 http://dx.doi.org/10.1109/GLOBECOM46510.2021.9685095

BePOCH: Improving Federated Learning Performance in Resource-Constrained Computing Devices

Lenart Ibraimi
Max van der Stoel Institute
South East European University
North Macedonia
l.ibraimi@seeu.edu.mk

Mennan Selimi
Max van der Stoel Institute
South East European University
North Macedonia
m.selimi@seeu.edu.mk

Felix Freitag
Department of Computer Architecture
UPC BarcelonaTech
Spain
felix@ac.upc.edu

***Abstract*—**
Inference with trained machine learning models is now possible with small computing devices while only a few years ago it was run mostly in the cloud only. The recent technique of Federated Learning offers now a way to do also the training of the machine learning models on small devices by distributing the computing effort needed for the training over many distributed machines. But, the training on these low-capacity devices takes a long time and often consumes all the available CPU resource of the device. Therefore, for Federated Learning to be done by low-capacity devices in practical environments, the training process must not only target for the highest accuracy, but also on reducing the training time and the resource consumption. In this paper, we present an approach which uses a dynamic epoch parameter in the model training. We propose the BePOCH (Best Epoch) algorithm to identify what is the best number of epochs per training round in Federated Learning. We show in experiments with medical datasets how with the BePOCH suggested number of epochs, the training time and resource consumption decreases while keeping the level of accuracy. Thus, BePOCH makes machine learning model training on low-capacity devices more feasible and furthermore, decreases the overall resource consumption of the training process, which is an important aspect towards greener machine learning techniques.

***Index Terms*—Federated Learning; medical dataset; healthcare applications**

I. Introduction

Inference with trained machine learning models is now possible on the smallest computing devices while only a few years ago it was mostly run in the cloud only [1]. While powerful computers in the cloud are needed to train large machine learning models with huge amounts of data, the recent technique of Federated Learning (FL) allows to distribute the computing effort of the model training over many distributed machines with lower capacity [2]. In addition, Federated Learning does not transfer the local training data, which opens new opportunities to applications which train with private data, such as sensitive medical data [3].

The training of machine learning models is a compute-intensive task. If available, GPUs instead of CPUs are used for better performance. On low-capacity devices such as single-board-computers (SBCs), training takes a much longer time than on higher end devices [4] [5]. In addition, often all cores of the CPU are consumed during the training process. This fact reduces the QoS (Quality of Service) which machine learning model training on low-capacity devices can deliver, such as having large training times. As a consequence, training of machine leaning models on low-capacity devices may not be feasible for applications that have time constraints. Furthermore, computing nodes in user environments are often not dedicated exclusively to the machine learning application. Instead, they are used for instance as home servers providing to the user other applications as well. Since training can severely affect the user's QoE (Quality of Experience) perceived from other applications, it is of interest to avoid any unnecessary training epochs.

For Federated Learning be used in realistic user environments where the machine learning application co-exists with other applications on the low-capacity device, novel optimizations for efficient training will need to come into play: the training process must not only target for the highest accuracy, but also on reducing the training time and the resource consumption of the device [6] [7].

In this paper we introduce a novel approach of using a dynamic epoch parameter in the model training. We propose the BePOCH (Best Epoch) algorithm to identify what is the best number of epochs per training round in federated machine learning model training taking into account the CPU consumption of the devices. We show in experiments that when applying the BePOCH suggested number of epochs, the training time and resource consumption decreases while keeping the level of accuracy.

The main contributions of the paper are:

1) We present the concept of the dynamic epoch parameter to reduce the resource consumption and training time for Federated Learning in environments with low-capacity devices based on empirical evidences.
2) We propose the BePOCH algorithm and show with experiments that train medical datasets with Federated Learning conducted in a Raspberry Pi testbed how the BePOCH dynamic epoch selection can optimize the training process.

The insights gained from the work presented in this paper reveal two initial benefits: First, the potential of the proposed dynamic epoch parameter for leading to less resource con-

sumption in the machine learning model training process. Secondly, by reducing the training time and CPU usage on the client nodes, the feasibility from reducing the training time and user acceptance from a higher QoE to conduct machine learning model training on low-capacity devices increases.

The rest of the paper is organized as follows. In Section II we describe the related work. Section III presented the system model. Section IV describes the experimental setup, the experiments carried out and the obtained results. Section V concludes and points to future work.

II. Related work

In this section, we review selected work related to the application of Federated Learning on low-capacity devices and works in which the research explored mechanisms for the preservation of the privacy of data, a critical requirement when training concerns personal health or medical data.

The work of Y. Gao et al [8] makes an empirical evaluation of two different state-of-art machine learning techniques, namely split neural networks (SplitNN) and Federated Learning. For and end-to-end evaluation, a variety of datasets, different model architectures, multiple clients and various performance metrics was considered. The learning performance was assessed for two types of distributed data, imbalanced and Non-IID (Independent and Identically Distributed) data. Model training was done on Raspberry Pi devices, where the CPU consumption, memory usage, communication overhead and training time was measured. From the experiments the authors conclude that FL overall perform better in comparison with SplitNN, because of the lower communication overhead of FL. Since Y. Goa et al. conducted some experiments with single Raspberry Pi devices, we apply some of the insights from their work in our experimentation to set up the baseline system. Specifically, we apply as reference epoch values the configuration of 1 and 5 epochs which these authors used.

The privacy aspect in the training with medical data was a primary issue for the proposition of Dopamine [9], a Federated Learning system to be trained with medical data. The work studies different machine learning and privacy preserving techniques. Medical data is used to train deep neural networks (DNN) for medical diagnosis. DNNs are trained on distributed datasets by using Federated Learning with differentially-private stochastic gradient descent (DPSGD), which in combination with secure aggregation, was shown to provide a better trade-off between differential privacy (DP) guarantee and DNN's accuracy.

A combination of blockchain technology and Federated Learning techniques was proposed in [10] for training machine learning models without exposing or leaking the actual data. The authors consider the case of rapid data generation from connected devices in an Industrial Internet of Things (IIoT) scenario. The authors argue that leakage of such data is a serious concern in industrial environments. For prevention they firstly designed a blockchain-empowered secure data sharing architecture. Then the data sharing problem is dealt by incorporating privacy-preserving mechanisms into Federated Learning. From their results they found that privacy of the data is maintained with the blockchain support while sharing the model in the FL process.

In [11] the BlockFL system is proposed, which introduces another blockchain-combined Federated Learning approach. Mobile devices' locally learned model updates are exchanged by with the support of a distributed ledger. Their architecture therefore does not use a central aggregator component. The authors do an end-to-end investigation about the latency of the learning completion in BlockFL. In their work, multiple low-capacity devices, in this case mobile devices, are used to train the model locally while leveraging the distributed ledger based on blockchain technology.

In the paper of Jonathan Passerat-Palmbach et al. [12], the authors purpose a blockchain-orchestrated Federated Learning architecture for healthcare consortia. Their use case presents the contribution of data to improve a machine learning model while privacy and proper accounting is assured. The architecture delivers privacy preserving audit trails that log events in the network without revealing identities.

In the work of H. Kim et al. [13] the blockchain is an enabler to use private distributed data in Federated Learning. The authors present a system architecture and describe how the use of the blockchain integrated in the Federated Learning protocol can improve privacy while using distributed data the training of the machine learning models.

The work of [14] presents another approach to decentralized the Federated Learning process. Their proposal consists in using IPFS[1] to replace the centralized server component. Their work also breaks the rigid roles of worker nodes and server and allows any node to initiate and join a Federated Learning process. Furthermore it is interesting for the scenario of low-capacity nodes that the layers of machine learning models are split into partitions, and the responsibility over the partition is distributed among the nodes. While the evaluations conducted focused on model convergence and fault-tolerance, the approach seems also to have an important potential for reducing the storage and computing requirements for Federated Learning nodes.

In the paper on BrainTorrent [15] a peer-to-peer approach is presented in which the clients of Federated Learning directly communicate with themselves. The target scenario are multiple medical centers which share whole-brain MRI T1 scans to solve a 20-class segmentation problem. The results characterized the performance of the system with settings which can be found in realistic environments such as an unbalanced data distribution.

From the reviewed related works it can be seen that research results have led to different approaches for privacy preservation of the data withing Federated Learning, highlighting the approach of differential privacy or combinations which leverage the support of external components such as those built with blockchain technology. It could also be observed that machine learning applications which train with personal data (e.g. medical data) and confidential data (e.g. sensor data from the IoT) have a new opportunity with the technique of Federated Learning. The application scenario of our work

[1] Interplanetary File System. https://ipfs.io/

addresses *machine learning applications for the increasing personal and medical data generated in the user environments*. We envision that user-owned computing devices like SBC in home environments will become increasingly used to run the training and inference of machine learning applications in peer-to-peer fashion, such as proposed in [14] and [15]. Therefore, our contribution aims to provide new insights to improve the *practical feasibility of doing Federated Learning in real resource-constraint devices* exemplified by the Raspberry Pi found in the user environments.

III. System Model

A. Federated Learning

Federated Learning (FL) [2] is a distributed machine learning technique where many clients - workers (e.g. mobile devices) in a collaborative way train a model under the orchestration of a central server (e.g. hosted in the cloud). During this process, the training data is kept locally on the devices as highlighted in Figure 1.

In general, in the Federated Learning process the model training occurs in the following way: first, the server initializes the global model w_t and sends it to all participating clients or workers. Each client k, after receiving the model w_t, trains the global model on its local data. Each client trains the local model for several *epochs* before sending it to the server in one communication round. When training completed each client returns the updated model w_t^k to the server (s_k is number of training samples held by client k). The server receives the updated model from each client, aggregates all those models to update the global model to get w_{t+1}. The iteration of this process often is called *rounds*. The client's local iterations of the model training is called *epochs*. The process can stop either by having reached the maximum number of rounds or until the model converges.

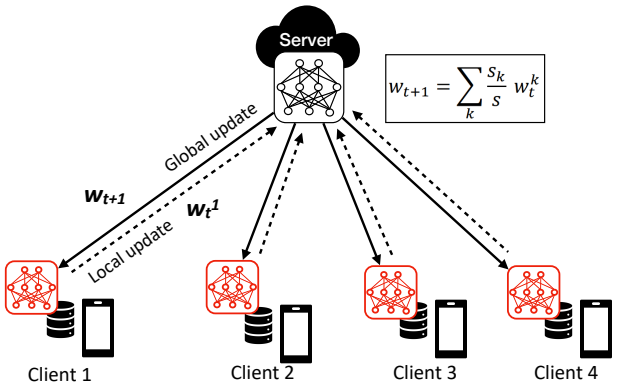

Fig. 1. Federated Learning Training Process.

B. Low-capacity computing nodes at the network edge

Without doubt, the computing capacity at the network edge is growing. While an important share of this growth can be attributed to mobile devices (e.g. smartphones), there is also a growing availability of computing resources by single-board computers (SBCs) on user premises, such as the Raspberry Pi and other mini-PCs. Indeed, there is an intense development in the major machine learning framework, e.g., Tensorflow Lite[2], to optimize trained machine learning models for running efficiently on low-capacity devices. At the same time, there is also a very active user community that builds machine learning applications for these tiny computers.

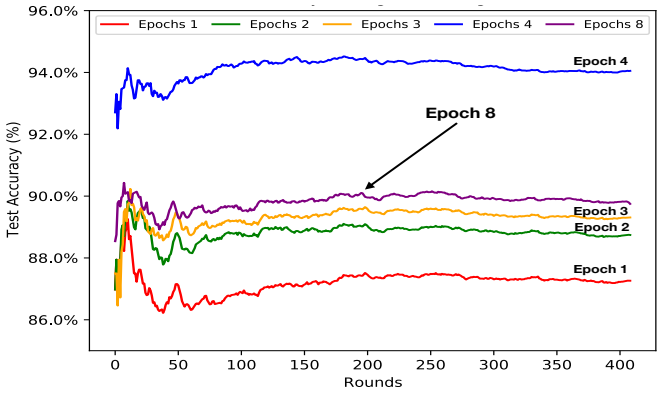

Fig. 2. FL under different epochs. Increasing the number of epochs does not necessarily lead to higher accuracy. Model trained with Epoch = 8 has lower accuracy than Epoch = 4.

For doing machine learning on edge devices, typically the training and inference phase takes place at different environments. Due to the lack of computing power at these low-capacity devices, the training of the models is normally done in the cloud. Once the models are trained with high-capacity machines, they are optimized for a lower resource consumption to fit better to the constraints of the low-capacity devices, and finally inference can ruin on the edge devices.

We consider in this paper a system of distributed low-capacity computing nodes, which are pooled together and collaborate in a Federated Learning process run over edge devices. The machine learning models are then trained at the computing nodes. We look at model accuracy, but also on the potential to reduce local epochs if there is no accuracy gain.

C. Proposed Algorithm: BePOCH

Performance of the Federated Learning (FL) algorithms is known to be highly sensitive to the FL system parameters, the hyper-parameters of the used machine learning model and the data distribution [16]. In practice, tuning the parameters and finding the right set for an FL algorithm is an expensive task and there is no clear way how to explore the parameter space. Taking into account the condition of resource-constrained devices which reach limits, i.e. they do not have the elasticity of a cloud resource, the problem becomes even more complex.

For instance, Figure 2 demonstrates how FL algorithm tries to find the most optimal epoch for the Federated Learning process. The most accurate and suitable epoch was 4, which dominance can also be seen in our evaluation results (Table I). Figure 2 reveals that increasing the number of epochs does not necessarily lead to higher accuracy. Using more epochs, however, increases the computation time and the use of computing resources.

Taking into account the importance of the epoch metric when training FL models in resource-constrained devices, we designed the ***BePOCH Algorithm*** (pseudocode highlighted in

[2]https://www.tensorflow.org/lite

Algorithm 1). BePOCH addresses the lack of knowing the right training parameter configuration of Federated Learning in an edge environment. It aims to find the set of training parameters that can lead to better learning performance in resource-constrained devices, specifically Raspberry Pis. The algorithm runs in several phases where the automation and deployment of various FL components occur (Phase 1, Phase 2 and Phase 3). For each of the pre-defined rounds (e.g., $R = 100$) by the user, the performance of each epoch candidate configuration is estimated by evaluating the learning performance in terms of model accuracy and CPU consumption (Phase 4). The configuration with the best estimated performance is considered the optimal parameter setting (Phase 5). Taking into account the devices with limited resources, the algorithm is executed once per dataset.

IV. Experiments and Results

In this section, first we present our real testbed created with Raspberry Pi devices, the dataset and models used. Then, we study the learning performance of our BePOCH algorithm comparing with the FL-Gao and OPT-Gao algorithms [8] which serve as our baselines. We focus our evaluation on the resource consumption of the BePOCH (CPU, RAM memory) and training time and temperature of the devices. In the context of our use case, these metrics are of high importance, since they prove applying machine learning on edge devices. A moderate resource consumption of the FL models is desirable since we are dealing with resource-constrained devices.

A. Testbed

The experimental setup contains real devices that are connected over a wireless network. As shown in Figure 3, there are 16 Raspberry Pi devices connected to two different wireless access points (APs). The two LANs at each AP are interconnected over a wireless link. The Federated Learning experiments use worker nodes in the LANs of both APs. Ubiquiti Nanostation M5[3] devices are used to enable this interconnection. With 150 Mbps real outdoor throughput and up to 15km range operating at 2.4GHz and 5 GHz, these devices deliver a good performance over the network. As computing nodes, we use the latest Raspberry Pi 4 Model B devices. They have a Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz processor, 8 GB SDRAM, 64 GB storage and IEEE 802.11ac wireless connection. The software used is: PyTorch version 1.8.0, OS Raspbian GNU/Linux 10 (buster) and Python version 3.7.3. In our experiments, 15 Raspberry Pi devices act as workers and one of them (or two) acts as a server.

[3]https://www.ui.com/airmax/nanostationm/

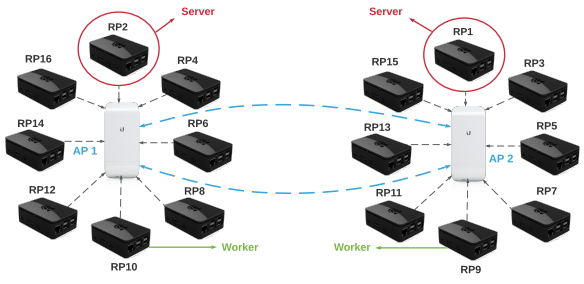

Fig. 3. Topology of the testbed

Algorithm 1 Best Epoch Algorithm - BePOCH

Require:
- `workers` ▷ Number of clients (2,4,8)
- `epoch` ▷ Local epoch (1,2,3,4,5)
- `batchsize` ▷ Batch size = 32
- `rounds` ▷ Number of training rounds (R=100)
- α ▷ Default parameter for Accuracy (α = 1.5)
- β ▷ Default parameter for Epochs (β = 1)
- `clients.csv` ▷ Clients and their IP Addresses

Phase 1 – Connecting to the nodes

procedure NodeConnection
  `nodes` ← *read_hosts*(`clients.csv`)
  ssh_to_hosts(`nodes`)
end procedure

Phase 2 – Server Start

procedure ServerStart
  server_start() ▷ Start the server and wait for the workers
end procedure

Phase 3 – Workers Start

procedure WorkerStart
  if (`worker` has the `ECGDataset`) **then**
    worker_start(`ServerIP`) ▷ Start the workers
  else *DownloadDataset*(`ECGDataset`)
    worker_start(`ServerIP`)
  end if
end procedure

Phase 4 – Best Epoch

procedure FindBestEpoch
  if (`rounds` $<$ `100`) **then**
    if (`test_acc` $>$ `best_test_acc`) **then**
      `best_epoch` ← `epoch`
      `best_test_acc` ← `test_acc`
      if (`epoch` $=$ `5`) **then**
        `best_test_acc` ← `0`
        `epoch` ← `0`
        `rounds` ← `rounds` + `10`
      end if
    end if
  else `epoch` ← `epoch` + `1`
  end if
end procedure

Phase 5 – Best Candidate

procedure BestCandidate
  α ← `1.5`
  β ← `1`
  getBestEpoch()
  getBestAccuracy()
  for each `d` in `data[]` **do**
    if (`best_epoch` − `d[epoch]` $\geq$ β) **then**
      if (`best_acc` − `d[acc]` $\leq$ α) **then**
        `best_candidate` ← `d`
      end if
    end if
  end for
end procedure

B. Dataset

For the performance evaluation of our BePOCH algorithm, we are using the popular ECG (Electrocardiogram) datasets from the MIT/BIH arrhythmia database [17] [18]. MIT-BIH is a dataset for ECG signal classification or arrhythmia diagnosis detection. In total we collect $26,490$ samples, with total of $68,901$ parameters. The baseline accuracy for the centralized model of ECG is 97.78%. The samples collected represent

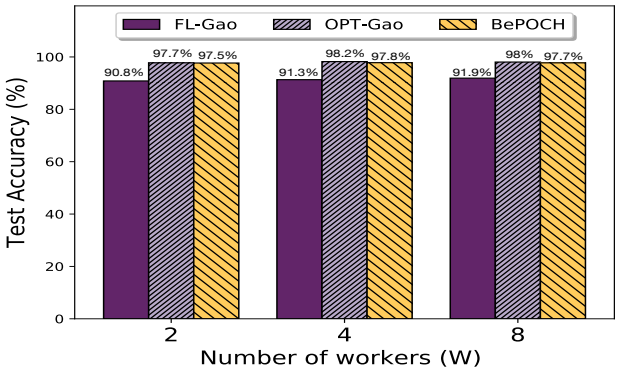

Fig. 4. FL-Gao vs OPT vs BePOCH (2, 4, and 8 workers)

5 heartbeat types (# of labels) : N (normal beat), L (left bundle branch block), R (right bundle branch block), A (atrial premature contraction), and V (ventricular premature contraction).

C. Performance Evaluation

1) Learning Performance (IID and Balanced Dataset) : Starting with balanced data distribution among workers, we evaluate the learning performance of the BePOCH and our baselines FL-Gao [8] and OPT-Gao using the ECG dataset. Figure 4 detail the testing accuracy over 100 number of rounds when the machine learning models are trained with 2, 4 and 8 distributed clients (Raspberry Pi devices) using three above-mentioned algorithms. Baseline: As e baseline we are using two algorithms:

- **FL-Gao:** As a first baseline we are using the implemented version of FL for Raspberry Pi devices from Yansong Gao et. al., [8]. In their tests, the 4conv+2dense 1D CNN model architecture is used. The learning rate is set to be 0.001. The batch size = 32. Further, they are testing the models just with 1 and 5 epoch and they are using up to 100 simulated clients (not real devices).
- **OPT-Gao:** is the second baseline which refers to the optimum configuration values (accuracy) in the FL-Gao algorithm. For each of the pre-defined rounds (e.g., R = 100) by the user, the performance of each epoch candidate configuration (1-5 epochs) is estimated in terms of model accuracy. Best accuracy epochs are plotted in the Figure 4. Epoch number is not fixed and can change per round.

Figure 4 depicts the testing accuracy of the FL-Gao, OPT-Gao and BePOCH when two, four and eight distributed workers are used to train the model. As can be seen from the Figure 4, in the case of two workers, the overall average accuracy improvement of BePOCH over FL-Gao is 6.7%. Comparing the BePOCH performance with the optimum values (OPT-Gao), we can see the BePOCH is slightly under-performing (0.2%). However, taking into consideration the number of epochs used, BePoCH is using 3 epochs per round comparing to OPT-Ga0 which is using 5 epochs per round. Less epochs result in less CPU consumption. BePOCH slightly sacrifices the accuracy but reduces the number of epochs per round which is a critical metric for resource-constrained devices. When increasing the number of workers to four, the overall average accuracy improvement of BePOCH over FL-Gao is 6.5%. Again, comparing to optimum values, BePOCH is slightly under-performing (0.4%), but is achieving to use less number of epochs per round (3 epochs per round comparing to 5 epoch per round used in OPT-Gao).

In our final use-case we use eight workers to train the model. It is interesting to note that augmenting three algorithms with more workers is not improving the accuracy. Figure 4 reveals that BePOCH is performing better than FL-Gao and slightly is under-performing with OPT-Gao. However, in the case of 8 workers, BePOCH is using 2 epochs per round which is a significant improvement comparing to 5 epochs per round used by the OPT-Gao algorithm. Overall, during the model training with two, four and eight workers, BePOCH is using 800 epochs in total comparing to 1500 used by the OPT-Gao, by slightly sacrificing the accuracy. Table I shows the best configurations selected by the OPT-Gao (highlighted with purple color) and BePOCH (highlighted with yellow color). *To summarize, BePOCH decreases up to* 46.7% *the number of epochs for* 100 *round experiment while keeping the level of accuracy. Further, up to 4 training workers, BePOCH decreases the training time on average for* 37%.

2) CPU and Memory Consumption : Figure 5 depicts the CPU and memory utilization of a single worker node during continuous training operations for 15 minutes (epoch=10). Figure 5 reveals that when the client trains the local model for several epochs before sending it to the server in one communication round, almost 4 cores of CPU is fully used (98% of CPU). The results suggest that training complicated models with several million parameters would be infeasible for such devices. Regarding memory, the consumption is low (110 MB) with regards to the available 8 GB of RAM.

3) Training Time and Temperature:: When comparing the average training time for different number of workers, when the number of workers increases, the training time increases slightly due to the fact that the server awaits for all clients to report back their newly trained model. As shown in Figure 6, with 2 workers the average training time is 914 seconds and with 8 workers 1046 seconds, respectively. As expected, more local epochs (e.g., epoch=5) proportionally prolongs training time on the client side. Regarding the CPU temperature, when training on the Raspberry Pi devices, the temperature goes high—usually more than 70°C (since FL training uses all 4 cores of the CPU). It is needed to cool down the device (.e.g.

	Workers								
	2			4			8		
R	E	A	T (s)	E	A	T (s)	E	A	T (s)
10	5	95.22	179	5	95.17	195	4	90.13	209
20	5	95.19	370	5	95.61	383	5	97	389
30	4	97.38	504	5	95.48	544	5	97.01	611
40	4	97.44	767	5	97.78	738	5	98.05	768
50	3	97.68	813	3	97.62	953	5	97.73	999
60	4	97.56	1110	3	97.81	1160	5	97.67	1209
70	4	97.66	1128	5	97.31	1306	2	97.75	1470
80	4	97.6	1491	4	97.9	1591	4	97.7	1719
90	5	97.75	1502	5	98.22	1631	5	97.79	1838
100	5	97.61	1772	3	97.74	1987	4	95.47	1960

TABLE I
BEST CONFIGURATION SELECTED: BEPOCH (YELLOW COLOR), OPT-GAO (PURPLE COLOR). EPOCH (E), ACCURACY (A), T (TIME-SEC)

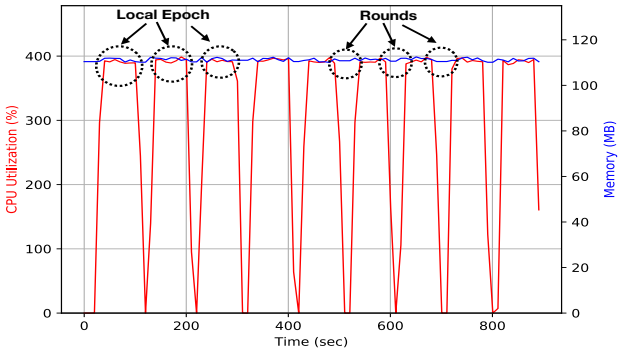

Fig. 5. CPU and Memory Consumption

fan, heat sink) to not suffer heat-related restrictions of the device. In the Figure 6 Client 0 refers to the server, and other 15 nodes are the clients performing model training.

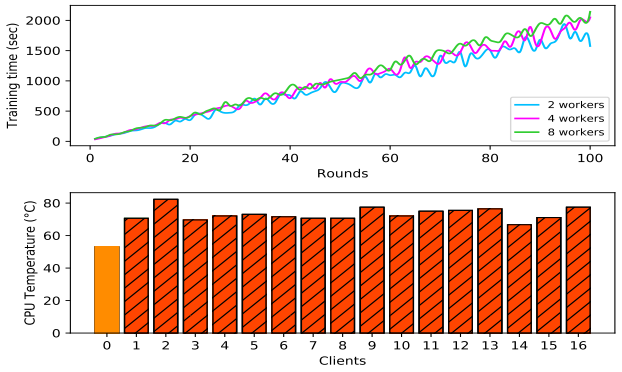

Fig. 6. Training Time and Device Temperature

V. Conclusion

This paper addressed a practical computing environment consisting of a set of distributed Raspberry Pi devices being used for federated machine learning model training. Such an environment is relevant as it becomes more and more available among individuals and communities of users. On the other hand, additional practical insights on resource consumption and guidelines for the best training configurations are needed in order to deliver an acceptable *training experience* on these low-capacity devices.

The BePOCH algorithm was proposed as a tool to obtain practical guidance on suitable parameter settings for the Federated Learning process in low-capacity devices, such as Raspberry Pi or similar. In the experiments it was shown that local training in these devices has a very high CPU consumption, for which it is of interest to be reduced at its minimum required value. Using the BePOCH algorithm it is possible to observe the best tradeoff in terms of model accuracy between the number of the local training epochs and the global model aggregation. In the experimentation it could be observed that the best learning performance was achieved when changing dynamically the number of training epochs at each round of the Federated Learning process.

In future work we aim to investigate how the knowledge of the most suitable number of training epochs can be extended to individual client behaviours. A potential usage is that the decision on the resources spent at the local client training would be aware of the learning benefit.

Acknowledgement

This work was partially funded by the Spanish Government under contracts PID2019-106774RB-C21, PCI2019-111850-2 (DiPET CHIST-ERA), PCI2019-111851-2 (LeadingEdge CHIST-ERA), and the Generalitat de Catalunya as Consolidated Research Group 2017-SGR-990. Suport was given also by the Agency for Electronic Communications (AEK) of North Macedonia.

References

[1] F. Sakr, F. Bellotti, R. Berta, and A. De Gloria, "Machine learning on mainstream microcontrollers," *Sensors*, vol. 20, no. 9, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/9/2638

[2] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, Jan. 2019. [Online]. Available: https://doi.org/10.1145/3298981

[3] O. Choudhury, A. Divanis, T. Salonidis, I. Sylla, Y. Park, G. Hsu, and A. Das, "Differential privacy-enabled federated learning for sensitive health data," 2020.

[4] M. Selimi, A. Lertsinsrubtavee, A. Sathiaseelan, L. Cerdà-Alabern, and L. Navarro, "Picasso: Enabling information-centric multi-tenancy at the edge of community mesh networks," *Computer Networks*, vol. 164, p. 106897, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128618312787

[5] J. Panadero, M. Selimi, L. Calvet, J. M. Marquès, and F. Freitag, "A two-stage multi-criteria optimization method for service placement in decentralized edge micro-clouds," *Future Generation Computer Systems*, vol. 121, pp. 90–105, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X21000935

[6] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "Federated learning for resource-constrained iot devices: Panoramas and state-of-the-art," 2020.

[7] Y. Jiang, S. Wang, B. Ko, W. Lee, and L. Tassiulas, "Model pruning enables efficient federated learning on edge devices," *CoRR*, vol. abs/1909.12326, 2019. [Online]. Available: http://arxiv.org/abs/1909.12326

[8] Y. Gao, M. Kim, S. Abuadbba, Y. Kim, C. Thapa, K. Kim, S. A. Camtep, H. Kim, and S. Nepal, "End-to-end evaluation of federated learning and split learning for internet of things," in *2020 International Symposium on Reliable Distributed Systems (SRDS)*, 2020, pp. 91–100.

[9] M. Malekzadeh, B. Hasircioglu, N. Mital, K. Katarya, M. E. Ozfatura, and D. Gunduz, "Dopamine: Differentially private federated learning on medical data," *ArXiv*, vol. abs/2101.11693, 2021.

[10] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4177–4186, 2020.

[11] H. Kim, J. Park, M. Bennis, and S. Kim, "On-device federated learning via blockchain and its latency analysis," *CoRR*, vol. abs/1808.03949, 2018. [Online]. Available: http://arxiv.org/abs/1808.03949

[12] J. Passerat-Palmbach, T. Farnan, R. Miller, M. S. Gross, H. L. Flannery, and B. Gleim, "A blockchain-orchestrated federated learning architecture for healthcare consortia," *CoRR*, vol. abs/1910.12603, 2019. [Online]. Available: http://arxiv.org/abs/1910.12603

[13] Q. Zhang, P. Palacharla, M. Sekiya, J. Suga, and T. Katagiri, "Demo: A blockchain based protocol for federated learning," in *2020 IEEE 28th International Conference on Network Protocols (ICNP)*, 2020, pp. 1–2.

[14] C. Pappas, D. Chatzopoulos, S. Lalis, and M. Vavalis, "Ipls : A framework for decentralized federated learning," 2021.

[15] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, "Braintorrent: A peer-to-peer environment for decentralized federated learning," 2019.

[16] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[17] G. Moody and R. Mark, "The impact of the mit-bih arrhythmia database," *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 45–50, 2001.

[18] R. Mark and G. Moody, "Mit-bih arrhythmia database directory." http://ecg.mit.edu/dbinfo.html, accessed: 2021-04-28.