Multi-mode Learning: Not One Learning Mode is Strictly Better than the Other

Harshit Daga
Georgia Institute of Technology

Ada Gavrilovska
Georgia Institute of Technology

ABSTRACT

In contrast to federated learning, collaborative learning aims to reduce data transfer caused by frequent model updates by creating lightweight tailored models at the edge nodes. If the model needs to adapt due to environmental changes such as drift in workload characteristics, collaborative learning offers a head start by transferring knowledge from other edges that have experienced similar patterns. This learning paradigm presents a distinct point in a design space of distributed learning, creating opportunities for different tradeoffs. In this paper, we demonstrate that neither learning mode is universally superior (strictly better than the other), discuss the characteristics of the problem space that favor one model over the other, and make a case for a future system that enables seamless support for multi-mode learning.

1 INTRODUCTION

The core value proposition of edge computing relies on bringing data analysis closer to the source to make intelligent decisions in real time. To learn from the geo-distributed data, different machine learning methods (learning modes) have emerged, such as distributed, federated and collaborative learning. Currently, developers decide on a specific learning mode, and all the nodes in the system stay in that mode. The decision to choose a learning mode depends on several external conditions, such as input data characteristics, and infrastructure resources such as compute, memory and network costs. Existing machine learning systems allow only one learning mode for the entire system and do not consider the different characteristics around each node or a set of nodes in the system. These temporal and spatial variations in the characteristics can impact individual nodes, such as degrade the model performance or result in frequent model update or knowledge transfer across nodes, in federated vs. collaborative learning, respectively. Thus, in a system consisting of large number of distributed nodes the learning mode may need to be changed over time, or not all nodes should be in the same learning mode.

Today, a dominant mode of distributed learning is *federated learning* (FL) [12]. FL leverages computational resources closer to the sources, such as at many distributed server locations in the edges of the network [18, 30, 31], to learn in place from localized data and summarize the changes as a model update. Model updates from participants are aggregated in a centralized location to improve a global model. The model update is then shared through periodic communication with the distributed participants using a *push* based approach. However, highly distributed environments, particularly such as those observed across nodes in mobile edge computing (MEC) [1], have been shown to benefit from models tailored to the context of specific locations [24, 26, 32]. In addition, distinct locations may not exhibit any change in their localized inputs, thus not benefiting from model updates due to churn in data trends observed at other locations. Abandoning any cross-node coordination in learning in favor of *learning in isolation*, eliminates data transfer costs. However, if data trends shift across locations (i.e., due to concept or data drift), models learned in isolation suffer from loss of accuracy and need more time to adapt to changes, compared to when relying on a global model.

In response, distributed machine learning models for *collaborative learning* across edge nodes have been proposed, as a way to create robust tailored models. Collaborative learning (CL) allows distributed nodes to adapt quickly to input shifts and to retain model accuracy, with low data transfer costs [5, 6, 18]. The key idea in CL is that participating nodes learn in isolation, but upon detecting input drift and degradation of the local model performance, they ask for help and *pull* a (partial) model update from one of the other nodes in the collaborative learning system. Much like in relaxed consistency models, relying on an on-demand pull method allows for reduction in data movement costs (for model aggregation and updates), but with a potential impact on model staleness and accuracy convergence.

This *push* vs. *pull* distinction presents a fundamental difference between these two learning modes and exposes new tradeoffs in training performance and costs. We hypothesize that changing the learning mode could potentially reduce

FedEdge '23, October 6, 2023, Madrid, Spain

ACM ISBN 979-8-4007-0344-7/23/10.
https://doi.org/10.1145/3615593.3615722

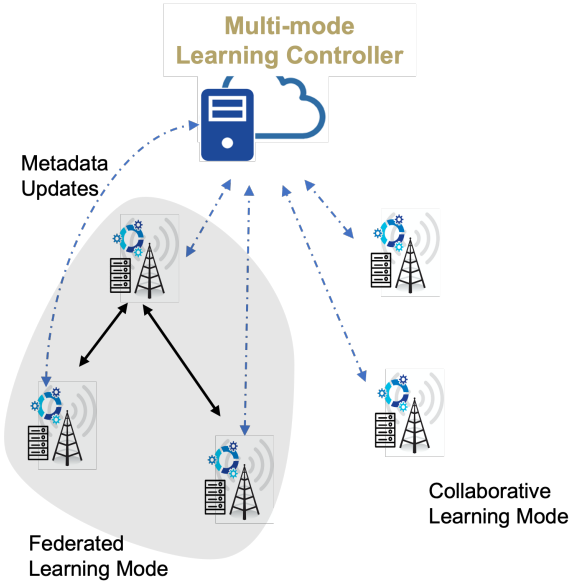


Figure 1: A multi-mode learning system supports the co-existence of different learning modes, and the transition of nodes from one learning mode to another.

the overall data transfer while attaining similar model performance in certain distributed training scenarios. To support this, in this paper we discuss the tradeoffs between these learning modes. In particular, we identify specific scenarios which favor one mode over the other. We observe that real systems may transition across these different scenarios, or localized parts of the system may exhibit different characteristics. This motivates the need for systems which allow for multi-mode learning, permit different parts of the system to be configured with a different learning mode, and enable online transition of (groups of) nodes from one learning mode to another. While discussing these scenarios, we observe several requirements for such future systems.

2 FEDERATED VS. COLLABORATIVE LEARNING

Multi-access Edge Computing. MEC provides compute capabilities at the edge of the network, in close proximity to client devices, such as on wireless gateways, cellular base stations, enterprise on-prem edge server infrastructure, etc. [28]. It offers infrastructure for execution of applications requiring low latency (e.g., AR/VR, automation), reduction in backhaul data demand (e.g., IoT analytics, content delivery), and geo-localized data handling (e.g., for GDPR). These base stations are commonly equipped with servers that typically possess multi-core processors, ample memory capacity, and fast storage subsystems [2, 3]. To further enhance their performance, they may also be outfitted with hardware accelerators such as GPUs, Edge TPUs, and FPGAs. One challenge

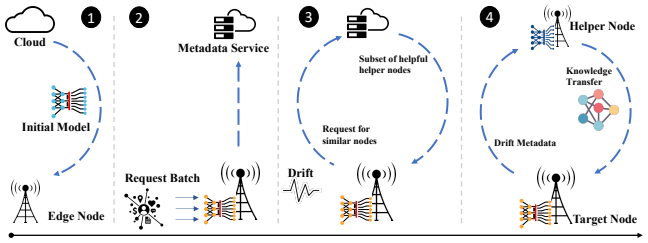


Figure 2: Collaborative learning in distributed edge computing.

MEC exposes is that of creating an extremely dispersed distributed infrastructure, which differs from datacenter-based distributed environments in its scale and in the fine-grained geo-localization of the inputs observed at each node [26]. In such settings, COLLA [18], a collaborative system to predict user behavior patterns, highlights that a shared global model might not achieve the desired performance for all the users and trains smaller more tailored models. Cellscope [24] also demonstrated the ineffectiveness of using a shared global model for managing base stations in a mobile network; similar observations are confirmed in other research [25, 32].

Federated Learning. Federated learning (FL) has emerged as the state-of-the-art technique for distributed machine training. FL offers improved training performance, data privacy, and reduced communication and data transfer costs compared to other distributed ML approaches like centralized learning. It enables training of models on decentralized data sources without the need for data aggregation in a central server. The models are trained locally on individual devices or edge nodes, and only aggregated updates are shared. There are many variants of federated learning. Many focus on algorithms for improving fairness, accuracy and convergence time [15, 21, 22, 27]. Others propose to select clients intelligently [13, 23] or to carefully sample a client's dataset [29] for faster convergence. Beyond classical FL, where all clients talk directly to a centralized aggregation server, other approaches have emerged that are based on different connectivity patterns among participating nodes, such as hierarchical [16, 19], hybrid [9] and peer-to-peer FL [14].

Collaborative Distributed Learning. Collaborative learning shares similarities with FL as it also trains models locally without sharing raw data and aims to reduce data transfer costs. However, CL exploits edge locality to further minimize data movement, while offering comparable model performance to FL [4]. Figure 2 illustrates a typical workflow in a collaborative distributed learning system, similar to what is enabled by our prior work Cartel [6]. Edge nodes use their resident model to perform inference, following which the model is re-trained locally. Upon shift in the data distribution, the model performance at the edge node deteriorates. In such a scenario, the node relies on a centralized metadata

Table 1: Raw data transferred (MB) and gains per *update* per node in FL vs. CL. In Federated Learning data is transferred to send and received model update while in Collaborative Learning data is transferred to share metadata information and during knowledge transfer.

CNN Model	Million Parameters	Federated Learning		Collaborative Learning		Data Transfer Gains	
		Out	In	KT (In)	MdS (Out)	No Drift (×)	With Drift (×)
MobileNet	3.54	6.38	7.36	4.57	0.00058	23684	3
DenseNet	8.06	16.18	18.35	8.80	0.00058	59507	4
ResNet	11.69	12.19	13.75	8.55	0.00058	44713	3

service to find a helper node, and to leverage the helper's model (knowledge) to boost its own model [5]. The common metadata service (MdS) aggregates information from each node in the system, that helps the system differentiate the nodes. Each edge node uses drift detection algorithms [8, 11] on the local metadata information to decide when a model at the target edge node needs to be updated, and to query the MdS for candidates for a possible helper. The MdS uses the aggregated metadata to determine helper node candidates using similarity detection algorithms.

The workload characteristics shared with MdS is a histogram or trend of the characteristics observed by the nodes, which is then used to determine logical neighbor(s) which have seen similar workload patterns. This makes it possible for collaborative learning to proceed without exchange of raw data among edge nodes. We assume that collaboration is done across distributed nodes part of the same service, where the primary goal is reducing data transfer costs, and sharing information regarding data trends at specific locations is not a major concern. For instance, it would allow edge nodes managing base station configurations or distributed MEC services and applications to share information.

To update the target node's model and improve its predictive performance, a collaborative system uses knowledge transfer [5, 6] or distillation [18]. Knowledge distillation [10] uses a (larger) teacher model to train a (smaller) student model. Knowledge transfer (KT) is a mechanism to transfer the knowledge regarding several (typically not all) classes from one machine learning model to another. We use knowledge transfer mechanism to collaborate among edge nodes, because in distillation the teacher model is assumed to be a complete model, whereas, we already point out, a complete global model may not be needed for MEC, and it may be too costly to maintain it at large scale in a highly distributed settings.

Benefits of CL over FL. Our prior work with CL demonstrates opportunities for drastic reduction in the data transfer costs associated with distributed training, both for traditional machine learning techniques [6] and for deep neural networks [4]. Data transfer is a critical metric, particularly when considering widely distributed MEC settings. In Table 1, we include a single result to illustrate the scope of the possible gains. In FL, for each update period, for a member node the data transfer cost consists of the delta in the local model since the last batch (*Out*) and the received aggregated model update (*In*). For CL, the per-update data transfer cost at each node consists of the metadata sent to the MdS server (*MdS(Out)*). Additional knowledge transfer cost (*KT*) are incurred only when data drift is detected on a node, and for that node only. For neural networks, the change in a local model (*Out*) or aggregated model updates (*In*) are significantly larger when compared to the metadata (few kBs) used by CL, and this leads to drastic reductions in the data transfer costs, compared to FL, up to $10^4\times$, as seen from the table. Moreover, since only a fraction of model parameters are transferred as part of knowledge transfer in CL, this too could result in data saving of up to 4× when compared to a single model update request in FL. The precise reduction in data transfer is impacted by the model, the number of participating nodes, the frequency of drift occurance, the details of the participant selection algorithm used as part of FL [13], and other factors; our evaluation shows opportunities for reduction of 2-3 orders of magnitude [4, 6]. This makes CL an important learning mode for settings where the network can be a bottleneck.

3 FACTORS IMPACTING THE MODE CHOICE

CL shows tremendous opportunities for data transfer reductions by exploiting the data trend localization property of distributed edge environments. These benefits are not totally surprising, as we have observed similar trends in other settings when switching from a proactive push model for state distribution, to an on-demand pull model. However, strictly favoring CL over FL is not sufficient. We demonstrate that in this section using several concrete scenarios, and make the case for future system support for multi-mode learning.

3.1 Not Enough Data

Edge nodes use online learning to update the models incrementally as new data becomes available. Due to the highly geo-distributed nature of the MEC certain edge nodes might not receive sufficient data to retrain an accurate model. Absence of sufficient data could impact the quality of the trained

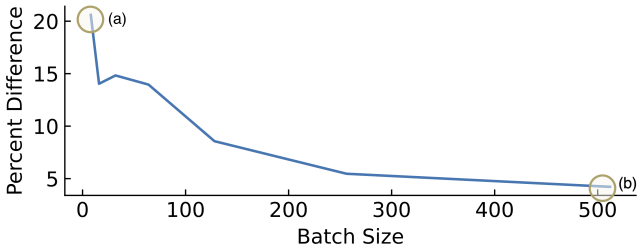
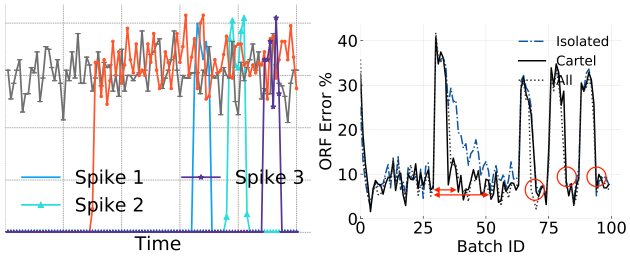


Figure 3: Percentage difference between federated and collaborative learning model error rate as request batch size changes. (a) For lower batch sizes federated learning mode can improve the models performance by up-to 20% when compared to collaborative learning. However, with more data (b) comparable model performance can be achieved by both learning modes.

model. We evaluated the model performance, performed for 100 request batches on the EMNIST dataset using the MobileNet CNN. We varied the number of request in each batch from 8 to 512. As illustrated in Figure 3 for smaller batch sizes the FL mode can improve the model's performance by up to 20% compared to CL.

Since low volume of data leads to slower convergence rate, and an insufficient level of model accuracy, in such scenarios, combining nodes with similar workload distribution to build models in federated mode would help improve the model quality. As the workload characteristics and distribution can differ from node to node concerning the classes and/or the number of requests associated with a class, a more efficient design would be to create groups of nodes with similar data distribution. This approach can exploit the similarity among nodes (e.g., using similar methods as what is used in CL), and help reduce the cost of data transfer [20].

Current studies on data trends in MEC settings provide evidence of continuously changing patterns in terms of data volumes and distributions across different base station locations and over time. For instance, Iyer et al. in [24], showed that the skewed traffic patterns of data observed at the edge causes some nodes to wait a long time to collect enough data to build a sufficiently accurate pattern. Similar trends of data volume variability can be observed for different examples of MEC applications, which have common diurnal (or double diurnal) patterns. It is therefore easy to argue that concerning data volume as an edge node property, edge nodes may transition between low to high data volume state over time, and will therefore exhibit preference for different data modes over time. A future system can accommodate the management of such transitions via support for threshold-based methods for tracking data volume trends at each node, and via support for clustering techniques that can effectively group similar nodes in a federated learning cluster.



(a) Spike workload **(b) Model Performance**

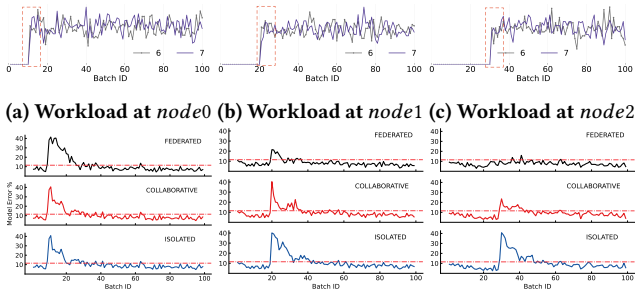
Figure 4: Collaborative learning mode is impacted by the successive introduction of new classes at edge nodes. Figure (a) shows the spike workload, while Figure (b) presents the corresponding model performance. The model performance is evaluated for three different approaches, 'All,' 'Cartel,' and 'Isolated.' The 'All' approach, which requests help for all consecutive spikes in the workload distribution, does not necessarily lead to improved model performance when compared to ignoring the spike ('Cartel') or just learning in isolation ('Isolated').

3.2 Temporal and Spatial Drift Propagation

The characteristics of the data typically change over time, which leads to system drift. This might be because of weekly and seasonal patterns that introduce both high- and low-frequency drift, while catastrophes and upgrades cause sudden drift. This is discussed in the recent work of Liu et al. [17] where they discuss the gradual and sudden deviations that occur at different nodes in the system. Another study of base station logs by AT&T [26] reports consistent daily patterns where each base station exhibits unique characteristics over time.

A change in the workload distribution at an edge node represents a concept drift, in response to which the model performance is impacted. Consider the workload in Figure 4a, and the corresponding model performance in Figure 4b. In response to the first spike, the model performance deteriorates, but CL, shown by the Cartel line, is able to adapt the model more quickly compared to if modes were learning in isolation. However, the subsequent burst of spikes (spikes 1-3 in the figure), appear too close to each other, making the on-demand, one-by-one model update triggered by CL not as effective as when model updates are proactively pushed to all nodes.

Another example shows the effects of spatial distribution of a drift. The top graphs in Figure 5 show the workload at three different nodes. Below we show the corresponding model accuracy for when the nodes are learning in isolation vs. via CL or FL. Edge *node*0 observes a drift due to introduction of a new class 5a). Over a short period of time, similar change in workload distribution is observed in other parts of the system (*node*1 5b and *node*2 5c). When considering an isolated change at a single, or small number of nodes, the



(a) Workload at *node*0 (b) Workload at *node*1 (c) Workload at *node*2

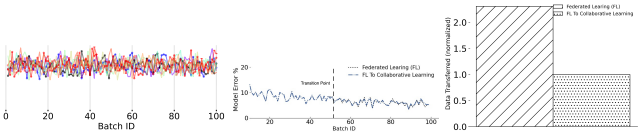
(d) Edge *node*0 (e) Edge *node*1 (f) Edge *node*2

Figure 5: Impact on node's model performance where drift occurs at *node*0, *node*1, and *node*2 successively. Initially, all nodes learn collaboratively, which aids in adapting to changes. However, as drift propagates, this would result in repeated model degradations and updates at the successive nodes. Using FL would be more effective in immediately delivering updates to *node*1 and *node*2, and improving overall model performance.

data transfer reductions provided by CL outweigh the downside of any impact on accuracy. When the drift propagates to a larger portion of the system, the repeated on-demand model updates triggered by each of the affected nodes reduce the data transfer benefits, compared to FL, while at the same time, the proactive global model update in FL is able to eliminate the negative impact on the model as the drift propagates. In such settings, choosing FL is better than CL.

This type of spatial and temporal drift propagation characteristics can be common in edge data settings, and makes the case for transitioning the system (or a subset of the nodes in the distributed system), from a CL to a FL mode. Future systems can provide support to detect such drifts, using metadata regarding model performance and input distributions. For instance, the systems control plane logic can analyze the rate of change in model performance over the last W batches at each node; a negative change in model accuracy indicates a drift in the system. If this drift propagates to multiple nodes within a short time interval, it may be better to configure these nodes in federated learning mode. This way, knowledge updates (i.e., model updates) are proactively pushed to all nodes before they are affected.

Drift propagation trends could also be predicted using historic data of drift occurrences across the nodes. One simple approach is to use a window-based policy where drift history over the last W batches is compared to determine which nodes are experiencing negative changes. Sophisticated machine learning models have been shown effective in trend analysis and prediction in other settings, and may be effective for this use case as well. Once a drift propagation is estimated, the affected nodes can be appropriately grouped with other nearby nodes for federated learning.



(a) Uniform Workload (b) Model Error (c) Data Transfer

Figure 6: A node with stable model error rate and no changes in workload characteristic could switch to collaborative learning mode while maintain the model performance and reducing the data transfer.

3.3 Uniform Workload Distribution

The process of configuring a learning mode within an online, dynamic system is not a static, one-time definitive decision. Instead, the system must be capable to transition seamlessly among multiple modes as needed by changing circumstances. For instance, when the model performance remains constant (or within a specific range), and the workload distribution exhibits no changes (no introduction of new classes or drift), it suggests system stability. This is the case with the workload shown in Figure 6a. In this case, if the node remains in FL mode, it will lead to an increase in data transfer of 2.4× compared to CL, as shown in 6c.

One simple approach to determining node stability is to maintain and analyze the historical trends of the rate of change for the last W batches, and to transition a node from federated learning mode to independent collaborative mode when stable conditions are detected. Additional care must be placed on considering the impact that the target node may have on other nodes which were part of its FL mode configuration. For instance, a system should not remove a node if the other nodes are not observing sufficient data.

4 SUMMARY AND OPPORTUNITIES

The observations in this paper point to an opportunity to exploit the unique benefits of different learning modes by allowing a distributed learning system to transition from one mode to another. The CL and FL configurations used in the experiments presented in this paper present concrete discrete points in the tradeoff space. By further considering the mode of individual nodes and groups of nodes, it will be possible to realize a continuum of system configurations, each favoring specific data and drift dynamics. As the demand for distributed machine learning continues to grow, it will continue to exacerbate the challenges with ingesting and moving the associated data. Future systems support that will make it possible to navigate a fine-grained continuum of accuracy, convergence time, and data movement tradeoffs, will provide critical new capabilities for the practical and efficient application of machine learning in distributed edge scenarios. Our recent work with the system Flame [7], provides the programming support for describing and deploying

the different distributed machine learning topologies that can be realized on this continuum, and can further facilitate the exploration of the opportunities of multi-mode learning.

ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers. This work has been partially supported by NSF projects CCF-2217070 and CNS-1909769, by funding and equipment gifts from Cisco, VMware, Adobe and Intel, and resources from the NSF Chameleon Cloud testbed.

REFERENCES

[1] [n.d.]. ETSI Mobile Edge Computing. https://www.etsi.org/technologies/multi-access-edge-computing.

[2] [n.d.]. PowerEdge C Servers. Proven performance and hyper-efficiency at scale. https://i.dell.com/sites/csdocuments/Shared-Content_data-Sheets_Documents/ar/dz/ESG-PowerEdge-C-Portfolio-Brochure.pdf

[3] [n.d.]. State of the Edge 2021: A Market and Ecosystem Report for Edge Computing. https://www.lfedge.org/wp-content/uploads/2021/08/StateoftheEdgeReport_2021_r3.11.pdf

[4] Harshit Daga, Yiwen Chen, Aastha Agrawal, and Ada Gavrilovska. 2021. Canoe : A System for Collaborative Learning for Neural Nets. arXiv:cs.LG/2108.12124

[5] Harshit Daga, Yiwen Chen, Aastha Agrawal, and Ada Gavrilovska. 2023. CLUE: Systems Support for Knowledge Transfer in Collaborative Learning with Neural Nets. *IEEE Transactions on Cloud Computing* (2023), 1–14. https://doi.org/10.1109/TCC.2023.3294490

[6] Harshit Daga, Patrick K Nicholson, Ada Gavrilovska, and Diego Lugones. 2019. Cartel: A System for Collaborative Transfer Learning at the Edge. In *Proceedings of the ACM Symposium on Cloud Computing*. 25–37.

[7] Harshit Daga, Jaemin Shin, Dhruv Garg, Ada Gavrilovska, Myungjin Lee, and Ramana Rao Kompella. 2023. Federated Learning Operations Made Simple with Flame. arXiv:cs.LG/2305.05118

[8] João Gama, Indre Zliobaite, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A survey on concept drift adaptation. *ACM Comput. Surv.* 46, 4 (2014), 44:1–44:37. https://doi.org/10.1145/2523813

[9] Yuanxiong Guo, Ying Sun, Rui Hu, and Yanmin Gong. 2022. Hybrid Local SGD for Federated Learning with Heterogeneous Communications. In *International Conference on Learning Representations*.

[10] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the Knowledge in a Neural Network. In *NIPS Deep Learning and Representation Learning Workshop*. http://arxiv.org/abs/1503.02531

[11] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. 2004. Detecting change in data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. VLDB Endowment, 180–191.

[12] Jakub Konečny, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492* (2016).

[13] Fan Lai, Xiangfeng Zhu, Harsha V Madhyastha, and Mosharaf Chowdhury. 2020. Oort: Informed participant selection for scalable federated learning. *arXiv preprint arXiv:2010.06081* (2020).

[14] Anusha Lalitha, Osman Cihan Kilinc, Tara Javidi, and Farinaz Koushanfar. 2019. Peer-to-peer Federated Learning on Graphs. https://doi.org/10.48550/ARXIV.1901.11173

[15] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems* 2 (2020), 429–450.

[16] Lumin Liu, Jun Zhang, SH Song, and Khaled B Letaief. 2020. Client-edge-cloud hierarchical federated learning. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 1–6.

[17] Shinan Liu, Francesco Bronzino, Paul Schmitt, Nick Feamster, Ricardo Borges, Hector Garcia Crespo, and Brian Ward. 2021. Understanding Model Drift in a Large Cellular Network. *CoRR* (2021).

[18] Yan Lu, Yuanchao Shu, Xu Tan, Yunxin Liu, Mengyu Zhou, Qi Chen, and Dan Pei. 2019. Collaborative learning between cloud and end devices: an empirical study on location prediction. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*. 139–151.

[19] Siqi Luo, Xu Chen, Qiong Wu, Zhi Zhou, and Shuai Yu. 2020. HFEL: Joint Edge Association and Resource Allocation for Cost-Efficient Hierarchical Federated Edge Learning. *IEEE Transactions on Wireless Communications* 19, 10 (2020), 6535–6548. https://doi.org/10.1109/TWC.2020.3003744

[20] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. 2020. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619* (2020).

[21] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.

[22] John Nguyen, Kshitiz Malik, Hongyuan Zhan, Ashkan Yousefpour, Mike Rabbat, Mani Malek, and Dzmitry Huba. 2022. Federated Learning with Buffered Asynchronous Aggregation. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research)*, Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera (Eds.), Vol. 151. PMLR, 3581–3607. https://proceedings.mlr.press/v151/nguyen22b.html

[23] Takayuki Nishio and Ryo Yonetani. 2019. Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE international conference on communications (ICC)*. IEEE, 1–7.

[24] Anand Padmanabha Iyer, Li Erran Li, Mosharaf Chowdhury, and Ion Stoica. 2018. Mitigating the latency-accuracy trade-off in mobile data analytics systems. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. 513–528.

[25] Utpal Paul, Anand Prabhu Subramanian, Milind Madhav Buddhikot, and Samir R Das. 2011. Understanding traffic dynamics in cellular data networks. In *2011 Proceedings IEEE INFOCOM*. IEEE, 882–890.

[26] Michele Polese, Rittwik Jana, Velin Kounev, Ke Zhang, Supratim Deb, and Michele Zorzi. 2018. Machine Learning at the Edge: A Data-Driven Architecture with Applications to 5G Cellular Networks. *CoRR* abs/1808.07647 (2018). arXiv:1808.07647 http://arxiv.org/abs/1808.07647

[27] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H. Brendan McMahan. 2020. Adaptive Federated Optimization. https://doi.org/10.48550/ARXIV.2003.00295

[28] Mahadev Satyanarayanan, Wei Gao, and Brandon Lucia. 2019. The computing landscape of the 21st century. In *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*. 45–50.

[29] Jaemin Shin, Yuanchun Li, Yunxin Liu, and Sung-Ju Lee. 2022. FedBalancer: Data and Pace Control for Efficient Federated Learning on Heterogeneous Clients. In *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services (MobiSys '22)*. Association for Computing Machinery, New York, NY, USA, 436–449. https://doi.org/10.1145/3498361.3538917

[30] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan. 2019. Adaptive Federated Learning in Resource Constrained Edge Computing Systems. *IEEE Journal on Selected Areas in Communications* 37, 6 (2019), 1205–1221. https://doi.org/10.1109/JSAC.2019.2904348

[31] Xiaofei Wang, Yiwen Han, Chenyang Wang, Qiyang Zhao, Xu Chen, and Min Chen. 2019. In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning. *IEEE Network* 33, 5 (2019), 156–165.

[32] Xuan Zhou, Zhifeng Zhao, Rongpeng Li, Yifan Zhou, and Honggang Zhang. 2012. The predictability of cellular networks traffic. In *2012 International Symposium on Communications and Information Technologies (ISCIT)*. IEEE, 973–978.