Latency-Energy Tradeoffs in Federated Learning on Resource Constrained Edge Computing Systems

Simone Pampaloni, Nicola Tonellotto and Carlo Vallati

Information Engineering Department, University of Pisa, Pisa, Italy

Abstract

Artificial intelligence and machine learning have become of crucial importance in many scientific and industrial fields, thanks to the ability to extract information, make predictions and identify patterns on data. For the creation of increasingly accurate predictive models, these technologies are based on the collection and control of large amounts of data within controlled systems. Federated learning is a new framework that exploits the computational capabilities and local data of a set of multiple resource-constrained devices coordinated by a central server for the creation of a shared global predictive model, without any centralised data collection.

In this work, we focus on assessing the performance of federated learning executed on resource constrained Edge computing system. A set of experiments to assess the energy consumption and processing times on a set of heterogeneous GPU-enabled embedded systems were executed. Our analysis shows that, by varying the amount of data that each system is in charge of processing, it is possible to identify a trade-off between the overall energy consumption of the devices and the processing time required to train an effective predictive model.

Keywords

Federated Learning, Latency-Energy Tradeoffs, Performance Evaluation, Embedded Systems

1. Introduction

Machine and user generated data is rapidly increasing due to the rapid advancement of Internet of Things (IoT), social networking applications and ubiquitous wireless connectivity via 5G/6G networks. The analysis of such amount of data is often carried out via Artificial Intelligence (AI) techniques to create predictive models to support users in planning and decision making detection or to improve cyber-physical systems efficiency via detection of anomalies, e.g. for predictive maintenance [1]. Such an increasing amount of data and the consequent demand to train AI systems on has generated new data management problems, especially in application fields where it is necessary to train predictive models on large quantities of data, but it is not convenient or feasible, in terms of communication efficiency or bandwidth available, to transmit them to the cloud or it is not possible to collect them, for legal, strategic or economic reasons.

Recently a novel paradigm has received significant attention, Federated Learning (FL) [2, 3], as a solution to train a deep learning model from decentralized data. FL removes the need to move

AI6G'22: First International Workshop on Artificial Intelligence in beyond 5G and 6G Wireless Networks, July 21, 2022, Padua, Italy

s.pampaloni@studenti.unipi.it (S. Pampaloni); nicola.tonellotto@unipi.it (N. Tonellotto); carlo.vallati@unipi.it (C. Vallati)

0000-0002-7427-1001 (N. Tonellotto); 0000-0002-7833-5471 (C. Vallati)

large amount of data for training of models, thus ensuring scalability and data privacy/security. Initially proposed for data analysis on mobile devices, FL is also expected to be widely adopted on Edge computing, which extends the traditional centralized cloud computing paradigm with additional computing layers installed in proximity of users and cyber-physical systems, for instance exploiting computing devices installed at the edge of the network [4]. Edge computing systems can support the execution new applications characterized by stringent latency and privacy requirements, such as industrial applications. The joint adoption of FL and Edge computing is also expected to play a central role for the implementation of novel network management strategies thus to ensure scalability and minimal human intervention in future 5G/6G networks [5]. In this context, FL can be exploited at the edge to develop applications that could process data locally, resulting in a reduced latency for data analysis and deep learning model training.

Edge systems, however, are characterized by limited computing and storage capabilities as they are implemented via embedded systems, which offer an environment quite constrained if compared with a resource rich cloud environment. In addition to this, some edge computing devices might be characterized by energy constraints, as they might be installed on energy restricted environments, e.g. edge devices installed in autonomous vehicles [6]. FL solutions deployed on edge systems will require proper configuration in order to vary the amount of data assigned to each device in order to take into account heterogeneous computing capabilities and master the latency-energy consumption tradeoff.

In this paper we analyze the performance of FL when executed on different edge computing systems, characterized by heterogeneous resource capabilities. The goal is to analyze the overall performance in order to offer some guidelines on how FL applications can be configured to optimize the processing time required to train an effective predictive model on different devices, while considering the energy consumption at the same time.

In our experiments the training of a model for convolutional neural network for image recognition implemented in a distributed manner using FL is considered as an example. A large set of experiments with different heterogeneous GPU-enabled embedded systems are executed to measure training latency and energy consumption of different devices. Our analysis highlights that, by varying the amount of data that each system is in charge of processing, it is possible to master a trade-off between the overall energy consumption of the devices and the processing time required to train an effective predictive model.

The rest of the paper is organized as follows: in Section 2 we overview some background concepts and overview the related work, in Section 3 we present our experimentation methodology, in Section 4 we analyze the results of our experiments, while in Section 5 we present the conclusions of this work.

2. Background & Related Works

The pervasive diffusion of personal mobiles and IoT devices produces huge amounts of data. Currently, AI-based predictive analytic solutions built on such data are typically generated through centralised cloud-based services. The state-of-the-art approach for training a high quality AI model over mobile and IoT smart devices leverages the delivery of collected data to a service hosted on the Cloud, which is in charge of training the model [7]. However, these cloud-based solutions raise major concerns about the user privacy of online services. A recent approach to mitigate this issue is to move the computation where data is stored, i.e., on personal IoT devices connected to the Internet. In edge computing [8] data processing and storage capabilities are not exclusive characteristics of centralized data centers, but an additional layer, called *edge*, is placed in the middle between the Cloud and the IoT devices. This layer allows for storing data and executing applications on resource-constrained edge computing systems directly connected with IoT devices. Moreover, edge computing allows to preserve the confidentiality of user data. In this scenario, the recent introduction of *artificial-intelligence-as-a-service* (AIAAS) [9, 10, 11] tackles significant innovations across all the industrial sectors in particular in the *artificial intelligence of things* (AIoT) [12, 13], where the data required to train AI solutions are kept local to the device without disclosing private or sensitive data.

Federated learning [2, 3] is a leading approach for training AI solutions, i.e., neural networks, in AIoT solutions [14]. In FL, the computation involved in the training of a neural network is moved closer to where data are produced and store. FL can naturally apply to the IoT-Edge-Cloud scenario, requiring the preservation of data privacy and ownership. In an FL scenario applied to edge computing, every edge computing system receives a partially trained neural network from a central cloud server, performs additional training using data provided by the respective local IoT devices to refine the neural network without disclosing any private data, and sends back the refined neural network to the Cloud server. The Cloud server collects all the locally-trained neural networks, generates a new global neural network, and broadcasts the global network back to the edge computing systems for a new round of local training. By doing so, the complex aggregation of Machine Learning (ML) models on a cloud server is decoupled from the storage of training data on edge computing systems to preserve data ownership and privacy.

The adoption of FL in Edge computing environments is already proposed for many applications, ranging from IoT applications to crowd-sourcing data analysis [1]. Among them, the application of FL for the management of the computing and networking infrastructure is envisioned, e.g. to implement user management solutions, drive content caching and manage task offloading from user devices to the infrastructure [15]. Edge computing, however, is characterized by resource constrained systems, which are usually embedded devices with limited capabilities in terms of computing power, RAM and GPU performance, and, in some specific use-cases where the edge devices are battery powered, they might have constraints also in terms of energy. In this context, models and solutions for data and task management and allocation are crucial to ensure scalability and proper usage of scarce resources. In [16], for instance, an energy aware resource management strategy is proposed to manage the training model process in FL system, by minimizing the training loss and overall time, considering also energy constraints. This work, however, analyzes the performance of the proposed approach only via simulations.

In this paper, instead, we carry out an analysis based on real experiments exploiting different heterogeneous GPU-enabled embedded systems, thus re-creating a realistic edge environment for FL execution. Our goal is to derive guidelines to master the trade-off between energy consumption and processing times, so the overall process of distributing data for training can be characterized.

3. Experimental Setup

In this section we illustrate the experimental setup of our experiments, in terms of hardware exploited and software used. Then we describe the task under evaluation, together with the neural network adopted, the dataset employed and the performance metrics we measured. Finally, we report the research questions that we will investigate in our experiments, which are detailed in the following section.

Hardware. For our experiments we exploited a Cloud/Edge computing platform available at the University of Pisa. Such platform implements both the cloud and edge layers and provides a realistic environment to test distributed computing solutions that adopt the Cloud/Edge computing paradigm. The platform in particular includes the following components: (i) a Cloud platform based on OpenStack, an open-source Infrastructure as a Service platform that supports the creation of virtual machines, hosted at the datacenter of the university, (ii) an Edge/Fog testbed composed of heterogeneous embedded systems with GPU support, installed in the "Cloud Computing, Big Data and Cybersecurity" laboratory. The cloud platform and the testbed are connected through a fiber link.

The following embedded systems are considered for our experiments:

- **NVIDIA Jetson Nano**: an embedded system that is equipped with a quad-core ARM microcontroller, 4GB of RAM and a 128-cire NVIDIA GPU;
- **NVIDIA Jetson TX2**: an embedded system with a quad-core ARM microcontroller, 8GB of RAM and a 256-cire NVIDIA GPU

Two boards of each type are exploited in our experiments to test different configurations, the Cloud functionalities, instead, are deployed on a virtual machine hosted on the OpenStack platform. Boards overall energy consumption is measured via a commercial smart plug, namely the MEROSS MSS210.

Software. We use Flower[1] [17] to implement and execute federated learning on our computational resources. Flower automatically manages the distribution of the neural networks form the Cloud to the edge resources, as well as the collection of different locally-trained models on the edge resource to the Cloud, and their aggregation through the FederatedAveraging algorithm [2]. In our experiments, the number of federated learning rounds is 10. The communications are synchronous, i.e., the Cloud server perform model aggregation only when both devices communicate their local models, then it dispatches the new global models to the device.

Task. In our experiments we focus on an image recognition task. This task is representative of different IoT-Edge-Cloud scenario, such as intrusion detection [18] and occupational safety [19]. We exploit a convolutional neural network specifically designed for this task, called ResNet [20], and available in PyTorch[2]. We use the default hyperparameter values for the CIFAR dataset, namely the cross entropy loss function, a learning rate equal to 0.001, 4 epochs per local training at each edge resource and a batch size of 32 samples.

[1] https://flower.dev

[2] https://pytorch.org

Dataset. For the training and the evaluation of our task, we use the popular CIFAR-10 dataset [21]. CIFAR-10 contains 60,000 RGB images of 32×32 pixels, represnting objects in 10 different categories, i.e., 6,000 images per categories. In our experiments, we randomly split the whole dataset in a training set, with 50,000 images, and an evaluation set, with the remaining 10,000 images. The evaluation set will be used to assess the effectiveness performance of the final global model trained with FL.

Metrics. For the analysis of the performance of federated learning on the edge computing scenario, we focus on the behaviour of the various system components during our experiments. To assess the effectiveness of the federated learning procedure when varying the distribution of the training set among the different resources, we use *accuracy*, i.e. the ratio between correct classifications over the total number of classifications, computed on the evaluation set. Our main efficiency metrics are device training time and device energy consumption. The *device training time* measures the mean time required by a given device to locally train the ResNet neural network at each using the 4 epochs per FL round. The *device energy consumption* measures the mean the energy consumed by the given device during local training.

Research Question. In our experiments, we investigate the following research questions:

- **RQ1.** How does the mean training time per round vary with the computational load on each device and on the Cloud server?
- **RQ2.** How does the mean energy consumption per round vary with the computational load on each device and on the Cloud server?

In the following section, we illustrate and discuss the experimental results w.r.t. these RQs.

4. Experimental Analysis

Figure 1 reports the mean training time per round (in seconds) w.r.t. the computational load on each device and on the Cloud server. The red bars, resp. the blue bars, report the mean training times per round on the Jetson Nano, resp. on the Jetson TX. The purple line reports the average training time of the whole model on the Cloud server. Since the communication times are negligible, the total training time corresponds to the maximum time required among the different devices. The training dataset configuration that minimises the total training time corresponds to a split of the 27.5% of the dataset on the Jetson Nano, and the remaining 72.5% on the Jetson TX. In this case, the mean training times per round are almost identical on the two devices.

Regarding RQ1, we conclude that, in order to minimise the total mean training time, a dataset should be distributed across the device balancing the corresponding computational power.

Figures 2 reports the mean energy consumption per round (in kilojoules) w.r.t. the computational load on each device and on the Cloud server. The red bars, resp. the blue bars, report the mean energy consumed per round on the Jetson Nano, resp. on the Jetson TX. The purple line reports the average energy consumed during a federated round to train the local models on all devices. The total energy consumed decreases as more training data is assigned to the

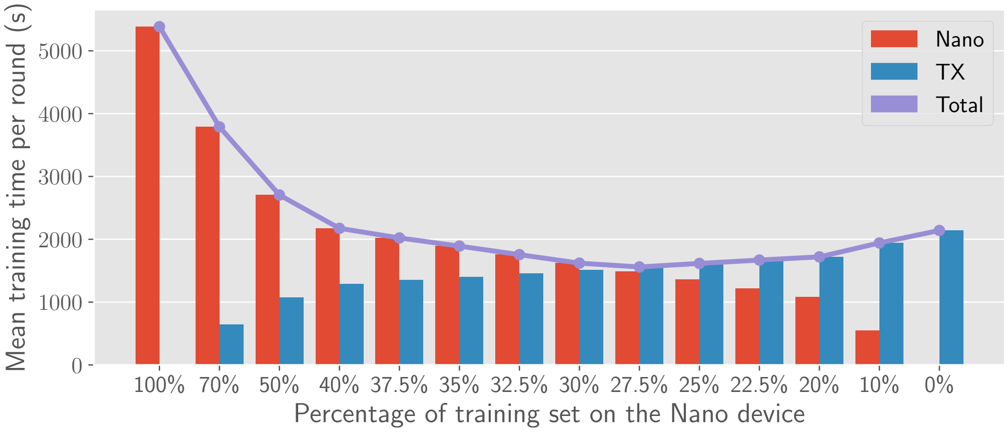

Figure 1: Mean training time per round, in seconds, on each device and on the Cloud server, by varying the training set splitting across the devices.

Jetson TX. This is due to the higher computing performance of the Jetson TX w.r.t. the Jetson Nano. In fact, the Jetson TX is ∼2.5 times faster than the Jetson Nano (see Fig. 1), and therefore it completes the whole training in a shorter time.

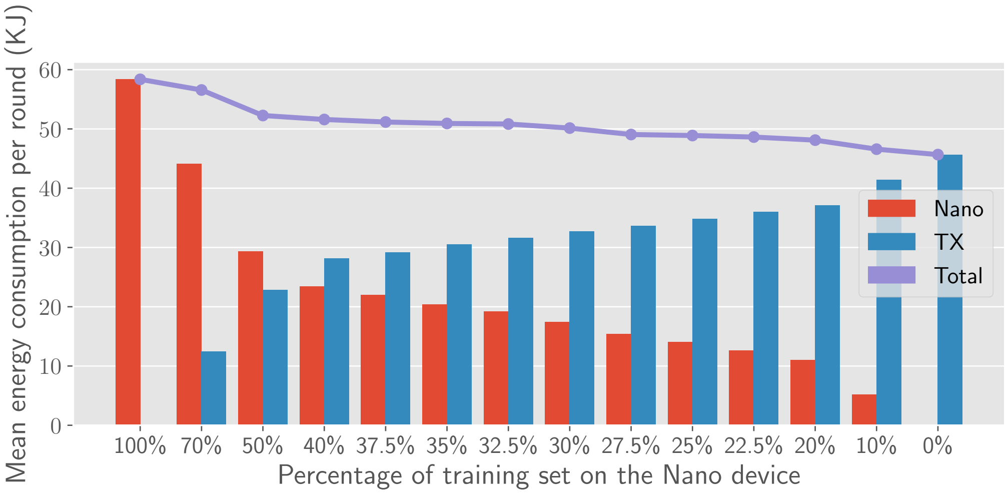

Figure 2: Mean energy consumption per round, in seconds, on each device and on the Cloud server, by varying the training set splitting across the devices.

Regarding RQ2, we conclude that the total energy consumption of a federated learning infrastructure depends on both the characteristics of the edge devices and the size of the datasets assigned to each of them.

Overall, we conclude that there is a tradeoff between training latency and energy consumption in federated learning infrastructures, depending on the characteristics of the dataset, and its splitting, as well as the specific devices. To minimise the overall training time, a system designer should adopt a load balancing strategy taking into account both the computational power of the edge devices and the dataset available to each of them. The solution identified by this approach, however, could not minimise the overall energy consumption, which depends on the specific devices.

5. Conclusions

In this work a set of experiments on a realistic edge computing environment is carried out to measure training time and energy consumption of a FL model. Two different board models were considered, one board with basic capabilities in terms of RAM and GPU and another more powerful board. Different configurations for the allocation of the training data were assessed in our experiments in order to measure both energy consumption and training time. Our experiments highlighted a tradeoff between training latency and energy consumption, which depends on the characteristics of the dataset and its partition between boards with different capabilities. A load balancing strategy could be adopted, however, the minimization of the training time might not result in the minimal energy consumption as it mainly depends on specific device characteristics.

As future work, we plan to further investigate the pareto front highlighted by our results and to derive a model for both processing time and energy consumption to be used for designing a load balancing strategy.

Acknowledgments

Work funded by the Italian Ministry of Education and Research in the framework of the CrossLab project (Departments of Excellence), and by the University of Pisa in the framework of the PRA 2020 program (AUTENS project, Sustainable Energy Autarky).

References

[1] 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 Journal on Selected Areas in Communications 37 (2019) 1205–1221. doi:10.1109/JSAC.2019.2904348.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Proc. AISTATS, 2017.

[3] P. Kairouz, et al., Advances and open problems in federated learning, ArXiv abs/1912.04977 (2019).

[4] W. Shi, S. Dustdar, The promise of edge computing, Computer 49 (2016) 78–81. doi:10.1109/MC.2016.145.

[5] S. Niknam, H. S. Dhillon, J. H. Reed, Federated learning for wireless communications: Motivation, opportunities, and challenges, IEEE Communications Magazine 58 (2020) 46–51. doi:10.1109/MCOM.001.1900461.

[6] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, W. Shi, Edge computing for autonomous driving: Opportunities and challenges, Proceedings of the IEEE 107 (2019) 1697–1716. doi:10.1109/JPROC.2019.2915983.

[7] H. Li, K. Ota, M. Dong, Learning IoT in edge: Deep learning for the Internet of Things with edge computing, IEEE Network 32 (2018) 96–101.

[8] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, G. Srivastava, A survey on security and privacy of federated learning, Future Generation Computer Systems 115 (2021) 619–640.

[9] S. B. Calo, M. Touna, D. C. Verma, A. Cullen, Edge computing architecture for applying AI to IoT, in: Proc. BIG DATA, IEEE, 2017, pp. 3012–3016.

[10] M. S. Munir, S. F. Abedin, C. S. Hong, Artificial Intelligence-based Service Aggregation for Mobile-Agent in Edge Computing, in: Proc. APNOMS, IEEE, 2019, pp. 1–6.

[11] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, A. Sadeghi, Dïot: A federated self-learning anomaly detection system for IoT, in: Proc. ICDCS, 2019, pp. 756–767.

[12] F. Samie, L. Bauer, J. Henkel, From cloud down to things: An overview of machine learning in internet of things, IEEE Internet of Things Journal 6 (2019) 4921–4934.

[13] M. Mohammadi, A. Al-Fuqaha, S. Sorour, M. Guizani, Deep learning for iot big data and streaming analytics: A survey, IEEE Communications Surveys & Tutorials 20 (2018) 2923–2960.

[14] J. Mills, J. Hu, G. Min, Communication-efficient federated learning for wireless edge intelligence in iot, IEEE Internet of Things Journal 7 (2019) 5986–5994.

[15] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, M. Chen, In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning, IEEE Network 33 (2019) 156–165. doi:10.1109/MNET.2019.1800286.

[16] C. W. Zaw, S. R. Pandey, K. Kim, C. S. Hong, Energy-aware resource management for federated learning in multi-access edge computing systems, IEEE Access 9 (2021) 34938–34950. doi:10.1109/ACCESS.2021.3055523.

[17] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, T. Parcollet, N. D. Lane, Flower: A Friendly Federated Learning Research Framework, ArXiv abs/2007.14390 (2020).

[18] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, W.-Y. Lin, Intrusion detection by machine learning: A review, Expert Systems with Applications 36 (2009) 11994–12000. URL: https://www.sciencedirect.com/science/article/pii/S0957417409004801. doi:https://doi.org/10.1016/j.eswa.2009.05.029.

[19] G. Gallo, F. Di Rienzo, P. Ducange, V. Ferrari, A. Tognetti, C. Vallati, A smart system for personal protective equipment detection in industrial environments based on deep learning, in: 2021 IEEE International Conference on Smart Computing (SMARTCOMP), 2021, pp. 222–227. doi:10.1109/SMARTCOMP52413.2021.00051.

[20] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778. doi:10.1109/CVPR.2016.90.

[21] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images, Master's thesis, Department of Computer Science, University of Toronto (2009).