On the distribution of ML workloads to the network edge and beyond

Georgios Drainakis*, Panagiotis Pantazopoulos†, Konstantinos V. Katsaros‡, Vasilis Sourlas§, Angelos Amditis¶
Institute of Communication and Computer Systems (ICCS), Athens, Greece
Email: *giorgos.drainakis@iccs.gr, †ppantaz@iccs.gr, ‡k.katsaros@iccs.gr, §v.sourlas@iccs.gr, ¶a.amditis@iccs.gr

***Abstract*—The emerging paradigm of edge computing has revolutionized network applications, delivering computational power closer to the end-user. Consequently, Machine Learning (ML) tasks, typically performed in a data centre (Centralized Learning - CL), can now be offloaded to the edge (Edge Learning - EL) or mobile devices (Federated Learning - FL). While the inherent flexibility of such distributed schemes has drawn considerable attention, a thorough investigation on their resource consumption footprint is still missing.**

In our work, we consider a FL scheme and two EL variants, representing varying proximity to the end users (data sources) and corresponding levels of workload distribution across the network; namely Access Edge Learning (AEL), where edge nodes are essentially co-located with the basestations and Regional Edge Learning (REL), where they lie towards the network core. Based on real systems' measurements and user mobility traces, we devise a realistic simulation model to evaluate and compare the performance of the considered ML schemes under an image classification task. Our results indicate that FL and EL can act as viable alternatives to CL. Edge learning effectiveness is shaped by the configuration of edge nodes in the network with REL achieving the prominent combination of accuracy and bandwidth needs. Energy-wise, edge learning is shown to offer an attractive choice (for involved stakeholders) to offload centralised ML tasks.

I. Introduction

The unprecedented surplus of data supplied by a variety of mobile devices (smartphones, smart sensors, wearables), supplemented by ML-specific hardware acceleration has contributed to a proliferation of ML techniques in order to capitalize on the information richness and therefore develop intelligent network applications. From virtual reality [1] to video analytics [2], ML has led to an era of data-driven application development. While the current ML paradigm remains centralized (CL) in nature *i.e.*, tons of data are accumulated in a resourceful central entity *e.g.*, in a data-centre (DC) where the training occurs, the emergence of edge computing has changed the scene, providing new opportunities, such as computational load distribution towards the edge (EL). At the same time, privacy concerns have even lead to UE-based ML schemes, like Federated Learning (FL), where the devices perform the training themselves.

To facilitate efficient distributed learning over the network, in favor of the involved stakeholders interests (end-user, network operator, cloud/edge provider), a distributed ML scheme is required to optimize the ML performance (convergence speed, accuracy) subject to constraints related to resource consumption (bandwidth, energy) and environmental factors (device computational heterogeneity, client mobility patterns). In relation to the traditional approach of CL, two key points emerge: 1) What are the prons and cons of each ML scheme in relation to the traditional approach of CL in terms of performance *vs.* its respective resource consumption and 2) Depending on the workload distribution scheme, the associated costs, such as energy, are also distributed accordingly to the various network stakeholders and we lack an understanding of how this happens.

These questions have so far been addressed in a limited way. While prior work was focused on exploring the convergence capabilities of decentralized ML as in [3], where the theoretical bounds of distributed Stochastic Gradient Descent algorithm are compared against its centralized alternative, the effect of such implementations on the underlying network's resources is still missing. This also holds for emerging works that have introduced edge-based learning as an alternative to CL. In [4] the convergence rate of EL is studied and a control algorithm is designed to optimize the ML algorithm's performance, approximating that of its centralized counterpart. In [5] an edge-based algorithm is discussed to mitigate the performance loss of distributed ML, while in [6] an edge-based scheme is developed to reduce communication cost and increase convergence speed. On the other hand, research on FL-based schemes has provided some insights on the parameters that affect performance as well as the involved trade-offs, such as ML accuracy in relation to the communication costs [7], the effect of reliability [8] and accuracy (jointly) with fairness [9]. Nevertheless, an across the network parameter analysis when implementing the various ML schemes has not been performed yet.

In order to bridge this gap, we propose a system model that incorporates the cloud elements, the core network's infrastructure, the edge network's devices and the UEs, along with their throughput, computational capacity and energy consumption characteristics. The latter are taken from measurements in practical systems, to ensure realism. Moreover, the model accounts for the effect of client mobility, which is captured by mobility traces. Using our proposed model we study the performance of three basic ML schemes; namely CL, FL and EL. Addressing different degrees of load distribution/aggregation we further differentiate EL into two variants; AEL, where the edge nodes are attached to a network cell and REL, where each edge node serves multiple cells. Using a FL-simulation framework we compare the resulted classification accuracy and convergence speed of the schemes, along with their respective resource consumption in terms of system

bandwidth and energy. On top of that, we estimate how this consumption's costs are divided into the system's stakeholders, revealing the benefits of each scheme and exploring the trade-offs that emerge. This work extends our previous research [10] on the CL-FL debate, introducing EL as an alternative ML scheme, revisiting the network and energy model to support edge nodes and extending the per-stakeholder cost analysis.

In our results, REL is proven as a more efficient CL-alternative than AEL, in terms of achieved accuracy *vs.* traffic. FL shows a potential for reduced communication overhead, at a cost of slower convergence and increased energy consumption, mainly caused by the mobile devices' processing. Lastly, EL exhibits the highest energy efficiency system-wise, reducing energy costs up to 4 times both for the end-user and the cloud. The remainder of the paper is structured as follows. In Section II we analyze our system model. In Section III, our simulation results are provided. Lastly, we conclude in Section IV pointing also to future explorations.

II. System Model

In our model, a cellular network environment is assumed comprising several mobile clients, each holding an amount of training data. In each cell, a wireless link connects the clients with the basestation unit (BS). Also, BSs are able to communicate with the edge nodes and with a central data centre (DC) cloud server, via the intermediate core (wired) network (Fig. 1). The system's aims to perform a ML task, utilizing available client data[1] through the following schemes:

CL: In each training round the server selects a group of clients; they upload their data directly to the server. The training task is thereafter performed centrally by the server using the total data acquired in that round. This process repeats for several rounds, each time with a selection of a random group of clients, until a predefined time limit is reached or all data are depleted.

FL: The server communicates the training model to the selected clients (learners), which in turn are responsible to train it using their own (private) data and computing resources. As opposed to CL, they no longer require to upload their actual data. Instead, once the training is completed, they communicate the updated model parameters, which are generally considerably lightweight compared to the actual data, to the central server. Upon collecting the updated model parameters, the server performs model aggregation and (re)-distributes the updated (aggregated) model to a different group of clients.

EL: This approach functions as a middle-ground solution between CL and FL, where the edge nodes act as learners. In each round, each edge node receives the central model from the central server and the data from the clients in its service area. Thereafter, the models are trained and the respective model parameters are sent to the central server, which in turn performs the aggregation (similarly to the FL case).

A. Network model

We assume a mobile Long-Term Evolution (LTE) cellular network where a BS lies in the centre of each cell. We refer to the wireless part of the network (BS-client link) as access network, while the wired part (BS-edge-cloud link) comprises the core network (Fig. 1). We assume two distinctive cases for EL depending on the network's architecture; Access Edge Learning (AEL), where each edge node serves a specific cell, thus attached to the cell's BS and Regional Edge Learning (REL), a more appealing in practical installation terms, where few edge nodes serve the whole network.

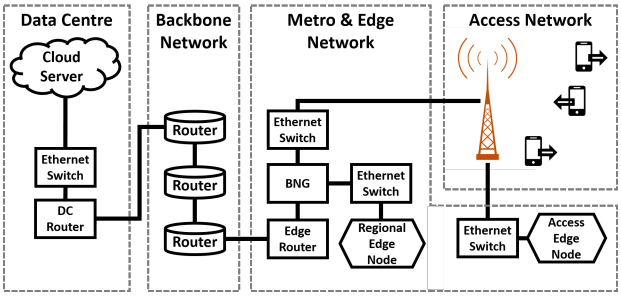

Fig. 1. From the mobile user to the cloud: end-to-end logical network diagram

Access network: The client throughput (R) in both the uplink (UL) and the downlink (DL) (in MBytes/sec) is modelled as a Gaussian random variable ($\tilde{N}$). Its mean value is assumed equal to the average cell throughput[2] C, divided by the number of online clients. Including the number of online clients in the computation, allows to factor-in the way the locally-present number of users shapes the throughput provision in the considered area. An artificial standard deviation parameter (sigma) is also introduced, equal to 20% of the mean value [12], to account for throughput variations *e.g.*, due to path loss. Given that a client is online, however, its throughput value should be forced to be positive, since some BS-client communication exists. We therefore assume a minimum throughput threshold (C_{min}), both in DL and in UL. C_{min} is assumed equal to the 5% cell edge rate[3], which represents the worst wireless conditions. Thus, the client throughput for UL and DL, is:

$$R = max\{\tilde{N}(\frac{C}{\|K_O\|}, sigma), C_{min}\} \quad (1)$$

where $\|.\|$ is a vector's norm, K_O the number of online clients.

Core network: It typically includes the following elements [13]: 1) An interface to the access network (BS); 2) The metro and edge network's elements, *i.e.*, an ethernet switch, a broadband network gateway (BNG) and the edge router; 3) the backbone network's routers and 4) the DC's elements, *i.e.*, an edge router and a data center switch. The average throughput of each element is based on Cisco routers/switches performance benchmarking [13] and measurements on a 3-sector 2×2 Multiple-Input-Multiple-Output remote radio 4G/LTE [13]. A total number of 3 core routers is considered, in line with [14], which shows that a hopcount of maximum 3 in the core network suffices to reach the DC (from the edge network) for the majority of popular services.

For the CL and FL case, no edge nodes are utilized, therefore are not considered in the network model. In AEL,

[1]Without harming the model's accuracy, control and management plane's messages *e.g.*, signalling are neglected, given their negligible size

[2]That is, 5.9 (UL)/7.73 (DL) MBytes/sec for 2.5 GHz LTE according to [11]

[3]That is, 0.24 (UL)/0.22 (DL) MBytes/sec according to [11]

edge nodes are assumed attached to the BSs via an ethernet connection. Essentially, the total number of edge nodes equals that of the BSs. Each edge node serves its BS thus no cell aggregation occurs. For REL, on the other hand, a certain level of cell aggregation is assumed, since edge nodes serve multiple BSs. We assume that 3 Regional Edge Nodes in total, attached to the BNG, sufficiently cover the area of our experiments.

B. Mobility model

Real-world traces were selected to capture clients' mobility in a realistic manner. In specific, the Shanghai Telecom Dataset [15] was used, which contains records (LTE traces) of mobile devices accessing the Internet through a BS in a period of 15 days. The database includes timestamps (taken every minute, which is the dataset's time granularity) for connection initialization and termination marking the clients' online presence. Clients are considered online as long as they remain in the network. A cell in the dataset in defined by its corresponding BS's coordinates; totally, 1853 cells are included. During the BS-client communication, if the client moves to another cell *i.e.*, serviced by another BS, we assume that a handover (HO) occurs, during which, communication is not disrupted. A change of cell, however, may affect client throughput in line with the cell's congestion (see paragraph II-A).

In an actual network, the identification of the edge node locations would ideally be the outcome of a facility location problem, accounting for various factors *i.e.*, spatial characteristics, cellular architecture, average user demand *etc.* As our baseline approach, a uniform distribution of cells across the edge nodes is considered. Although non-realistic, it is insightful as it reduces the involved problem parameters. A more skewed distribution is left for future work.

C. Client selection model

We now detail the employed client selection algorithm. The training dataset, including data and labels, is firstly shuffled and then uniformly divided into Z partitions. Then, a fixed number of per round participating clients K_N is chosen (K_N<Z). In each communication round, the server identifies all online clients (K_O), which are determined by the mobility dataset. A total of K_N clients are randomly chosen to participate in this round and each one is assigned a dataset partition. Partitions that have already been used are not reassigned. If there are not enough online clients present, to reach K_N in number, all currently online clients are chosen. In case no clients are found, a waiting period is introduced, equal to the mobility dataset granularity (60 sec). In case the algorithm selects a previously chosen client, a (new) dataset partition is still assigned. This is a valid assumption for most applications, considering that the client has a continuous data acquisition rate, *e.g.*, a vehicle gathering traffic-scene photos.

Upon partition assignment, the training procedure takes places. In the CL case, the selected clients upload their local datasets to the central server in a parallel manner. The time required for each upload is simply modelled as $Time = \sum Ti = \sum Dataset/R_i$, for all (access and core) network components (i), where R denotes the component's throughput. The time to upload all datasets equals to that of the "slowest" client, since a parallel transmission is assumed. The central server, thereafter merges the datasets into a single super-dataset and performs the ML training task, marking the end of the round. This procedure is repeated until a global time limit is reached or all datasets are used. In case a client goes offline during upload (irrespective of the upload completion percentage), we define a communication failure. If such a failure occurs, the client's contribution is neglected by the central server. However, to account for the time and resources spent for the (partial) communication, we consider a delay time equal to the estimated upload time assuming the worst-case scenario of losing the connection when the data upload is almost finished. The estimated upload time can be calculated, given our constant throughput model described in Eq. (1) and the a-priori known per-client dataset size.

Same settings overall apply to the FL case. Here, the central server first shares the training model to the clients. Then, each client trains the model using only its available (local) data and finally uploads the updated model back to the server, again in a parallel manner. The server is the one to perform aggregation of all the collected models. In FL, a communication failure may also occur during the period the server shares the training model (downlink). In EL, each participating client uploads its data to its serving edge node; the latter has already received the training model from the central server and then the training process occurs per edge node. Subsequently, updated models are sent back to the central server, similar to the FL case.

D. UE and Servers' Computational Capacity

User equipment: The computational capacity of a mobile device to perform a ML task, measured in (processed) training samples/sec depends on the dataset content *e.g.*, images pose different requirements than natural language, the user equipment's (UE) capabilities and the training model's complexity. A good approximation for popular large-scale classification tasks can however be deducted from [16] since different models have been tested in various configurations. For our case, we use a reference (average) value of 125 training samples/sec, as the most appropriate for our training dataset and model.

Edge nodes: Edge node characteristics have not been finalized yet, given that they are not fully deployed in practical systems. We have therefore considered as our main reference the latest commercial solutions specified by Amazon's AWS Wavelength services [17]. It offers cloud services specialized for ML (Amazon EC2 P3 instances) and is equipped with an NVIDIA Tesla V100 Graphics Processing Unit (GPU). GPU computational capacity values for ML can be found in [18], where we select an average value of 6000 training samples/sec.

Cloud server: The computational tasks for the cloud server include training (in the CL case) and model parameter aggregation (FL, EL cases). For the former, similar to the edge node modelling, we are based on [18]. We select an average value of 40,000 training samples/sec, assuming a DC is equipped with a Tensor Processing Unit (TPU), as opposed to GPU for the edge server. For the latter, no reference values can be found in the

literature, thus we rely on an empirical approach; we measure the average capacity for training and aggregation tasks in our personal computer (PC) setup (*i.e.*, 6250 training samples/sec and 1.56 model aggregations/sec respectively) and compare against the training capacity reference value of 40,000 training samples/sec that was selected, according to [18]. Assuming a linear relation, the average cloud aggregation capacity is calculated as 10 model aggregations/sec.

E. Energy Consumption

User equipment: In this study, the energy expenditure occurred as a result of a UE's standard operation *e.g.*, displaying is not considered. Instead, we focus on expenditure due to data transmission (TX) or reception (RX) and training (ML) related tasks. Thus, the energy consumption E_i *i.e.*, battery discharge of the i^{th} device is computed as: E_i=E_i^{TX}+E_i^{RX}+E_i^{ML}, where the superscript TX, RX and ML marks one of the aforementioned functions. In a given time period t, this can be calculated as $E_{\mathrm{i}} = P_{\mathrm{i}} * t$, where P_i stands for the respective (average) power consumption. For LTE, average power consumption values related to transmission are reported in [19], where P_i^{TX}=2.2 Watts and P_i^{RX}=1.5 Watts. Likewise, for ML, based on [16], we assume P_i^{ML}=2 Watts, as the most appropriate to our model and SVHN training tasks (see paragraph II-F). The sum of all device energies (E_i) comprises the total client energy expenditure.

Edge nodes: For edge devices, energy is consumed only in the EL case, due to ML training. Given a specific number of dataset samples, its training time t_{train} is calculated using the computational capacity values from paragraph II-D. The respective energy expenditure is given by $E_{edge} = t_{train} * P_{edge}$, where P_{edge} stands for the average power consumption for training. Using [20] for CPU-GPU power benchmarking, we obtain an average value of 50 Watts for P_{edge}.

Network devices: Energy consumption in the core network is calculated by summing the energy consumption of core network devices (routers, gateways, switches, interfaces to Access/Cloud), which are given by [13] (as averages in Joules/bit) in relation to the data exchanged in the UL/DL streams.

Cloud server: Similar to the edge nodes, the computational capacity values of cloud tasks (training and aggregation) are discussed in paragraph II-D. Energy expenditure per task can thus be calculated[4], given an average power expenditure. For the training task (CL), being an intensive processing task, we assume an average power of 384 Watts, based on Google's TPU benchmarking [21]. For the (less-intensive) aggregation task (FL, EL) we assume an average value of 15 Watts, based on measurements for matrix multiplication tasks [22], which are similar in complexity to weighted averaging (aggregation).

F. Machine Learning

We have selected an image-classification problem, as a representative ML task and in specific digit recognition from given images taken from the Street View House Numbers (SVHN) dataset. SVHN, which is widely used in bibliography *e.g.*, [23], is based on a set of real-world images, with digits taken from natural scenes (house numbers in Google Street View). It contains a training dataset of 531K 32x32 colour training images (of 1.3 GB size) split in 10 classes (for digits 0-9) and a test dataset of 26K test images. The original SVHN training dataset is replicated 10 times, adding a random Gaussian noise factor (blurring) $\tilde{N}(0, 0.02)$ to the images vectors, essentially resulting in a total synthetic 13GB dataset.

A neural network was developed to address the above-mentioned task, comprised of an input layer of 3072 neurons, which correspond to the total pixels of the input SVHN images (32x32x3), an output layer of 10 neurons, equal to the total output classes of SVHN and a hidden layer of 512 neurons. Rectified Linear Unit (ReLU) activation is applied on the hidden linear layer (ReLU functions as a filter, allowing only positive values to pass through), while on the output layer LogSoftmax activation [24] is selected, being more effective for N-element classification tasks. Regarding hyper-parameter settings, a batch size of 64 samples was chosen, along with a learning rate of 0.1, based on the default settings for similar image classification tasks in PySyft library [25]. The total model size reaches 6.1 MB[5]. Federated Averaging (FedAvg [26]) algorithm is used in all distributed learning cases in our PySyft implementation [25].

III. Experimental Evaluation

For the simulation, the PySyft distributed learning environment [25] was used in a desktop machine with the following characteristics: Intel Core i7-10700 CPU @ 2.9 GHz, 16 GB RAM. The synthetic training dataset in divided into Z equal-sized partitions. We have chosen two key scenarios, based on our previous work [10]; 1) Z = 1000, accounting for a per client data to model ratio r=2.1, where CL and FL are under an equal data exchange and 2) Z = 250 (r=8.5), as a more realistic case for ML applications, where each client holds considerably larger amounts of data than the ML model. Per round partipants (K_N) is fixed to 50. For each scenario, we consider 5 2-hr samples from the mobility dataset, where the predefined time limit is also set to 2hr. For each sample we test the various ML schemes (CL, FL, AEL, REL), essentially resulting in a total of 40 experiments.

To evaluate the performance of each scheme the following metrics are considered: 1) Testing Accuracy, representing the ratio (%) of successful to total classifications. 2) Traffic Volume, calculated as the sum of total data exchanged between the central server and the clients (FL, CL) or between the central server, the edges and the clients (AEL, REL). For the sake of clear representation, traffic volume is normalized to the total training dataset size (*i.e.*, 13 GB). 3) Energy consumption, measured as the total energy expenditure due to processing (training, model aggregation) and data transmission (see section II-E) for the various network's components.

[4]An end-to-end ML-related service should also account for time and resources spent due to inference *i.e.*, applying the trained model on active (new) data to extract an output. Our work though, focuses on the computational-heavy and network-challenging ML training process (and not the inference task that might as well be realised as a stand-alone task, much later in time)

[5]Using larger model sizes (up to hundreds of MBs) such as multi-layer convolutional neural networks is feasible but would face practical limitations *e.g.*, disk capacity in a considerable number of modern UEs; it would affect users' willingness to participate in the training and limit our system's realism.

Accuracy and amount of exchanged data: In terms of the resulted accuracy (*i.e.*, at the end of training), for the first scenario (r=2.1), REL exhibits similar performance to CL, while AEL equals that of FL (Fig. 2). Also, it is shown that both CL and REL outperform AEL and FL, respectively, by an average value of 8%. Intuitively, the above observations match the expectation of AEL achieving the same accuracy as FL (*i.e.*, in the former case, training just takes place closer to the cloud compared to the later) while REL exhibiting a behavior close to CL (*i.e.*, in both cases large amounts of data are gathered centrally and used for training). When client data scales (r=8.5) similar behavior is observed (Fig. 3) regarding the resulted accuracy of the considered approaches, with the (AEL,FL- REL,CL) pairwise gap reducing (up) to 5%. Regarding the bandwidth expenditure, under r=2.1 FL and CL (shown already in [10] to exchange similar data for this ratio) together with REL, converge to 100% (Fig. 4). REL slightly exceeds 100% since apart from all the available data uploaded to the edge servers, the (lightweight) model parameters and aggregated result are exchanges with the cloud server. AEL, on the other hand consumes twice the amount of data, stemming from the fact that numerous edge nodes send/receive model parameters to/from the cloud server. When large amounts of data are available (*i.e.*, r reaches 8.5), REL-CL exhibit similar performance as before, while both FL and AEL mark a 80% reduction (Fig. 5), since less clients (therefore less models) are participating totally. Still, the total expenditure for AEL is 20% greater than CL-REL; given a fixed total dataset size, larger values of r will further decrease the total client number, thus this difference is expected to be minimized.

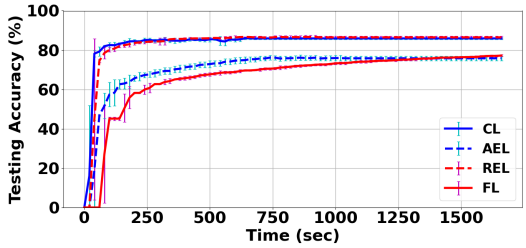

Fig. 2. Testing accuracy for different approaches across time (r = 2.1)

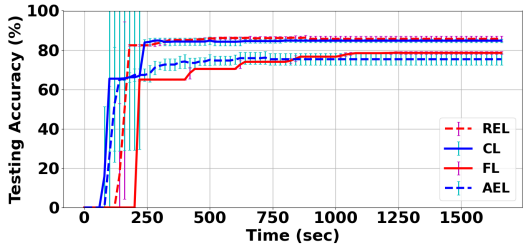

Fig. 3. Testing accuracy for different approaches across time (r = 8.5)

Consequently it is shown, that for REL the trade-off between exchanged data and achieved accuracy follows the behavior of CL. REL thus, stems as a viable solution to ML workloads offloading. On the contrary, AEL fails to achieve CL's accuracy levels, whilst consuming more data; therefore is not recommended for ML-related applications, even if a choice of placing edge nodes close to the base station is widely adopted (in the future). In the latter case, ML tasks and relevant infrastructure *i.e.*, GPUs appear more efficient when moved to a regional level. Going beyond the edge, FL appears as a viable alternative to CL, especially relying on its reduced communication costs for large r values.

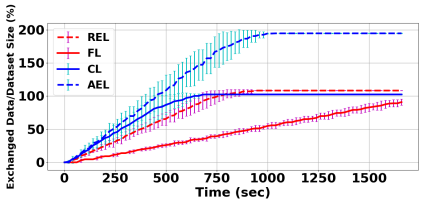

Fig. 4. Exchanged data ratio for different approaches across time (r = 2.1)

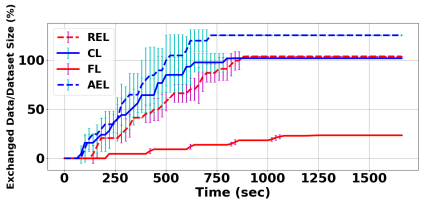

Fig. 5. Exchanged data ratio for different approaches across time (r = 8.5)

Behavior in the course of time: From the viewpoint of convergence speed, CL and REL are able to reach their peak performance during the first training rounds (at 200 secs). At this stage, they outperform AEL by 15% and FL by 21% for r=2.1 (Fig. 2) and by 14% and 18% for r=8.5 (Fig. 3), respectively. The slow accuracy improvement of FL compared to other schemes is mainly dictated by the low processing capacity of the mobile devices, as opposed to a GPU in the edges (AEL-REL) or TPU in the cloud (CL). The faster convergence of CL-REL comes at a cost in bandwidth expenditure. In specific, for r=2.1 REL (at 200 secs) requires 2.5 times more data (for CL that is almost 4) as compared to FL. For r=8.5 these values are 2 and 3, respectively. Fast convergence however, may imply (financial) gains in view of the per-hour cost for edge resources usage [17]. Finally, AEL utilizes more data than any other scheme.

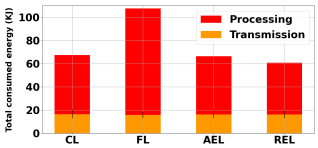

Fig. 6. Total energy consumed (r=2.1)

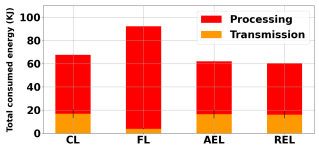

Fig. 7. Total energy consumed (r=8.5)

Energy expenditure: The overall energy consumption for each scheme is depicted in Fig. 6-7. FL achieves the worst performance, consuming 60% (r=2.1) and 32% (r=8.5) more energy overall, as compared to CL. As shown in Fig. 6-7, energy expenditure due to processing (UE training) governs this behavior for FL. A reduced transmission of data for r=8.5,

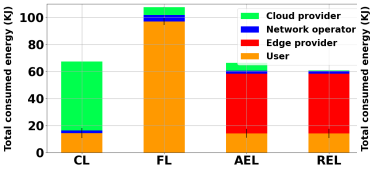

Fig. 8. Stakeholders energy cost (r=2.1)

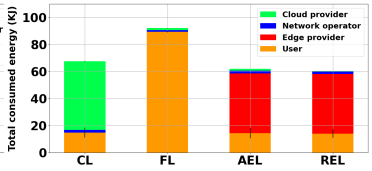

Fig. 9. Stakeholders energy cost (r=8.5)

comes hand to hand with a respective energy reduction. For r=2.1, AEL's energy performance is similar to that of CL, while REL is 8% more efficient. For r=8.5 both edge-based schemes exhibit this 8% efficiency. Regardless the ratio and the scheme, the energy costs of processing surpass that of transmission by at least 2 times.

Considering energy consumption from a stakeholder standpoint (Fig. 8-9), we deduct that the user enjoys a reduction factor of 80% for both ratios, if an edge (REL, AEL) or centralised (CL) learning approach is used (instead of FL). From the cloud's perspective, any scheme would reduce energy costs by at least 85% as compared to CL for r=2.1, while for r=8.5 non-CL schemes minimize cloud's energy consumption. Assuming the network operator is only in charge of the network devices, its energy consumption accounts for 5% of the total energy consumption regardless the ML scheme. Finally, in case the cloud provider offers edge computing services, a welcome 13% reduction appears (for r=8.5) if CL learning tasks are offloaded either to AEL or REL.

IV. Conclusions

In this work, we have introduced two variants of edge-based ML (EL), accounting for the case where an edge-node is attached to a basestation (AEL) or is associated with a region of multiple cells (REL) respectively. Together with FL, EL schemes were implemented and drawing on our cloud-to-UE system model for the underlying network resources, we compared the distributed schemes against the traditional CL. Our simulations, capturing ML accuracy/convergence speed subject to bandwidth and energy constraints, suggest that unlike AEL, REL exhibits similar performance to CL, thus, it emerges as a prominent edge learning approach. Beyond the edge, FL (given adequate per client data) enjoys reduced communication costs, despite being the least energy-efficient scheme system-wise, mainly due to processing needs.

Our results point to multiple future work threads; an investigation of non-uniform distribution of cells across the edge nodes (clustering) and its impact on the edge learning performance is one key direction. Another is the exhaustive exploration of performance-resources trade offs as the edge learning functionality moves hop-by-hop closer to the cloud. Finally, the (currently) linear model for the UE/edge/cloud energy consumption and processing capacity can be extended.

Acknowledgements

This paper is part of the 5G-LOGINNOV project, co-funded by the EU under the H2020 Research and Innovation Programme (grant agreement No 957400).

References

[1] J. Pfeiffer, T. Pfeiffer, M. Meißner, and E. Weiß, "Eye-tracking-based classification of information search behavior using machine learning: evidence from experiments in physical shops and virtual reality shopping environments," *Information Systems Research*, 2020.

[2] X. Ran *et al.*, "Deepdecision: A mobile deep learning framework for edge video analytics," in *IEEE Conference on Computer Communications (INFOCOM)*, 2018, pp. 1421–1429.

[3] X. Lian *et al.*, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," in *Advances in Neural Information Processing Systems*, 2017, pp. 5330–5340.

[4] S. Wang *et al.*, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *IEEE Conference on Computer Communications (INFOCOM)*, 2018, pp. 63–71.

[5] Z. Tao and Q. Li, "eSGD: Communication efficient distributed deep learning on the edge," in *{USENIX} Workshop on Hot Topics in Edge Computing (HotEdge 18)*, 2018.

[6] Y. Huang *et al.*, "When deep learning meets edge computing," in *2017 IEEE 25th Intern'l Conference on Network Protocols (ICNP)*, pp. 1–2.

[7] Y. M. Saputra *et al.*, "Energy demand prediction with federated learning for electric vehicle networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[8] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Federated learning for ultra-reliable low-latency v2v communications," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7.

[9] M. M. Wadu, S. Samarakoon, and M. Bennis, "Federated learning under channel uncertainty: Joint client scheduling and resource allocation," *arXiv preprint arXiv:2002.00802*, 2020.

[10] G. Drainakis *et al.*, "Federated vs. centralized machine learning under privacy-elastic users: A comparative analysis," in *IEEE 19th Intern'l Symposium on Network Computing and Applications*, 2020, pp. 1–8.

[11] M. R. Akdeniz *et al.*, "Millimeter wave channel modeling and cellular capacity evaluation," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1164–1179, 2014.

[12] M. Rizwan and S. A. Abbas, "Median path loss, fading and coverage comparison at 3.5GHz and 700Mhz for mobile WiMax," in *IEEE International Multitopic Conference*, 2008, pp. 266–271.

[13] A. Vishwanath *et al.*, "Energy consumption comparison of interactive cloud-based and local applications," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 4, pp. 616–626, 2015.

[14] Y.-C. Chiu *et al.*, "Are we one hop away from a better internet?" in *Procs. of the 2015 Internet Measurement Conference*, pp. 523–529.

[15] S. Wang *et al.*, "Edge server placement in mobile edge computing," *Journal of Parallel and Distributed Computing*, vol. 127, 2019.

[16] J. Liu, J. Liu, W. Du, and D. Li, "Performance analysis and characterization of training deep learning models on mobile device," in *25th IEEE Intern'l Conf. on Parallel and Distributed Systems*, 2019, pp. 506–515.

[17] Amazon AWS Wavelength - EC2 instance types. [Online]. Available: https://aws.amazon.com/ec2/instance-types/

[18] Y. Kochura *et al.*, "Batch size influence on performance of graphic and tensor processing units during training and inference phases," in *International Conference on Computer Science, Engineering and Education Applications*. Springer, 2019, pp. 658–668.

[19] A. Nika *et al.*, "Energy and performance of smartphone radio bundling in outdoor environments," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 809–819.

[20] D. Li *et al.*, "Evaluating the energy efficiency of deep convolutional neural networks on CPUs and GPUs," in *IEEE Intern'l Conference on Big Data and Cloud Computing*, 2016, pp. 477–484.

[21] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, 2017, pp. 1–12.

[22] T. Jakobs, M. Hofmann, and G. Rünger, "Reducing the power consumption of matrix multiplications by vectorization," in *IEEE Int'l Conference on Computational Science and Engineering (CSE)*, 2016, pp. 213–220.

[23] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *Advances in neural information processing systems*, 2017, pp. 1195–1204.

[24] PyTorch Neural Network API. [Online]. Available: https://pytorch.org/docs/stable/nn.html

[25] T. Ryffel *et al.*, "A generic framework for privacy preserving deep learning," *arXiv preprint arXiv:1811.04017*, 2018.

[26] B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, 2017, pp. 1273–1282.