Resource-Aware Split Federated Learning for Edge Intelligence

Amna Arouj*
Queen Mary University of London
United Kingdom

Ahmed M. Abdelmoniem†
Queen Mary University of London
United Kingdom

Ahmad Alhilal
Hong Kong University of Sci & Tech
Hong Kong

Linlin You
Sun Yat-Sen University
China

Chen Wang
Huazhong University of Sci & Tech
China

ABSTRACT

This research investigates Federated Learning (FL) systems, wherein multiple edge devices cooperate to train a collective machine learning model using their locally distributed data. The focus is on addressing energy consumption challenges in battery-constrained devices and mitigating the negative impact of intensive on-device computations during the training phase. In the widespread adoption of FL, variations in clients' computational capabilities and battery levels lead to system stragglers/dropouts, causing a decline in training quality. To enhance FL's energy efficiency, we propose *EAFL+*, a pioneering cloud-edge-terminal collaborative approach. *EAFL+* introduces a novel architectural design aimed at achieving power-aware FL training by capitalizing on resource diversity and computation offloading. It facilitates the efficient selection of an approximately-optimal offloading target from Cloud-tier, Edge-tier, and Terminal-tier resources, optimizing the cost-quality tradeoff for participating client devices in the FL process. The presented algorithm minimizes the dropouts during training, enhancing participation rates and amplifying clients' contributions, resulting in better accuracy and convergence. Through experiments conducted on FL datasets and traces within a simulated FL environment, we find *EAFL+* eradicates client drop-outs and enhances accuracy by up to 24% compared to state-of-the-art methods.

KEYWORDS

Federated Learning, Heterogeneity, Resource-Aware, Offloading

ACM Reference Format:
Amna Arouj, Ahmed M. Abdelmoniem, Ahmad Alhilal, Linlin You, and Chen Wang. 2024. Resource-Aware Split Federated Learning for Edge Intelligence. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

*Work done during an internship at Queen Mary University of London.
†Corresponding author (ahmed.sayed@qmul.ac.uk)

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Through the advancements in technology and the rise of the wireless industry, a large amount of data is born at the edge. Each device is networking and crunching data faster than ever before and the number is expected to reach 7.33 and 1 billion, for smartphones and wearable devices respectively [9]. In this ever-expanding network of devices, IoT (Internet of Things) surfaces as a significant terminal, playing an important role in the formation of the digital landscape. IoT devices include smartphones, home appliances, wearables sensors, and autonomous vehicles. With their ability to sense and communicate with their surroundings, IoT acts as a channel for the integration of digital and physical realms.

These interconnected devices and distributed data that are dispersed geographically lead to the emergence of **Edge Intelligence**, which allows multiple edge nodes to learn a shared ML model while respecting the end user's data privacy.

Federated Learning (FL) is an emerging distributed model aggregation method where the end users run stochastic gradient descent (SGD) locally and then send the updated models to the server which accumulates them to obtain a new global model. This is fueled by advancements in mobile System-on-Chips (SoCs), edge resources, and 5G/6G communications. Nowadays, mobile devices are equipped with neural processing units (NPUs) and graphic processing units (GPUs) to execute computations for AI/ML applications.

FL in Edge IoT [18, 26] is seen in prominent applications like next-word predictions (Gboard) [16], vocal/face classifiers (Face ID and Siri) [4], virtual assistants, smart cities and augmented reality, etc [19]. However, FL in IoT also faces various challenges [27, 28, 41]. Typical attributes of these devices include limited energy, storage, and computing resources; hence, considering these limitations, energy-efficient FL is crucial for these services. In tackling these challenges, most studies primarily focus on optimizing either the statistical model efficiency (improving training accuracy with fewer rounds) [21, 30] or the system efficiency (shorter training rounds) [20, 24]. Additionally, other research works prioritize enhancing privacy guarantees and robustness [7, 8, 13]. For instance, in the case of Oort [20], the emphasis is on optimizing time-to-accuracy while disregarding the potential impact on resource and energy consumption [3, 5].

In this work, we leverage heterogeneous resources distributed across the cloud-edge-terminal continuum and present *EAFL+*, a collaborative approach to improve the FL system performance by leveraging available resources at the edge via split computing [23]. This realizes a resource-aware mechanism to reduce the burden on weak devices by offloading part of their computations to power-rich devices. Our contributions are:

979-8-3503-6336-4/24/$31.00 ©2024 IEEE
DOI 10.1109/SenSys-ML62579.2024.00008

(1) We introduce a novel design towards resource-aware split FL suitable for battery-powered FL scenarios to reduce client dropouts and increase participation levels.
(2) We propose, *EAFL+*, a novel approach that improves FL performance by optimizing the selection of Cloud-Edge-Terminal assistance to which computations are offloaded.
(3) We show, via experiments on real FL benchmarks, that *EAFL+* achieves its objectives of realizing energy-efficient FL systems and produces high-quality models compared to the state-of-the-art methods.

The paper includes background information in Section 2, and the motivation of this work in Section 2.4. Then, we present, *EAFL+*, the proposed split learning approach in Section 3. Next, we give the experimental setup and results in Section 4. Finally, we give the related work in Section 5 and the conclusion in Section 6.

2 BACKGROUND AND MOTIVATION

We present FL, the cloud-edge-terminal continuum and heterogeneity issues followed by the motivation of this work.

2.1 Federated learning

Federated learning, a promising networking architecture approach, runs machine learning on decentralized data. It leverages the available computing resources across a massive pool of edge devices. More precisely, all participants collectively train on one global learning model and through continuous interactions perform regular model parameter updates

FL design consists of two main entities —*Clients*, the data owners (e.g. smartphones, tablets, laptops, auto-vehicles, etc.) and — *Aggregator*, the model owner (e.g. server). The global model undergoes training by specifying the number of epochs **E** (rounds) till the model converges. Additional parameters such as minibatch size **B**, and participants per round **K** are also specified and are defined by the FL-based services [7, 19]. In each training round begins with the aggregator selecting **K** participants among **N** available devices based on the given criteria. *(Step 1)* Server transmits the model's current version along with other necessary hyper-parameters to the selected devices. *(Step 2)* Each participant starts training the model by performing **E** local optimization steps of batch size of **B**. *(Step 3)* Learners transmit their computed model updates back to the server. *(Step 4)* The server gathers the model updates from each participant for aggregation and checkpoints it. *(Step 5)* The stages will repeat until the desired accuracy is attained.

2.2 Environment Heterogeneity

In Federated Learning (FL) training, the occurrence of client dropout mirrors the straggler issue, wherein certain clients exhibit delays in uploading their local models [24]. To address this, a straggler-resilient framework has been introduced in [34] which dynamically selects clients by incorporating statistical attributes of their data. However, client drop-out presents a more critical scenario, as these clients fail to upload their models in the ongoing round and are expected to remain inaccessible for a period, rendering existing straggler mitigation strategies ineffective [38].

Existing FL solutions have simply overlooked the resulting consequences due to the interplay of client devices and training speed (e.g., using significant local steps to save communication) [24] leads to intensive on-device computation for extended periods, causing sudden drain of the battery or even worse – burning of the device, thus resulting in unavailability of the client. The majority of the previous works assume that FL training is activated once smartphones are connected to a power source because of the significant energy consumption during FL training [31, 37].

2.3 Emerging Cloud and Edge Computing

An emerging option to enable cloud computing at the edge is to carry out the processing directly at the User Equipment (UE), through an ad-hoc cloud. Numerous UE in each other's vicinity pool their computation power and thus process highly demanding applications. Several challenges need to be addressed to facilitate the ad-hoc cloud: 1) No guarantee that data will be sent back to the source; 2) coordination of UE; 3) Given the battery and other constraints, UE have to be encouraged to share their computing power with others; 4) security and privacy issues.

Therefore, Edge-to-Cloud continuum approaches has been emerging [10, 25, 29, 33, 35]. In these approaches, cloud computing mitigates the limited storage and computing capabilities of end devices, while, edge computing brings the functionality closer to the end device overcoming the latency, security, and privacy constraints.

2.4 Motivation

In this study, we address a gap identified in previous research on federated learning, particularly in contexts where MEC support could be advantageous. Prior studies predominantly concentrate on minimizing training duration or computation latency in federated learning systems. Yet, exploiting MEC resources enables the enhancement of underpowered devices by implementing techniques to distribute computation across alternative resources. This alleviates computational strain and shortens model training duration, effectively mitigating the occurrence of stragglers. Nonetheless, integrating MEC into federated learning encounters certain hurdles, which serve as the impetus for our investigation. We outline these challenges as follows:

(1) *Impact of heterogeneity:* How to mitigate client dropouts in FL training in resource heterogeneous setup?
(2) *Limitations in State-of-the-art:* How to design a resource-aware method that optimize resource allocation and offloading for reduced FL time-to-accuracy?
(3) *System scalability and dynamism:* What strategies can be employed to achieve scalability and resilience with challenges such as dynamic resource availability, diverse computational resources, and unstable communication bandwidth?

Fig.1 illustrates the hierarchical structure delineating three potential offloading points for split computing: client, edge, or cloud. The choice of offloading point offers varied advantages contingent upon contributing factors. Opting for the client tier for the split typically results in reduced latency and increased bandwidth during communication. Conversely, edge/cloud servers offer advantages in terms of available computational power. Ideally, solutions with low latency and high accuracy are preferred. However, the overarching principle prioritizes solutions with minimal latency, accepting only a marginal decrease in accuracy, if any. We propose an adaptive

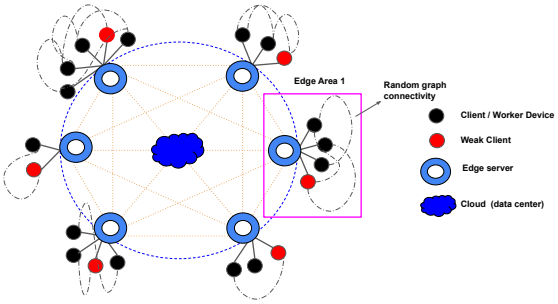

Figure 1: Depiction of MEC environment and the computation offloading opportunities in Edge-Cloud continuation.

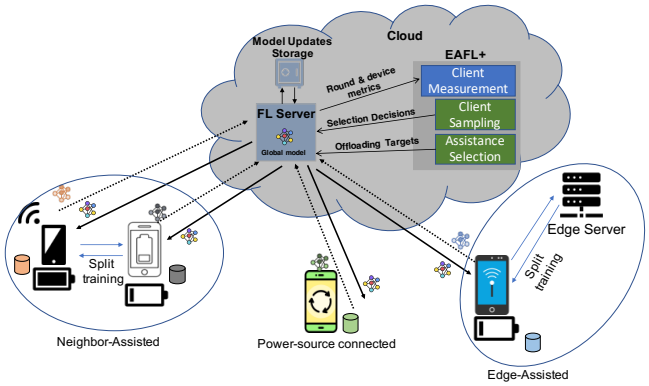

Figure 2: High-level Architectural Design of *EAFL+*.

auto mechanism to optimize the cost-benefit tradeoff, particularly benefiting weaker clients.

Motivated by the challenges discussed in current MEC-assisted distributed architectures and to leverage its benefits in providing richer resources, we propose *EAFL+* for a cloud-edge-terminal collaborative architecture for resource-aware federated learning. *EAFL+* is a technique, that addresses client heterogeneity by leveraging resource usability via split learning to further maximize the number of devices that can complete the training process. *EAFL+* mitigates the effect of dropouts and stragglers by offloading a part of its network to a selected offloading point chosen by the server.

3 *EAFL+* DESIGN

We present the architectural specifications of the proposed training algorithm, *EAFL+*, leveraging off-device resources through a multi-tier offloading approach to enhance overall system efficiency. The main objectives of this work are as follows:

(1) The primary objective of this approach is to reduce the computation load on clients with limited capacity. To achieve this, EAFL+ enables multi-tier split learning.
(2) The second objective is to mitigate the issues of stragglers and dropouts. EAFL+ achieves this by distributing the computation across the Cloud-Edge-Terminal continuum, resulting in optimized power consumption, improved statistical efficiency, and reduced impact of heterogeneity.
(3) The third objective is to select the optimal point for the computation. The innovation in EAFL+ lies in the multi-tier design to offload the computation. EAFL+ identifies the most suitable offloading target across Cloud-Edge-Terminal tiers.

3.1 System Model and Architecture

We present in Fig. 1 a network architecture for the edge-cloud continuum. This involves a heterogeneous environment characterized by varying computational capabilities and terminal nodes across different regions connected to their nearest edge servers within the edge network. Inspired by IoT, these edge servers establish connections with a distant cloud via the public internet. Terminal nodes possess the capability to communicate both among themselves and with the edge/cloud servers. Our network model encompasses two tiers: the client tier, depicted in black (red indicating weak devices), and the edge-to-cloud tier, represented in blue.

Typically, devices with limited computational capacity may delegate a portion of their computational workload, such as neural network computations, to a chosen offloading destination within the client, edge, or cloud tiers. Each offloading destination entails a trade-off between computational efficiency and latency, adding complexity to the selection process. Within the client tier, the algorithm exploits nearby high-capacity clients to facilitate computation offloading, resulting in a reduction in latency due to peer-to-peer communication proximity. Conversely, the Edge/Cloud tiers offer dependable and computationally robust resources to less powerful clients, albeit with extended communication latency. Additionally, Edge/Cloud introduces supplementary communication overheads and heightens privacy risks by transmitting data over the public internet. Despite federated learning (FL) providing a level of privacy, communication over the internet exposes the split learning stage to potential adversarial attacks, impacting its efficacy [12].

Automating the decision-making process in *EAFL+* presents significant challenges due to the dependency of offloading decisions on multiple factors, such as latency, network bandwidth, privacy considerations, and device computational capabilities. The objective is to identify the optimal offloading point within this complex multi-dimensional parameter space, ensuring a viable and economically efficient solution for devices with limited computational resources. Additionally, practicality necessitates the development of designs that enhance current federated learning (FL) systems.

3.2 Client Monitoring and Assistance Selection

The overarching architecture of *EAFL+* is depicted in Fig. 2, highlighting its multi-step search process aimed at identifying an optimal assisting node or offloading point from a range of feasible solutions. This addresses the computational requirements of identified weak clients through split computing methodologies. Client selection and aggregation in *EAFL+* follow established methodologies observed in prior studies, such as [5]. Each training round involves the identification of potential bottlenecks, characterized by dropout or straggler clients, i.e., weak clients. Subsequently, the server executes an assistance selection algorithm for each weak client, facilitating the selection of the appropriate offloading target

To identify a suitable offloading point, we employ Algorithm 1. Initially, we gather the set of potential splits, denoted as P, from the available tiers. Subsequently, for each split point within P, we compute a score utilizing a weighted sum, considering various

Algorithm 1: Assistance Target Selection

T **Input :** S- sampled clients , C_W- weak clients, $layer_c$- cut layer, T_{comp}- weak client comp time, T_{comm}- weak client comm time, T_{compt}- comp time of offloading point, T_{commt}- comm time of offloading point

Ouput : $target$-selected offloading point

`SelectTargetpoint`$(S, C_w, layer_c)$

$attributes = 3$;

$neighbours \leftarrow AssociatedNodes\ (C_W)$;

$P = [Edge, Cloud, neighbours]$;

Y= [[0 in len($attributes$)][0 in len(P)]];

for n *in* P **do**

  for j *in range(attributes)* **do**

    $Y[n][j] = Energy(T_{comp}, T_{comm}, layer_c)$;

    $Y[n][j] = Energy_{sp}(n, T_{compt}, T_{commt}, layer_c)$;

    $Y[n][j] = PowerClient(n)$;

    $scores[n] \leftarrow Score\_allocation$(Y);

  end

end

$index \leftarrow max\_score\_index(scores)$;

$target = P[index]$;

return $target$

factors influencing the offloading decision. Each factor is assigned a weight reflecting its significance in the decision-making process, which can be tailored according to specific application needs and circumstances. Consequently, the offloading point with the highest score within the pool is selected as the target offloading point.

Therefore, as depicted in Fig. 2, the server maintains continuous awareness of various factors' statuses, such as battery levels and training velocities across all clients. These dynamic attributes may change, as clients allocate portions of their computational resources during offloading, affecting their battery consumption. With comprehensive access to each client's online profiles, the server discerns vulnerable clients susceptible to incomplete training sessions due to low battery reserves (i.e., dropouts) or sluggish computational performance (stragglers).

In summary, the server employs a two-step process to identify a viable offloading destination. Initially, to delineate intricate edge device networks, a random graph is constructed to illustrate device connectivity, thereby compiling a roster of prospective offloading destinations for each vulnerable client. Subsequently, using a weighted sum, each potential destination is assigned a score derived from observations of various tier attributes (e.g., energy consumption, recharging potential, and latency). The process is concluded by selecting the destination with the highest score.

The proposed methodology adjusts the training to accommodate the dynamic characteristics of neighbors and edge/cloud servers. This adaptation involves orchestrating collaborative efforts among these entities, enabling energy conservation for vulnerable clients. Furthermore, employing the weighted sum model technique enables the methodology to customize the weights assigned to various factors in alignment with specific application demands. This customization facilitates the prioritization of a particular offloading point, thereby enabling the methodology to identify an optimal solution within a multi-dimensional design space, offering the most favorable cost-benefit tradeoff for the terminal device.

Table 1: Mobile device specification

Device	Average Power (W)	Perf/W	Memory	Battery Capacity
Huawei P9 (Kirin 955) (Low-end)	2.98	3.55 fps/W	3GB(RAM)	3000mAh
Nexus 6P (Snapdragon 810 v2.1) (Mid-range)	5.44	4.03 fps/W	3GB(RAM)	3450mAh
Huawei Mate 10 (Kirin 970) (High-end)	6.33	5.94 fps/W	4GB(RAM)	4000mAh

3.3 DNN splitting and offloading

Upon selection of the optimal offloading point, the server notifies the weak client to commence the offloading process. Our proposed methodology, *EAFL+*, integrates SplitNN, which entails partitioning a neural network into sub-networks for separate training across distributed entities, such as clients and servers [14, 36]. Clients and servers undertake the responsibility of the client-side and server-side portions of the network, respectively. Communication is facilitated through the activation of smashed data at the split layer, known as the **cut layer**, and the transmission of gradients from server-side operations.

For instance, in this work, we segment the ResNet model into two sub-networks at the third convolutional layer, denoted as **C3**. The weak client trains the first sub-network, while the selected offloading target trains the second sub-network. This approach enables each weak client to use only a subset of layers, with major computations performed at the offloading target. The selection of the target ensures that it remains oblivious to clients' data, enhancing privacy protection. Moreover, split computing enhances communication efficiency as it exchanges only the activations and gradients of the smashed data [36].

4 EVALUATION

We evaluate *EAFL+* and compare it with the state-of-the-art Oort [20], *EAFL* [5], and baseline Random selection methods.

Experimental Setup In our experimentation, we emulate a Federated Learning (FL) benchmark designed for speech recognition tasks, utilizing the Google Speech dataset [39] and ResNet model [17].

Device Profiles: We assess the communication and computation profiles of each device utilizing authentic device metrics from the AI Benchmark [6] and MobiPerf [22] benchmark. However, due to the limited availability of widely accessible energy consumption data, we categorize these profiles into three primary performance tiers (high, mid, and low) and associate each device with one of them. As shown in Table 1, three smartphones are chosen to exemplify the high, mid, and low-end categories, along with average power consumption metrics from GFXBench [15] and device specifications.

Data Partitioning: The client-to-data mappings used in Oort [20] are close to an IID distribution, so we introduce a more realistic non-IID distribution. The learners are assigned data samples from a random 10% of the labels (4 out of 35) while the data points per learner are sampled uniformly.

The simulation is event-driven wherein time progresses with the completion and communication times of learners. Training of the model takes place on a GPU server cluster, interleaving time intervals among individual learners. Each experimental iteration

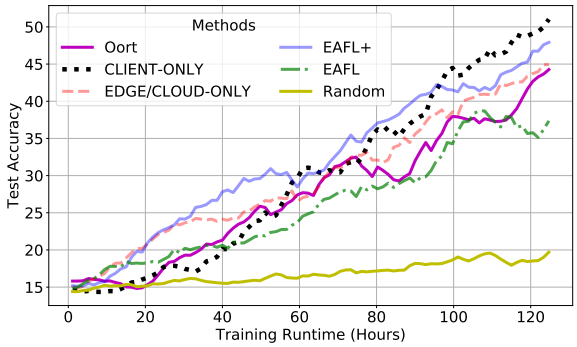

Figure 3: Comparison of testing accuracy vs time between Oort, Random, *EAFL* and all versions of *EAFL+*

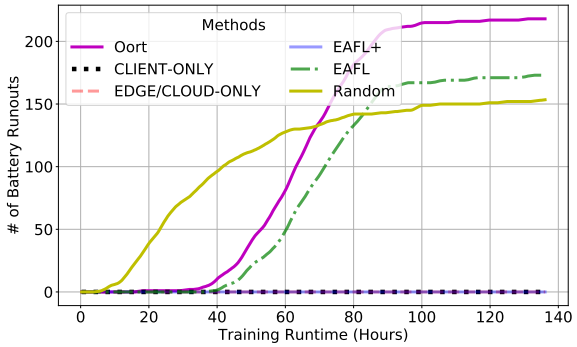

Figure 4: Devices experiencing battery runouts comparison of Oort, Random, *EAFL*, and *EAFL+* (and its variants). *EAFL+* has a lower battery runouts metric.

leverages 2 Nvidia P100 GPUs, PyTorch as the training framework, and YoGi as the aggregation protocol [32]. The Hyper-parameters are set as follows: a learning rate of 0.05, 500 epochs, and a batch size of 20. The target count of learners per round stands at 10, contingent upon client availability. Given the dynamic nature of battery charging patterns, client accessibility may fluctuate when batteries deplete, simulating practical scenarios more closely. Energy consumption during training is computed for selected devices using a power model akin to *EAFL* [5], considering energy consumed in a combination of idle or busy states for unselected devices. Our evaluation aligns energy and latency metrics with previous work [14], which explored SplitNN performance on Raspberry Pi devices while varying the number of split layers from 1 to 3.

In contrast to *EAFL* [5], *EAFL+* assigns a higher weight to the reward function to enhance statistical efficiency, as the assisting node alleviates the weak client's workload, conserving energy. Consequently, we set the parameter $f = 0.75$ to prioritize the statistical utility of the clients. To validate our approach, we perform simulations in a Mobile Edge Computing (MEC) environment, employing a realistic voice-recognition machine learning task with the GoogleSpeech dataset. Specifically, we model a cellular network comprising an edge server, a cloud, and K = 1000 clients. Clients are uniformly distributed within the cell, and the distances for offloading to the cloud, edge, or neighboring nodes are set to 1x, 0.5x, and 0.25x of the cloud distance, respectively.

4.1 Experimental Results

We introduce the results for FL setup with a **charging** scenario where devices can recharge. Specifically, in the experiments, we assume clients in the experimental setup will recharge once they reach a certain low-battery level. We compare *EAFL+* against the state-of-the-art Oort and *EAFL*, a random sampler (Random) and its other variants `CLIENT-ONLY` (offloading points picked from neighbors tier only) and `EDGE/CLOUD-ONLY` (offloading points picked from the servers tier only).

First, the testing accuracy trends over the training duration are depicted in Fig. 3. It is evident that the proposed *EAFL+*, along with its variations, achieves superior model quality concerning both training and test accuracy. This improvement is attributed to the effective power-aware offloading approach, particularly beneficial in power-constrained settings. Notably, *EAFL+* demonstrates a remarkable 24% enhancement in testing accuracy compared to the previous energy-efficient design, denoted as *EAFL*. Furthermore, when compared to Oort, *EAFL+* showcases a 9% enhancement in testing accuracy.

Fig. 4 illustrates the comparison of battery run-outs between *EAFL+* and other baseline methods. In this context, it is evident that *EAFL+* and its variations, namely `CLIENT-ONLY` and `EDGE/CLOUD-ONLY`, effectively eliminate battery run-outs. Conversely, the remaining methods, including *EAFL*, exhibit lower overall dropouts compared to the non-charging scenario. However, Oort demonstrates a notably high level of dropouts, whereas Random exhibits comparatively lower dropout rates. These findings underscore the significance of the energy-aware offloading approach proposed in *EAFL+*.

5 RELATED WORK

Federated Learning (FL) has emerged as a novel distributed machine learning approach renowned for its privacy-preserving and low-communication attributes. Its efficacy in safeguarding privacy and minimizing communication overhead has spurred its increasing adoption, aiming to enhance end-user experiences, exemplified by advancements in search suggestion quality for virtual keyboards [40]. Furthermore, the proliferation of FL frameworks aims to expedite the exploration of novel concepts and techniques in this domain. FL involves the distribution of global model training tasks among a subset of decentralized devices, such as mobile or IoT devices, which harbor private data and participate in model training based on their local datasets [7, 24].

System Heterogeneity: The inherent heterogeneity observed in many distributed systems significantly contributes to the unpredictability of system performance. Specifically within the Federated Learning (FL) paradigm, the diverse system configurations of devices, encompassing computation, communication, and battery capabilities, lead to performance unpredictability. Notably, slow workers, known as stragglers, can considerably prolong the training process [1, 21]. Various system and algorithmic solutions have been proposed to mitigate this issue [1, 21]. Additionally, in FL, heterogeneity extends beyond device characteristics to encompass other factors such as learner data distributions, participant selection methods, and device owner behavior, all of which contribute to the complexity of the FL environment [2, 7].

Energy-conservation: Given the uncertainties inherent in the mobile environment, various strategies have been devised for energy management [5, 11]. Furthermore, select studies have directed attention towards resource-aware FL training [3, 19].

Improvements in FL: In FL, several works aim to improve training time-to-accuracy by leveraging techniques such as periodic updates, adaptive compression, and asynchronous updates [1, 3, 5, 7].

Split Computing: has arisen as a promising technique for enhancing the efficacy of deep learning models, particularly on devices with limited resources [14, 23]. Nonetheless, the effectiveness of split computing is intricately linked to the choice of the cut layer, which can influence various metrics, including accuracy, communication, and computation overhead. Consequently, expertise is essential for identifying the optimal cut layer that balances efficiency and performance trade-offs. While prior studies inform the selection of the cut layer, expert input is indispensable in guiding this decision-making process.

6 CONCLUSION

In this work, we supplement existing research with the introduction of a novel resource-aware Federated Learning (FL) framework, termed *EAFL+*. This scheme employs a dynamic offloading strategy to address system heterogeneity and bolster FL effectiveness. *EAFL+* demonstrates significant enhancements, achieving a 24% and 9% increase in model accuracy compared to *EAFL* and Oort, respectively, by mitigating dropouts and stragglers. Additionally, our findings indicate a reduction in overall energy consumption. However, further investigation into energy efficiency is imperative for widespread FL adoption in practical battery-operated environments.

REFERENCES

[1] Ahmed M. Abdelmoniem and Marco Canini. 2021. Towards Mitigating Device Heterogeneity in Federated Learning via Adaptive Model Quantization. In *ACM EuroMLSys*.

[2] Ahmed M. Abdelmoniem, Chen-Yu Ho, Pantelis Papageorgiou, and Marco Canini. 2023. A Comprehensive Empirical Study of Heterogeneity in Federated Learning. *IEEE Internet of Things Journal* 10, 16 (2023), 14071–14083.

[3] Ahmed M. Abdelmoniem, Atal Narayan Sahu, Marco Canini, and Suhaib A. Fahmy. 2023. REFL: Resource-Efficient Federated Learning. *ACM EuroSys* (2023).

[4] Apple. 2022. "Siri". https://www.apple.com/siri.

[5] Amna Arouj and Ahmed M. Abdelmoniem. 2022. Towards Energy-Aware Federated Learning on Battery-Powered Clients. In *ACM FedEdge Workshop*.

[6] AI Benchmark. 2021. Performance Ranking. https://ai-benchmark.com/ranking.html

[7] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečný, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. 2019. Towards Federated Learning at Scale: System Design. In *MLSys*.

[8] K. Bonawitz, F. Salehi, J. Konečný, B. McMahan, and M. Gruteser. 2019. Federated Learning with Autotuned Communication-Efficient Secure Aggregation. In *53rd Asilomar Conference on Signals, Systems, and Computers*. 1222–1226.

[9] Chip 1 Exchange. 2022. The Wave of Wearables. https://www.eetasia.com/the-wave-of-wearables/.

[10] Shuiguang Deng, Hailiang Zhao, Weijia Fang, Jianwei Yin, Schahram Dustdar, and Albert Y Zomaya. 2020. Edge intelligence: The confluence of edge computing and artificial intelligence. *IEEE Internet of Things Journal* 7, 8 (2020).

[11] Ning Ding and Y. Charlie Hu. 2017. GfxDoctor: A Holistic Graphics Energy Profiler for Mobile Devices. In *Proceedings of the European Conference on Computer Systems (EuroSys)*.

[12] Ilias Driouich, Chuan Xu, Giovanni Neglia, Frederic Giroire, and Eoin Thomas. 2022. Local Model Reconstruction Attacks in Federated Learning and their Uses. *arXiv preprint arXiv:2210.16205* (2022).

[13] Keith Bonawitz et al. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[14] Yansong Gao, Minki Kim, Sharif Abuadbba, Yeonjae Kim, Chandra Thapa, Kyuyeon Kim, Seyit A Camtepe, Hyoungshick Kim, and Surya Nepal. 2020. End-to-end evaluation of federated learning and split learning for internet of things. *arXiv preprint arXiv:2003.13376* (2020).

[15] GFXBench. 2022. GFXBench 5.0. https://gfxbench.com/result.jsp.

[16] Google. 2016. Google gboard. https://apps.apple.com/us/app/gboard-the-google-keyboard/id1091700242.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*.

[18] Ahmed Imteaj, Urmish Thakker, Shiqiang Wang, Jian Li, and M Hadi Amini. 2021. A survey on federated learning for resource-constrained IoT devices. *IEEE Internet of Things Journal* 9, 1 (2021), 1–24.

[19] Young Geun Kim and Carole-Jean Wu. 2021. AutoFL: Enabling Heterogeneity-Aware Energy Efficient Federated Learning. *ArXiv 2107.08147* (2021).

[20] Fan Lai, Xiangfeng Zhu, Harsha V. Madhyastha, and Mosharaf Chowdhury. 2021. Efficient Federated Learning via Guided Participant Selection. In *USENIX OSDI*.

[21] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated Optimization in Heterogeneous Networks. In *MLSys*.

[22] M-Lab. 2021. MobiPerf: an open source application for measuring network performance on mobile platforms. https://www.measurementlab.net/tests/mobiperf/

[23] Yoshitomo Matsubara, Marco Levorato, and Francesco Restuccia. 2022. Split computing and early exiting for deep learning applications: Survey and research challenges. *Comput. Surveys* 55, 5 (2022), 1–30.

[24] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*.

[25] D. Milojicic. 2020. The Edge-to-Cloud Continuum. *IEEE Computer* 53, 11 (2020), 16–25.

[26] Dinh C Nguyen, Ming Ding, Pubudu N Pathirana, Aruna Seneviratne, Jun Li, and H Vincent Poor. 2021. Federated learning for internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials* 23, 3 (2021), 1622–1658.

[27] Van-Dinh Nguyen, Shree Krishna Sharma, Thang X Vu, Symeon Chatzinotas, and Björn Ottersten. 2020. Efficient federated learning algorithm for resource allocation in wireless IoT networks. *IEEE Internet of Things Journal* 8, 5 (2020), 3394–3409.

[28] Junjie Pang, Yan Huang, Zhenzhen Xie, Qilong Han, and Zhipeng Cai. 2020. Realizing the heterogeneity: A self-organized federated learning framework for IoT. *IEEE Internet of Things Journal* 8, 5 (2020), 3088–3098.

[29] Jihong Park, Sumudu Samarakoon, Mehdi Bennis, and Mérouane Debbah. 2019. Wireless network intelligence at the edge. *Proc. IEEE* 107, 11 (2019), 2204–2239.

[30] Xingchao Peng, Zijun Huang, Yizhe Zhu, and Kate Saenko. 2020. Federated Adversarial Domain Adaptation. In *International Conference on Learning Representations (ICLR)*.

[31] Xinchi Qiu, Titouan Parcollet, Daniel J. Beutel, Taner Topal, Akhil Mathur, and Nicholas D. Lane. 2020. A first look into the carbon footprint of federated learning. *ArXiv 2010.06537* (2020).

[32] Swaroop Ramaswamy, Om Thakkar, Rajiv Mathews, Galen Andrew, H. Brendan McMahan, and Françoise Beaufays. 2020. Training Production Language Models without Memorizing User Data. *arXiv 2009.10031* (2020).

[33] Thomas Rausch and Schahram Dustdar. 2019. Edge intelligence: The convergence of humans, things, and ai. In *2019 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 86–96.

[34] Amirhossein Reisizadeh, Isidoros Tziotis, Hamed Hassani, Aryan Mokhtari, and Ramtin Pedarsani. 2021. Straggler-Resilient Federated Learning: Leveraging the Interplay Between Statistical Accuracy and System Heterogeneity. In *International Workshop on Federated Learning - ICML*.

[35] Daniel Rosendo, Alexandru Costan, Patrick Valduriez, and Gabriel Antoniu. 2022. Distributed intelligence on the Edge-to-Cloud Continuum: A systematic literature review. *J. Parallel and Distrib. Comput.* 166 (2022), 71–94.

[36] Chandra Thapa, Pathum Chamikara Mahawaga Arachchige, Seyit Camtepe, and Lichao Sun. 2022. Splitfed: When federated learning meets split learning. In *Proceedings of AAAI Conference on Artificial Intelligence*. 8485–8493.

[37] C. Wang, Y. Yang, and P. Zhou. 2021. Towards Efficient Scheduling of Federated Mobile Devices Under Computational and Statistical Heterogeneity. *IEEE Transactions on Parallel & Distributed Systems* 32, 02 (2021), 394–410.

[38] Heqiang Wang and Jie Xu. 2021. Friends to Help: Saving Federated Learning from Client Dropout. *ArXiv 2205.13222* (2021).

[39] P. Warden. 2018. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *ArXiv 1804.03209* (2018).

[40] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. 2018. Applied Federated Learning: Improving Google Keyboard Query Suggestions. *arXiv 1812.02903* (2018).

[41] Tuo Zhang, Lei Gao, Chaoyang He, Mi Zhang, Bhaskar Krishnamachari, and A Salman Avestimehr. 2022. Federated learning for the internet of things: applications, challenges, and opportunities. *IEEE Internet of Things Magazine* 5, 1 (2022), 24–29.