, i.e.,* $\|\mathcal{L}_{comb}(W) - \mathcal{L}_{comb}(W')\| \leq \rho\|W - W'\|$ for random weights W and $\mathbf{w}'$.
b) $\mathcal{L}_{comb}(\cdot)$ is *β-smooth, i.e.,* $\|\nabla\mathcal{L}_{comb}(W) - \nabla\mathcal{L}_{comb}(W')\| \leq \beta\|W - W'\|$.

Definition 6.1 *(Gradient Discrepancy) For epoch e and WPM W, upper bound of $\|\nabla W_H(e) - \nabla W_E(e)\|$ is defined as:*

$$\begin{array}{ll} \|\nabla\mathcal{L}_{comb}(W_H(e)) - \nabla\mathcal{L}_{comb}(W_E(e))\| = 0; & \textbf{\textit{if}}\ e \leq h, \\ \|\nabla\mathcal{L}_{comb}(W_H(e)) - \nabla\mathcal{L}_{comb}(W_E(e))\| \leq \phi(e); & otherwise \end{array}. \tag{6.5}$$

From Equation 6.5, we can obtain:

$$\phi = \frac{\sum_{e=h}^{E} \|\nabla\mathcal{L}_{comb}(W_H(e)) - \nabla\mathcal{L}_{comb}(W_E(e))\|}{E - h}.$$

Lemma 6.1 *For epoch e, where $e \in (h \leq e \leq E)$, we have:*

$$\mathcal{V}(\|W_H(e) - W_E(e)\|_{e=h}^{E}) \leq q(e), \tag{6.6}$$

where $q(e) = (1 - \eta\beta)\frac{\phi}{\beta}((1 - \eta\beta)^e + (1 - \eta\beta))$, $\eta < \frac{1}{\beta}$, $\eta > 0$, and $\beta > 0$. $\mathcal{V}(\cdot)$ denotes the variance.

Proof: To prove the lemma, we consider the following induction $\mathcal{V}(\|W_H(e) - W_E(e)\|_{e=h}^{E}) \leq$

$q(e)$ and using the rule of gradient update, $W_H(e+1) = W_H(e) - \eta\nabla\mathcal{L}_{comb}(W_H(e))$, we obtain the following expression:

$$\begin{aligned}&\mathcal{V}(\|W_H(e+1) - W_E(e+1)\|_{e=h}^{E-1}) \\ &\quad=\mathcal{V}(\|W_H(e) - \eta\nabla\mathcal{L}_{comb}(W_H(e)) - (W_E(e) - \eta\nabla\mathcal{L}_{comb}(W_E(e)))\|_{e=h}^{E}).\end{aligned}$$

Using triangle inequality [21] and property of variance: $\mathcal{V}[aX+b] = a^2\mathcal{V}(X)$ for constant a and b, we obtain:

$$\mathcal{V}(\|W_H(e+1) - W_E(e+1)\|_{e=h}^{E-1}) \leq (1-\eta\beta)^2\mathcal{V}(\|W_H(e) - W_E(e))\|_{e=h}^{E}). \tag{6.7}$$

Using considered induction, we reach to following expression:

$$\begin{aligned}&\mathcal{V}(\|W_H(e+1) - W_E(e+1)\|_{e=h}^{E-1}) \leq (1-\eta\beta)^2 q(e), \\ \implies\ &\mathcal{V}(\|W_H(e+1) - W_E(e+1)\|_{e=h}^{E-1}) = q(e+1).\end{aligned} \tag{6.8}$$

Using Equation 6.8, we obtain, $\mathcal{V}(\|W_H(e) - W_E(e)\|_{e=h}^{E}) \leq q(e)$. Hence proved. □

Theorem 6.1 *If $\mathcal{V}(\|\mathcal{L}_{comb}(W_H(e)) - \mathcal{L}_{comb}(W_E(e))\|_{e=h}^{E}) \leq \epsilon$, then relation between ϵ and e $(e = h)$ is defined as:*

$$\epsilon = \sqrt{\frac{\rho\phi(1-\eta\beta)^{e+1}}{\upsilon\eta(1-\frac{\beta\eta}{2})}} = \sqrt{\frac{2\rho\phi(1-\eta\beta)^{e+1}}{\upsilon\eta(2-\beta\eta)}}, \tag{6.9}$$

where $\eta < \frac{1}{\beta}$, $\eta > 0$, $\beta > 0$ and $\upsilon = \frac{1}{\mathcal{V}(\|W_H(e) - W_E(e)\|_{e=h}^{E})^2}$.

Proof: For an epoch $e \in (h \leq e \leq E)$, we assume $\Psi(e)$ as:

$$\Psi(e) = \mathcal{V}(\|\mathcal{L}_{comb}(W_H(e)) - \mathcal{L}_{comb}(W_E(e))\|_{e=h}^{E}). \tag{6.10}$$

Using β-smoothness of the loss function and property discussed in [126]: $\mathcal{L}_{comb}(W) \leq$

$\mathcal{L}_{comb}(W') + \nabla\mathcal{L}_{comb}(W')^T(W - W') + \frac{\beta}{2}\|W - W'\|^2$, we get the following:

$$\begin{aligned}
&\mathcal{V}(\|\mathcal{L}_{comb}(W_H(e+1)) - \mathcal{L}_{comb}(W_H(e))\|_{e=h}^{E-1}) \\
&\quad\leq \mathcal{V}(\|\nabla\mathcal{L}_{comb}(W_H(e))^T(W_H(e+1) - W_H(e))\|_{e=h}^{E}), \\
&\quad\leq -\eta^2\Big(1 - \frac{\beta\eta}{2}\Big)\mathcal{V}(\|\nabla\mathcal{L}_{comb}(W(e))\|_{e=h}^{E})^2. \qquad (6.11)
\end{aligned}$$

From Equation 6.10, we have following expressions:

$$\begin{aligned}
\Psi(e) &= \mathcal{V}(\|\mathcal{L}_{comb}(W_H(e)) - \mathcal{L}_{comb}(W_E(e))\|_{e=h}^{E}), \\
\Psi(e+1) &= \mathcal{V}(\|\mathcal{L}_{comb}(W_H(e+1)) - \mathcal{L}_{comb}(W_E(e+1))\|_{e=h}^{E-1}).
\end{aligned}$$

Further, using the expression derived in Equation 6.11, we get:

$$\Psi(e+1) \leq \Psi(e) - \eta^2\Big(1 - \frac{\beta\eta}{2}\Big)\mathcal{V}(\|\nabla\mathcal{L}_{comb}(W(e))\|_{e=h}^{E})^2. \qquad (6.12)$$

Assuming independent $W_H(\cdot)$ and $W_E(\cdot)$, we have:

$$\begin{aligned}
\Psi(e) &= \mathcal{V}(\|\mathcal{L}_{comb}(W_H(e)) - \mathcal{L}_{comb}(W_E(e))\|_{e=h}^{E}), \\
&\leq \mathcal{V}(\|\nabla\mathcal{L}_{comb}(W_H(e))^T(W_H(e) - W_E(e))\|_{e=h}^{E}), \\
\frac{\Psi(e)}{\mathcal{V}(\|W_H(e) - W_E(e)\|_{e=h}^{E})} &\leq \mathcal{V}(\|\nabla\mathcal{L}_{comb}(W_H(e))\|_{e=h}^{E}).
\end{aligned}$$

Using value of $\mathcal{V}(\|\nabla\mathcal{L}_{comb}(W_H(e))\|_{e=h}^{E})$ in Equation 6.12, we get:

$$\Psi(e+1) \leq \Psi(e) - \upsilon\eta^2\Big(1 - \frac{\beta\eta}{2}\Big)\Psi(e)^2. \qquad (6.13)$$

Since, $\Psi(e+1)\Psi(e) > 0$, thus, it would not harm the inequality of Equation 6.13 upon

division on both side.

$$\frac{1}{\Psi(e+1)} - \frac{1}{\Psi(e)} \geq \frac{\upsilon\eta\Big(1-\frac{\beta\eta}{2}\Big)\Psi(e)}{\Psi(e+1)} \geq \upsilon\eta^2\Big(1-\frac{\beta\eta}{2}\Big). \tag{6.14}$$

$$\frac{\Psi(e)-\Psi(e+1)}{\Psi(e+1)\Psi(e)} = \frac{\mathcal{V}(\|\mathcal{L}_{comb}(W_H(e)-\mathcal{L}_{comb}(W_E(e))\|_{e=h}^{E})}{\Psi(e+1)\Psi(e)} - \frac{\mathcal{V}(\|\mathcal{L}_{comb}(W_H(e+1)-\mathcal{L}_{comb}(W_E(e+1))\|_{e=h}^{E-1})}{\Psi(e+1)\Psi(e)}.$$

Using ρ-*Lipschitz* property and Lemma 6.1, we have:

$$\frac{\Psi(e)-\Psi(e+1)}{\Psi(e+1)\Psi(e)} \geq \frac{\rho\eta\phi(1-\eta\beta)^{e+1}}{\Psi(e+1)\Psi(e)}. \tag{6.15}$$

Using Equation 6.10 and assuming $\epsilon > 0$, we get:

$$\mathcal{V}(\|\mathcal{L}_{comb}(W_H(e+1)) - \mathcal{L}_{comb}(W_E(e+1))\|_{e=h}^{E-1}),$$

$$\mathcal{V}(\|\mathcal{L}_{comb}(W_H(e)) - \mathcal{L}_{comb}(W_E(e))\|_{e=h}^{E}) \leq \epsilon^2. \tag{6.16}$$

$$\frac{1}{\Psi(e+1)\Psi(e)} \geq \frac{1}{\epsilon^2}. \tag{6.17}$$

Using Equation 6.17 and Equation 6.15, we obtain:

$$\frac{\Psi(e)-\Psi(e+1)}{\Psi(e+1)\Psi(e)} \geq \frac{\rho\eta\phi(1-\eta\beta)^{e+1}}{\epsilon^2}. \tag{6.18}$$

Using Equation 6.14 and Equation 6.18 and taking limiting condition:

$$\upsilon\eta^2\Big(1-\frac{\beta\eta}{2}\Big) = \frac{\rho\eta\phi(1-\eta\beta)^{e+1}}{\epsilon^2}.$$

Since, $\epsilon > 0$, taking positive value of square root, we obtain:

$$\boxed{\epsilon = \sqrt{\frac{\rho\phi(1-\eta\beta)^{e+1}}{\upsilon\eta(1-\frac{\beta\eta}{2})}} = \sqrt{\frac{2\rho\phi(1-\eta\beta)^{e+1}}{\upsilon\eta(2-\beta\eta)}}.} \tag{6.19}$$

Hence, proved. □

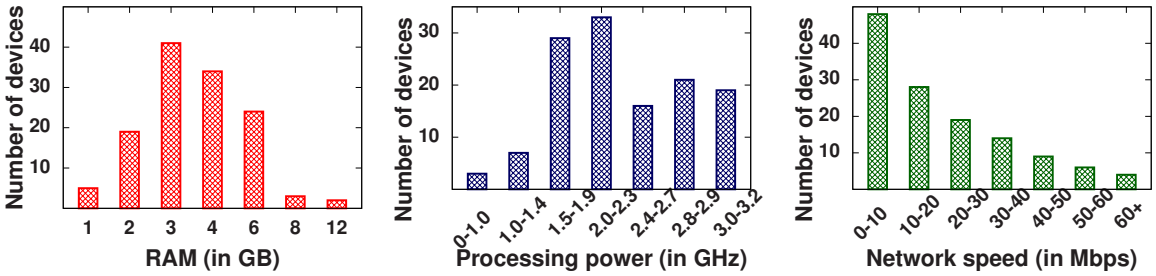
6.3 Real-world study

This section presents a real-world study to evaluate the feasibility and performance of the proposed work. Firstly, we describe the specifications of participants devices and central server used in the study. Next, we discuss the task of study, followed by the challenges observed during the study.

6.3.1 Participants devices and central server in real-world study

In this study, we recruited 128 student volunteers as the participants and a central server located in the institute to execute the FL operations. The smartphones of the volunteer work as participant devices which have different specifications and brands, including Samsung, Redmi, Realme, Honor, Apple iPhone, Oppo, *etc.* Due to the different specifications and brands, the participant devices have heterogeneous resources, *i.e.*, memory and processing power. Random Access Memory (RAM) on these smartphones lie in the range of 1 GB to 12 GB, as illustrated in Figure 6.4(a). However, the average availability of RAM is approximate $30\% - 40\%$ due to pre-installed applications and operating systems on smartphones. Similarly, the processing power or Central Processing Unit (CPU) clock speed lies in different ranges, as illustrated in Figure 6.4(b). Due to the COVID-19, most of the students are at homes in distinct locations; thus, network bandwidth between participants and the central server located at the institute possesses a high level of heterogeneity. Figure 6.4(c) illustrates the network bandwidth in the ranges. Besides, from Figure 6.4, we can observe that maximum volunteers have

RAM of 3 GB (41 volunteers), CPU clock speed in the range $2.0 - 2.3$ GHz (33 volunteers), and network speed < 10 Mbps (48 volunteers). Further, the used central server is a Dell server with an Intel Dual Xeon processor with 256GB RAM operating over a Gigabit Ethernet connection.



(a) Memory. (b) Processing power. (c) Bandwidth.

Figure 6.4: Participants devices and central server in real-world study. (a) RAM specification, (b) Processing power, and (c) Bandwidth.

6.3.2 Task of study: Locomotion Mode Recognition

This study considered the task of Locomotion Mode Recognition (LMR) using participant devices. LMR recognizes the following locomotion modes bicycle, car, bus, auto-rickshaw, bike, and train, using multiple sensors, including accelerometer, gyroscope, and magnetometer. LMR helps in several aspects such as estimating travel time, optimizing traffic flow, adequate journey planning, estimating travel expenses, *etc* [5, 127]. The study considered the LMR task because the FFL technique does not require sharing of locomotion data, which maintains the data privacy and overcomes the limitation of the limited bandwidth.

6.3.3 Challenges observed during study

The first challenge we encountered during the study was the *inconsistency in the availability* of the participant devices. This inconsistency is a matter of the fact that the student volunteers are at distinct and remote locations, where they face frequent connection and disconnection from the network. Next, the restrictive movement due to

pandemic COVID-19 created another challenge of *data scarcity*. The volunteers collected only a few samples of locomotion modes data. In addition, some volunteers collected data for a sub-set of classes only. Further, we encountered the challenge of variation in the sensory data instances due to *different brands of smartphones* used by the participants in the real-world study. For example, the sensory instances of Samsung smartphones were different to that of Xiaomi. Finally, the participants are at distinct locations; thus, they faced the interruption in power supply. This interruption adversely hampered the *remaining battery of smartphones*, and participants left the training.

6.4 Performance Evaluation

This section describes the datasets, baseline techniques, implementation details, and validation metrics used to evaluate the performance of the proposed FFL technique. In this section, we carry out the experimental evaluation to validate the performance of the FFL technique.

6.4.1 Datasets

In this work, we considered the collected LMR dataset and an existing SHL [55] dataset to evaluate and compare the performance of FFL technique.

6.4.2 Baseline techniques

We considered the existing techniques [52, 54] as baselines, noted as Baseline1 and Baseline2, to evaluate the performance of the proposed FFL technique. Baseline1 [52] had selected a lightweight model for FL, which work successfully even on insufficient resources participant devices. However, such a lightweight model does not give high accuracy in a heterogeneous scenario where some participants have sufficient or colossal resources. The proposed FFL technique considers this issue and resizes the selected model based on the available resources of participant devices. Baseline2 [54] proposed

the mechanism to disjoint the local training and global aggregation operations in FL. The mechanism allowed the participants to transfer their WPM upon completion of local training and immediately get the response of aggregated WPM. Baseline2 supports asynchronous aggregation where the server does not wait for the WPM of all participants. Baseline2 allowed the participants to transfer their WPM upon completion of local training and immediately get the response of aggregated WPM. The aggregated WPM in asynchronous aggregation creates WPM staleness issue.

6.4.3 Implementation details

We used the Deep Neural Networks (DNN) DeepZero discussed in [127] during experimental evaluation. We selected DeepZero because it computed automated deep learning features and combined them with hand-crafted features to achieve high-order accuracy and introduced some level of parallelism. We have implemented the FFL technique using Python language with Tensorflow and Keras libraries. FFL incorporated the functional API of Keras to implement DeepZero. To estimate fractional variables (λ_1, λ_2, and λ_3) in Equation 6.2, we adopt differential evolution technique [128] on different loss functions, *i.e.,* cross-entropy loss, distillation loss, and attention loss. We selected the window size of 20 to reduce the length of dataset instances from 6000 to 300 during pre-processing. Later, we perform the random and disjoint partitioning of LMR dataset into training and testing sub-datasets of ratio 80 : 20 using *sklearn.model_selection.train_test_split*() in Sklearn model selection [129].

We selected 100 participants from 128 volunteers depending upon their availability and willingness to participate in FL. All the participants registered their devices with the central server located in the institute, where unique identifiers are assigned to them after authentication. All the authorized participants performed the FL operations on commonly agreed and pre-specified schedules. We used the push style of Internet-based communication to get updated WPMs and transfer aggregated WPMs to the

participants. We randomly partitioned training and testing sub-datasets of SHL and LMR in 100 disjoint parts and provided a unique data portion to each volunteer. We develop a smartphone application, which incorporates Python libraries of Tensorflow lite and Keras. Each volunteer needs to install the application to successfully participate in FL, where training and inference of a lightweight (or large-size) version of DeepZero are performed on smartphone.

6.4.4 Validation metrics

In this work, we used the following standard classification metrics to evaluate and compare the performance of the FFL technique: F_1-score, accuracy, and leave-one-out test validation. Let a given dataset consists of a set of $\mathcal{A}$ classes, and $|\mathcal{A}|$ represents the number of classes. Let TP_i, TN_i FP_i, and FN_i are the true positive, true negative, false positive, and false negative counts of a class $i \in \mathcal{A}$, respectively. The *accuracy* metric is computed as:

$$\frac{1}{|\mathcal{A}|}\sum_{i=1}^{|\mathcal{A}|}\frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i}. \tag{6.20}$$

Next, the F_1-*score* is computed as:

$$\frac{1}{|\mathcal{A}|}\sum_{i=1}^{|\mathcal{A}|}\frac{2 \times TP_i}{2 \times TP_i + FP_i + FN_i}. \tag{6.21}$$

We finally consider the *leave-one-out test validation* metric that trains the model for all class labels except for one randomly chosen class label. However, during testing, the unseen class label is also supplied for predicting the output. Thus, it evaluates the performance of the classifier for one unseen class label.

6.4.5 Experimental results

In this section, we present the experimental results of the FFL technique on aforementioned datasets.

6.4.5.1 Impact of heterogeneous devices and networking resources

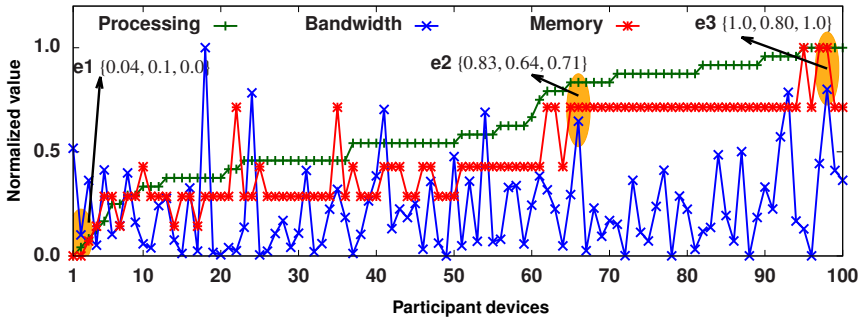
This experiment aims to select a generic model that can be easily accommodated on the participating devices for the FFL technique. The experiment considered the real-world study where the participant devices have heterogeneous resources (processing power, bandwidth, and memory).

Figure 6.5(a) illustrates the normalized values of resources of 100 participants devices. We used unit-based normalization, which brings all values into the range [0,1]. The devices are arranged in ascending order of their proceeding capacity then bandwidth and memory resources are arranged accordingly. Ellipses **e1**, **e2**, and **e3** in Figure 6.5(a) illustrate the normalized values of resources of participant devices, where **e1** and **e3** are the devices with least and colossal resources, respectively. We estimate **S**-factor using normalized values of resources of participant devices in Procedure 6.1. Since most of the smartphones of participants have sufficient memory to run the generic model, we set its weightage to the least value ($\gamma = 0.1$) while selecting the model. Figure 6.5(b) illustrates **S**-factor when $\{(\alpha, \beta, \gamma)\}$ are $\{(0.7, 0.2, 0.1)\}$ and $\{(0.5, 0.4, 0.1)\}$.

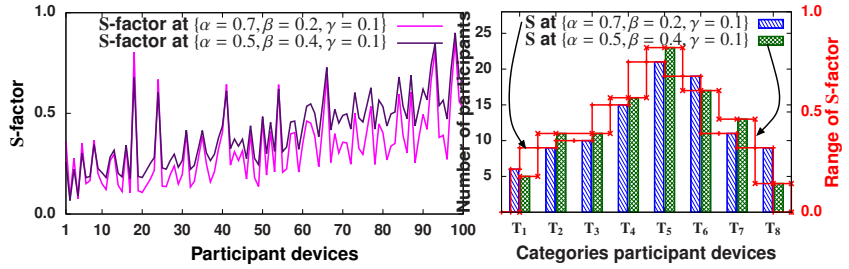
Further, we arrange the estimated **S**-factors in ascending order and divided the participants into eight categories $\mathbf{T_1}, \mathbf{T_2}, \cdots, \mathbf{T_8}$, based on the possible configurations of the models. Figure 6.5(c) illustrates the number of participants and the range of **S**-factor for each category when $\alpha = 0.7, \beta = 0.2$, and $\gamma = 0.1$. We observe from the result that the number of participants in $\mathbf{T_5}$ is highest for both **S**-factors. A generic model for $\mathbf{T_5}$ works on the maximum number of devices. Moreover, the model is also suitable for $\mathbf{T_6}$ to $\mathbf{T_8}$ because they have more than the required resources. The selection of a model for $\mathbf{T_5}$ also work on $\mathbf{T_1} - \mathbf{T_4}$ with some level of compression.

Observation: The first observation from the result is that the real-world study consists high level of heterogeneity, where all the resources of some devices are either insufficient or colossal. The next observation is that the networking resource (bandwidth) is independent of the device resources. Thus, while selecting the generic model, we need to

consider devices and networking resources simultaneously. Specially, we observed that the processing and bandwidth are the essential factors for selecting the generic model. Finally, we observed that a selected generic model must satisfy the requirement of insufficient or colossal resources devices.



(a) Devices and networking resources.



(b) **S**-factor. (c) Categories of participant devices.

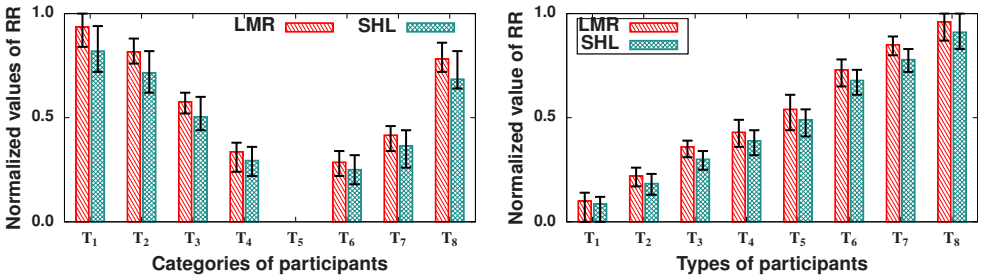
Figure 6.5: Illustration of devices and networking resources and categories of participant devices.

6.4.5.2 Training of the model on each participant device

The objective of this experiment is to highlight the requirement of the resources for resizing and training of the model based on the categories of devices. Figure 6.6(a) illustrates the normalized values of required device resources for resizing the model using LMR and SHL datasets. $\mathbf{T_5}$ category of devices directly used the selected model and therefore did not require resources for resizing. To fit the model on other devices, $\mathbf{T_1}$-$\mathbf{T_4}$ and $\mathbf{T_6}$-$\mathbf{T_8}$ categories of devices need to compress and enlarge the model, respectively, as discussed in Algorithm 6.1. Devices of $\mathbf{T_1}$ category needs high compression as compared to the enlarge the model for $\mathbf{T_8}$. Therefore, $\mathbf{T_1}$ needs more resources than $\mathbf{T_8}$.

Similarly, Figure 6.6(b) illustrates the normalized values of required device resources for training of resized model using FFL technique with LMR and SHL datasets. $\mathbf{T_1}$ to $\mathbf{T_8}$ increase the size of the models and therefore require more resources. Moreover, $\mathbf{T_6}$-$\mathbf{T_8}$ require high resources because they used two teacher models during training, as shown in Figure 6.6. The results also illustrate that the LMR dataset takes more resources during resizing and training than SHL datasets because the LMR has more instances than the SHL.

Observation: The early halting helps to reduce the requirement of resources during training. Therefore, FFL successfully trains the models on insufficient resource devices by using early halting.



(a) Resizing of the model. (b) Training of the model.

Figure 6.6: Illustration of the requirement of resources for resizing and training of the model based on the category of participant devices, where RR is Required Resources.

6.4.5.3 Performance of FFL technique

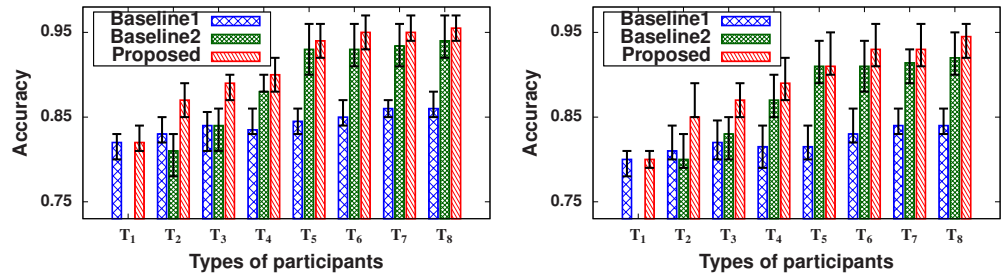
The objective of this experiment is to evaluate and compare the performance of the FFL technique with existing baseline techniques (Baseline1 [52] and Baseline2 [54]). The results used the generic model selected in previous result, collected LMR and SHL datasets, and validation metrics as given in Section 6.4.4.

Figure 6.7(a1) illustrates the accuracy of the techniques using Equation 6.20 and LMR dataset. It shows the FFL gives higher accuracy than Baseline1 and Baseline2 for all categories of devices. It is because Baseline1 used a lightweight model for all participant devices that inefficiently utilize the available resources of the devices. Sim-

ilarly, Baseline2 focused only on aggregation and did not consider the limitation of the resources on the devices. We also observed that FFL gives almost equal accuracy as Baseline1 and Baseline2 for $\mathbf{T_1}$ and $\mathbf{T_5}$, respectively. The FFL resizes the model based on the limited resources of $\mathbf{T_1}$ and directly uses the generic model for $\mathbf{T_5}$. The higher category of devices have more resources and therefore gives high accuracy in all the techniques, as shown in the result. The results in Figure 6.7(a2) give less accuracy than Figure 6.7(a1) for all the techniques and categories of participant devices because of the SHL dataset, which has imbalanced classes. In addition, the SHL dataset consists of more classes than the LMR dataset. This is another reason for less accuracy.

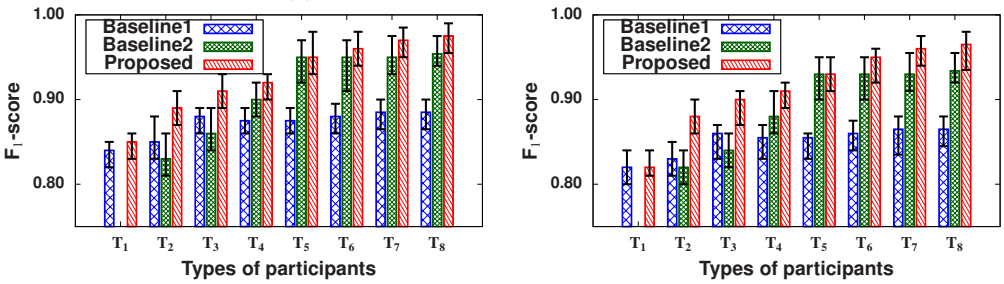
Figure 6.7(b) illustrates the F_1-score as given in Equation 6.21, for validating the preference of the techniques. The class distribution in the collected LMR dataset is similar; therefore, the true positives and true negatives are important and false negatives, and false positives are not crucial. The results in Figure 6.7(b1) illustrate the similar behaviour as shown in Figure 6.7(a1). However, it is not true for Figure 6.7(b2), where false negatives and false positives are crucial and give less accuracy. Finally, we performed experiments by using a leave-one-out validation metric where instances of one class are not considered during training. Figure 6.7(c) illustrates the results of the leave-one-out validation metric where instances of one class are not considered during training. The results illustrate that difference of the accuracy in Figure 6.7(a1) and Figure 6.7(c1) is less for colossal resources devices $\mathbf{T_8}$ then insufficient resource devices $\mathbf{T_1}$. This is because the colossal resource devices train more successfully than insufficient resource devices.

Observation: All the techniques give less accuracy and F_1-score while using the dataset with imbalanced class labels. It affects more on insufficient resource devices. We conclude from this observation that if the dataset has imbalanced class labels then the selection of the generic model must be favorable for insufficient resource devices.



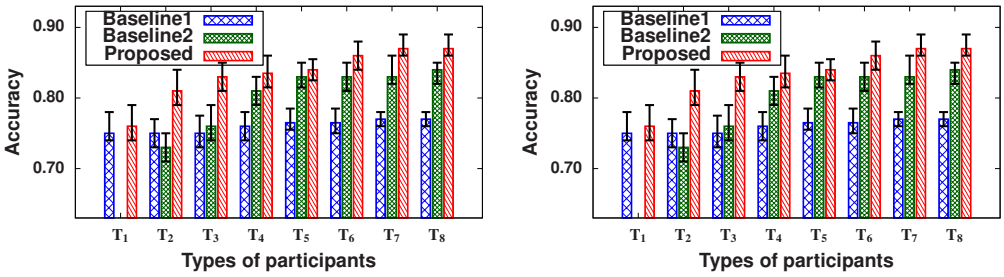
(a1) LMR dataset. (a2) SHL dataset.

(a) Estimated accuracy using Equation 6.20.



(b1) LMR dataset. (b2) SHL dataset.

(b) Estimated F_1-score using Equation 6.21.



(c1) LMR dataset. (c2) SHL dataset.

(c) Estimated accuracy using leave-one-out test.

Figure 6.7: Illustration of the accuracy, F_1-score, and leave-one-out test of the FFL and baseline techniques [52, 54].

6.4.5.4 Impact of early halting on performance and earliness

The objective of this experiment is to illustrate the impact of the early halting on the performance of participants with colossal and insufficient resources. We also demonstrate the reduction in FLoating-point OPerations (FLOPs) using the early halting.

We considered the participants of category $\mathbf{T_7}$ to describe the impact of the early halting mechanism on colossal resources participants. Table 6.1 illustrates that significant improvement in accuracy and F_1-score of M_o is observed up to 60 epochs of

simultaneous training with un-trained large-size M_i (model on participant ϱ_i) under the guidance of trained M_i. After that, we observe a minor improvement in the accuracy and F_1-score of M_o. However, the required resources for further training increases sharply. It indicates that the training of un-trained M_i with M_o do not improve performance despite consuming enormous resources (FLOPs). Therefore, we can halt the training of un-trained M_i at 60 epochs. Next, Table 6.2 illustrates the fractional contribution of different loss functions (given in Equation 6.2) on the performance of M_o for participant category $\mathbf{T_6}$ and $\mathbf{T_7}$. The result illustrates that the increase in resources of participant devices and reduction in the dataset's size leads to improved accuracy and F_1-score of M_o. It also decreases the contribution of distillation loss (λ_3). When the difference between the size of M_o and M_i is large, attention loss and cross-entropy loss help to reduce the deviation of M_o training due to random initialization un-trained M_i. Thus, contribution of un-trained M_i via distillation loss (λ_3) is reduced to improve performance of M_o.

Table 6.1: Impact of early halting on participants of $\mathbf{T_7}$ category.

Epochs	40	50	**60**	70	80	90	100
Accuracy (in %)	82.51	87.02	**93.08**	93.37	93.53	93.97	94.54
F_1-score (in %)	84.21	88.78	**94.73**	95.03	95.17	95.23	94.83
FLOPs ($\times 10^{13}$)	2.53	2.77	**2.93**	3.09	3.29	3.56	3.73

Table 6.2: Fractional contributions (λ_1, λ_2, and λ_3) of different loss functions on participants with colossal resources of categories $\mathbf{T_6}$ and $\mathbf{T_7}$ using LMR and SHL datasets.

Category	**Dataset**	**Fractional weights**			**Accuracy**	**F_1-score**
		λ_1	λ_2	λ_3		
$\mathbf{T_6}$	LMR	0.4739	0.3453	0.1808	92.81%	93.93%
	SHL	0.4817	0.3641	0.1542	92.43%	93.17%
$\mathbf{T_7}$	LMR	0.4953	0.3761	0.1286	93.08%	94.73%
	SHL	0.5122	0.3944	0.0934	92.72%	94.31%

Further, we considered participant type $\mathbf{T_3}$ to study the role of early halting on

insufficient resources participants. Table 6.3 depicts the performance improvement up to 70 epochs during training of M_o under the guidance of lightweight M_i is rapid. After that, the performance improvement is low, but resource consumption (FLOPs) is high. Thus, we halt the training of M_o for participant type $\mathbf{T_3}$ at 70 epochs.

Table 6.3: Impact of early halting on participants of $\mathbf{T_3}$ category.

Epochs	40	50	60	**70**	80	90	100
Accuracy (in %)	63.82	72.82	81.07	**87.92**	88.13	88.74	89.14
F_1-score (in %)	67.17	77.68	85.74	**90.03**	90.19	90.63	91.20
FLOPs ($\times 10^{11}$)	1.17	1.23	1.41	**1.54**	1.71	1.82	1.91

Observation: An interesting observation from the result is that the early halting of training preserves sufficient resources on the participant devices. Therefore, the selection of halting epoch should be made wisely to preserve resources, speed up the training, and achieve adequate performance.

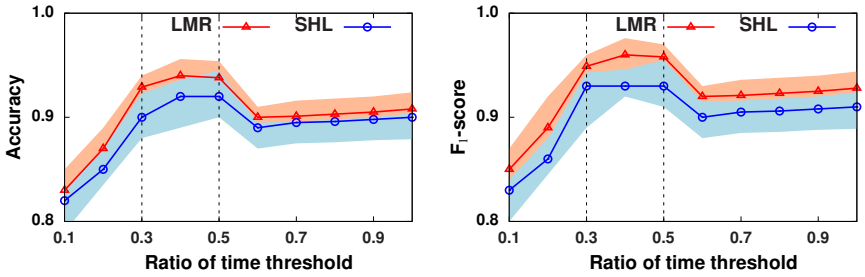
6.4.5.5 Impact of aperiodic global update

This experiment aims to determine the impact of aperiodic global updates on the performance of the FFL technique. We estimated the ratio of time threshold $\mathcal{T}$ with global iteration time interval to determine its impact on the performance using LMR and SHL.

Figure 6.8(a) illustrates the accuracy increases with the ratio of $\mathcal{T}$ and T up to 0.5 and decreases thereafter for the collected LMR dataset. It is because at a lower value of $\mathcal{T}$ only a few participants are involved in the intermediate aggregations, which results in inferior quality of aggregated WPM. On the other hand, the increment in $\mathcal{T}$ allowed more devices to participate in intermediate aggregation, which generate precise aggregated WPM. However, beyond the ratio of 0.5, the number of participants involved in the intermediate aggregation increases, but the aggregation is performed only once between two global iterations. The participant devices with colossal resources get the

updated WPMs quickly but wait for the next global iteration. It results in performance compromise of colossal resource participants. Figure 6.8(a) also illustrates that optimal value of $\mathcal{T}$ lies in the range of 0.3 to 0.5. $\mathcal{T}$ also depends upon the available network bandwidth. If the average bandwidth of all the participants is high then the optimal value of $\mathcal{T}$ is closer to 0.3 and closer to 0.5 when the average bandwidth is low. Similarly, Figure 6.8(b) illustrates that SHL gives low F_1-score than the LMR dataset for a given ratio of $\mathcal{T}$ and T. It is because SHL has imbalanced classes, as we have discussed Section 6.4.5.3.

Observation: An interesting observation from the result is that the value of the time threshold played a decisive role to ensure a high-order performance of the FFL technique. $\mathcal{T}$ should not exceed 0.5 *as it restricts the multi-round intermediate aggregation. This restriction reduced the role of aperiodic global updates and hence compromised the performance of the FFL technique.*



(a) Accuracy. (b) F_1-score.

Figure 6.8: Impact of ratio of time threshold with global iteration time interval.

6.5 Conclusion

This chapter proposed a FFL technique to train a model for a given task at the participant devices using its local dataset. Unlike the existing work, the FFL technique trained the model on participant devices with resource heterogeneity. We first presented an approach to obtain the generic model, which is not arbitrarily large or small and suitable for most participant devices. Next, we proposed an early halting approach

for faster training of the resized model, which fits the insufficient and colossal resource devices. We finally proposed an aperiodic global update approach for aggregation of WPM in the presence of unequal bandwidth and processing power of devices. We also did a real-world study to evaluate the feasibility and performance of the proposed work. We found the following conclusions from this work: federated learning works successfully only when the device and networking resources are considered simultaneously; generic model must satisfy insufficient and colossal resource devices; if the dataset is imbalanced, then the selection of the generic model must be favorable for insufficient resource devices.