On the Resource Consumption of Distributed ML

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 convergence of Machine Learning (ML) with the edge computing paradigm has paved the way for distributing processing-heavy ML tasks to the network's extremes. As the edge deployment details still remain an open issue, distributed ML schemes tend to be network-agnostic; thus, their effect on the underlying network's resource consumption is largely ignored.**

In our work, assuming a network tree structure of varying size and edge computing characteristics, we introduce an analytical system model based on credible real-world measurements to capture the end-to-end consumption of ML schemes. In this context, we employ an edge-based (EL) and a federated (FL) ML scheme and in-depth compare their bandwidth needs and energy footprint against a cloud-based (CL) baseline approach. Our numerical evaluation suggests that EL exhibits a minimum of 25% bandwidth-efficiency compared to CL and FL, if employed by a few nodes higher in the edge network, while halving the network's energy costs.

I. Introduction

The recent paradigm of shifting computation to the edge of the network and beyond, even reaching hand-held devices, promises to minimize latency and exploit local context. This allows for a new breed of applications to emerge [1] *e.g.*, virtual reality, ultra-high quality streaming, driving automation *etc.* Machine Learning (ML) applications appear to play a key role in such a shift [2], given their heavy computational needs and high data availability requirements. In fact, offloading ML tasks to the edge (Edge Learning - EL) or to the clients themselves (Federated Learning - FL) is increasingly gaining ground [3], as opposed to the traditional centralized approach (Centralized Learning - CL), where large amounts of data is centrally gathered for training *e.g.*, in a Data Centre (DC).

While the emerging distributed ML algorithms tackle the fundamental challenge of training heterogeneous data stemming from multiple locations [2], the energy footprint and bandwidth needs of these distributed schemes in a realistic network environment have been largely neglected; and so is the identification of their induced costs for the various network stakeholders (*i.e.*, cloud/edge provider, mobile network operator, end-user). Such an exploration relates to the underlying edge network topology that remains open to question. Until recently there have been numerous approaches in regards to the network location of edge computing nodes [1], from a base station (BS) level near the client to an intermediate nano-DC. The required convergence is expected to be shaped by multiple parameters such as the application and network requirements, geographical coverage, costs *etc.* Insights from an ML-standpoint are missing even if valuable, as ML tasks are fast becoming the dominant network workloads [4].

Background: A handful of works seek to compare EL with cloud-based schemes, as in [5], where a simulation-based study identifies the relation of the network overhead to the achieved ML accuracy and in [6], where experiments are conducted on an edge-testbed to measure the accuracy as well as the involved traffic overhead and computational costs. An overall modeling of the ML schemes' resource consumption is still missing. Some recent works following the modeling thread, are usually restricted to a specific stakeholder, thus lack a system-wise view. For CL, the work in [7] models the energy expenditure of the ML task itself in the cloud by analyzing the energy cost of each ML function (backward and forward propagation). Regarding EL, energy consumption and computational models are mostly considering the physical layer *i.e.*, user equipment (UE) [8]. Likewise, FL studies have focused on the participating devices; in [9] the UE's energy, computation and communication cost is explored and in [10] bare-metal measurements are carried-out on smartphones.

Our contribution: To compare the consumption of various ML schemes, we model a network in an end-to-end fashion; we consider the cloud elements, the core network, the edge nodes together with the user equipment (UEs). We factor-in each component's characteristics (*i.e.*, throughput, computational capacity and energy consumption) relying on credible measurements taken from real systems. Assuming a tree-like network (physical) topology, which is widely-adopted in literature [11], we draw on the different cases for the edge nodes network location, representing various degrees of data aggregation. Having secured (through our prior research) that those aggregated data suffices to achieve certain accuracy levels, if utilized to train ML models, we leverage our measurement-based system model to study the result of the application of both distributed (EL, FL) and centralized (CL) ML schemes over the network. Analytical calculations are derived to capture the bandwidth and consumed energy of each scheme along with the way the involved energy costs are broken down into the system's stakeholders; thus, revealing each scheme's benefits and emerging trade-offs. This study extends our previous work [12] on the CL-FL comparison by introducing EL as an alternative distributed ML scheme; we extend the system model to allow for edge node utilization and shed light on the resources consumption per-stakeholder. Our theoretical analysis and numerical results recommend for ML workloads an edge node deployment at a regional level, closer to the cloud. A regional EL scheme is shown to be at least 25% more bandwidth-efficient compared to CL and FL, while reducing the energy costs of the mobile network operator up to 50% and 75% compared to CL and FL, respectively.

The remainder of the paper is structured as follows. In Section II the model is introduced and analytical expressions are derived. Section III presents numerical results and parameter analysis while Section IV concludes discussing future work.

II. System Model

We consider a large-scale cellular network environment with several mobile clients, each generating an amount of training data. The clients are connected with a cloud server, located inside a data centre (DC) and with edge nodes, via an intermediate core network (Fig. 1). Utilizing client data[1], we aim to perform a ML task *i.e.*, to train a model[2], originally generated in the cloud server, in line with the following ML schemes:

CL: Training is performed by the cloud server. In each training round the server selects a group of clients, which are asked to directly upload their raw data to the server. The server utilizes all acquired round data, thereafter, to perform the training task. This process is repeated for several rounds, each time with a selection of another group of clients, until all data is depleted.

FL: Training is performed by the clients, while the cloud server performs the model aggregation. In each round, the server dispatches the current training model to the selected clients (learners), which in turn train it using their own data and computing resources. As opposed to CL, clients are no longer required to upload their actual data, guaranteeing data privacy. Instead, once the (local) training is completed, they upload the updated model parameters (which are typically far-more lightweight compared to the actual data) to the cloud 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: Training is performed in the edge nodes, aggregation in the cloud server. EL functions as an intermediate scheme between a fully-centralized (CL) and a fully-distributed (FL) solution, where the edge nodes act as learners. We consider a setup where edge nodes can be positioned in a varying depth of the network topology. The clients that participate at each round upload their raw data to the edge node that serves their cell. An edge node serving such a cell (or multiple) is called an active edge node. Active nodes also acquire the current training model from the cloud server. Afterwards, they perform the training using the collected data and upon completion, push the model parameters back to the cloud server. Similarly to the FL scheme, the cloud server is only responsible for model aggregation and (re-)distribution of the updated model.

A. Network model

We assume a mobile Long-Term Evolution (LTE) cellular network where a basestation (BS) lies in the centre of each cell. We refer to the wireless part of the network (BS-client link) as the access network, while the wired part (from BS up to the cloud) comprises the core network (Fig. 1). The core network is divided to the mobile network, where the edge nodes reside (metro and edge in Fig. 1), the backbone network and cloud infrastructure.

[1]Control and management plane's messages are of negligible size.
[2]We assume a recently generated model without pre-training.

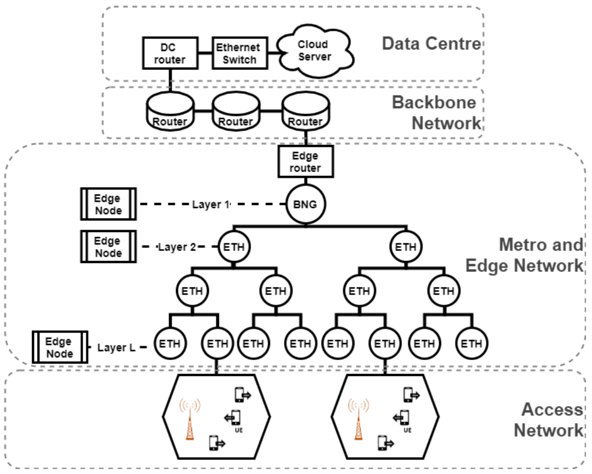

Fig. 1. End-to-end network architecture

Access network throughput: The client throughput s_{client} in both the uplink (UL) and the downlink (DL) (in MBytes/sec) is modelled as a Gaussian random variable (Ñ). Its mean value (mean client throughput) equals that of the average cell throughput[3] c, divided by the number of online clients q. 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 σ is also introduced, equal to 20% of the mean value [14], to account for throughput variations *e.g.*, due to path loss. Also, we assume that a minimum throughput c_{min} exists for any client, to allow for BS-client communication, both in DL and in UL, equal to the 5% cell edge rate[4], representing the worst wireless conditions. Thus, s_{client} for UL and DL is:

$$s^{UL/DL}_{client} = max\{\tilde{N}(\frac{c^{UL/DL}}{q}, \sigma), c^{UL/DL}_{min}\} \tag{1}$$

Core network throughput: The core network includes the following elements [15]: 1) An interface to the access network (BS); 2) The metro and edge network's elements, *i.e.*, several ethernet switches (ETH), 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 for the UL/DL ($s^{UL/DL}_{core}$) is based on Cisco routers/switches performance benchmarking [15] and measurements on a 3-sector 2×2 Multiple-Input-Multiple-Output remote radio 4G/LTE [15]. A total number of 3 backbone network's routers is considered, in line with [16], which shows that a hopcount of maximum 3 in the core network suffices to reach the DC (from the mobile network) for the majority of popular services.

Edge node topology: Unlike the well-established infrastructure of the rest of the core network [15], the edge network's topology is yet to be converged; the edge nodes' network location is an open issue [1]. For our model, we consider a tree topology [11], with BNG being the tree's root and ethernet switches constituting the tree's leaves (Fig. 1). As the tree

[3]That is, 5.9 (UL)/7.73 (DL) MBytes/sec for 2.5 GHz LTE according to [13]
[4]That is, 0.24 (UL)/0.22 (DL) MBytes/sec according to [13]

expands from the BNG down to the BS (DL direction), more and more leaves are generated. This behavior is governed by the tree height H and width W. A full W-ary tree is considered, where W is the fan-out degree of each tree node *i.e.*, each node has W leaves. Motivated by the applications' demands for low latency and high throughput, we adopt a "pervasive" edge-deployment paradigm [1]; the edge node placement may occur in any tree layer, representing various degrees of client aggregation. For example, layer 1 denotes edge placement in BNG, while layer L denotes edge placement at the BS. In any case, the lower the edge node is placed in the network (closer to the BS), the lower is the degree of aggregation; each edge node will serve a smaller number of clients (equivalently, cover a smaller service area). As a result, a larger number of edge nodes is required in total. When CL or FL is used, no edge nodes are utilized and thus, affecting relevant computations; data is directly exchanged between the server and the clients via the intermediate network nodes. In EL, however, the edge nodes location dictates its performance. Essentially, each L value is associated to a distinct EL scheme's variant, spanning from (access) variants approaching the BS to regional ones, residing towards the cloud. For each EL variant we study, all edge nodes are considered to be located in the same tree layer (L), therefore they present an equal degree of aggregation. So, the total number of active edge nodes per round is $g = W^{L-1}, L \in [1, H]$, where L is the tree layer where the nodes are located. We assume that any edge device considered is attached to a tree node, which is typically an ethernet switch (ETH).

B. Data assignment

Real-world client devices will generate data via their sensors, cameras *etc.* To emulate such an environment, we divide the original training dataset into partitions and assign them to the various clients. Each partition represents the data that is generated and stored in the client's device. In specific, we assume a dataset of d_{sample} training samples of a total size of d Bytes. We also define the dataset size to sample ratio $r_{sample} = d/d_{sample}$. The dataset, including data and labels, is divided into z partitions. A fixed number of per round participating clients k is chosen (k<z). The total number of rounds is then equal to $u = \lceil z/k \rceil$. In each round, the server identifies all online clients q, from which, a total of k clients are randomly chosen to participate. Each selected client is assigned a dataset partition. Marking the per client dataset size as d_i, the total dataset can be written as: $d = \sum_{i=1}^{z} d_i$. Assuming a fixed model size m, the per client data to model size ratio is defined as $r_{data_i} = d_i/m, \ i \in [1, z]$. If the initial dataset is equally distributed among the clients (*i.e.*, e.e.d. setting), the client dataset size (d_i) and dataset to model size ratio (r_{data_i}) are simplified to: d_i=d/z, r_{data_i}=$d/(m{\cdot}z), \forall i \in [1, z]$, respectively.

Upon partition assignment, the training procedure takes place. In the CL case, the selected clients upload their local datasets to the cloud server in a parallel manner. The time (t_i) required for each client's dataset (d_i) upload is equal to the time of the access upload plus the time of the core upload, therefore can be written as (Sec. II-A): $t_i = \frac{d_i}{s^{UL}_{client}} + \sum_j \frac{d_i}{s^{UL}_{core_j}}$, for j core components. The total time to upload all datasets equals to that of the "slowest" client, since a parallel transmission is assumed. The cloud 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 all datasets are used. Same settings overall apply to the FL case. Here, the cloud server firstly 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.

In EL, each participating client uploads its data to its serving edge node; the latter has already received the training model from the cloud server and then the training process occurs per edge node. Subsequently, updated models are uploaded back to the cloud server, similar to the FL case. In terms of data transmission, in each round, every edge node needs to exchange the model with the cloud server and acquire all datasets from its serving clients. For our modeling, we assume that each round's selected clients (k) are uniformly distributed across the network's cells. This ensures that in every round, all randomly selected clients across the network's cells will be evenly divided in the available (serving) edge nodes, therefore all available edge nodes will be active learners. In an actual network however, the edge node locations and their serving cells respectively are governed by various factors *e.g.*, spatial characteristics, cellular architecture, average user demand *etc.* Moreover, the network's cells can be further differentiated *e.g.*, picocells, nanocells *etc.* Although less realistic, our baseline assumption is insightful as it reduces the involved problem parameters and enables us to focus on the effect of network topology into EL's performance.

C. Traffic Overhead

We define the traffic overhead (b) of a ML scheme as the total amount of data exchanged for the duration of the training process, multiplied to the total number of network nodes (hops) the data transverses from source to destination. Based on the network topology (Sec. II-A) and the data assignment process (Sec. II-B) of our model, b for each ML scheme is calculated as follows:

$$b=\begin{cases} d \cdot h_{\text{total}}, & \text{CL} \\ 2 \cdot m \cdot z \cdot h_{\text{total}}, & \text{FL} \\ 2 \cdot m \cdot g \cdot \frac{z}{k} \cdot h_{\text{top}} + d \cdot h_{\text{bottom}}, & \text{EL} \end{cases} \tag{2}$$

where h_{total} are the total hops from the cloud to the client, h_{top} the hops from the cloud to an edge node and h_{bottom} the hops from an edge node to the client. Apparently, $h_{total} = h_{top} + h_{bottom}$. In CL, the whole dataset d traverses the network from the clients to the cloud (h_{total}), where the training occurs. In FL, instead, a total of z clients will both receive (DL) and push (UL) their models (of size m) to the cloud. In EL, there are two distinct communication streams; the clients push data to the edge nodes (for h_{bottom} hops) and the active edge nodes per round g exchange models with the cloud (for h_{top} hops).

D. UE and Servers' Computational Capacity

User equipment: The computational capacity of a mobile device to perform a ML task v^{ML}_{client}, measured in (processed) training samples/sec depends on the dataset content, the UE capabilities and the complexity of the ML model. An approximation for popular large-scale classification tasks can be found in [17], where different models have been tested in various configurations. We use a reference (average) value of v^{ML}_{client}=125 training samples/sec, as the most appropriate for our training dataset and model (Sec. III).

Edge nodes: Edge node characteristics have not been revealed yet, given that they are not fully deployed in practical systems. We have therefore considered the latest commercial solutions specified by Amazon's Wavelength services [18] as our main reference. It offers cloud services specialized for ML and is equipped with an NVIDIA Tesla V100 Graphics Processing Unit (GPU). Thus, an edge node's computational capacity v^{ML}_{edge} equals that of a GPU's computational capacity, whose values for ML tasks can be found in [19], from where we select an average value of v^{ML}_{edge}=6000 training samples/sec.

Cloud server: Computational tasks for the cloud server include training (in CL) and model parameter aggregation (in FL, EL). Regarding training, we assume that a DC is equipped with a Tensor Processing Unit (TPU), as opposed the edge server's GPU. Based on [19], we select an average value for the computational capacity for training v^{ML}_{cloud}=40,000 training samples/sec. In regards to aggregation, 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 [19]. Assuming a linear relation, the average cloud aggregation capacity v^{AG}_{cloud} is calculated as 10 model aggregations/sec. Our modeling does not look into scale out schemes, where clusters of servers may be used to increase parallelism in DCs.

E. Energy Consumption

User equipment: Only the consumption that occurred as a result of the UE participation in the ML scheme is considered; that is for expenditure due to data transmission (TX)/reception (RX) or training (ML) related tasks. Any expenditure due to UE's standard operation *e.g.*, displaying, is neglected. The energy consumption e_{client_i} *i.e.*, battery discharge of the i^{th} client's device is computed as: e_{client_i}=$e^{TX}_{client_i}$+$e^{RX}_{client_i}$+$e^{ML}_{client_i}$, where the superscript TX, RX and ML marks one of the aforementioned functions. In a given time period t, this can be calculated as $e_{client_i} = p_{client_i} \cdot t$, where p_{client_i}, $i \in [1, z]$ stands for the respective (average) power consumption. For LTE, average power consumption values related to transmission are reported in [20], where $p^{TX}_{client_i}$=2.2 Watts and $p^{RX}_{client_i}$=1.5 Watts, $\forall i \in [1, z]$. Likewise, for ML, based on [17], we assume $p^{ML}_{client_i}$=2 Watts, $\forall i \in [1, z]$, as the most appropriate to our ML model and training task (see Sec. III). The sum of all device energies e_{client_i} comprises the total client energy expenditure e_{client}.

$$\mathrm{e}_{\mathrm{client}} = \begin{cases} p^{TX}_{client} \sum_{i=1}^{z} \left(\frac{d_i}{s^{UL}_{client_i}}\right), & \text{CL, EL} \\ \frac{p^{ML}_{client} \cdot d}{v^{ML}_{client} \cdot r_{sample}} + m \cdot \sum_{i=1}^{z} \left(\frac{p^{TX}_{client}}{s^{UL}_{client_i}} + \frac{p^{RX}_{client}}{s^{DL}_{client_i}}\right), & \text{FL} \end{cases} \quad (3)$$

In CL and EL, total client energy is dictated by the upload time of each client, which in turn depends on the the client's dataset size d_i and its UL throughput $s^{UL}_{client_i}$. In FL, the total energy is not only affected by the transmission (exchange of models in UL/DL), but also by the time required for the local training.

Edge nodes: For the edge devices, energy is consumed only in the EL case, due to training. Given a specific number of dataset samples, its training time t_{train} is calculated using the computational capacity values from Sec. II-D. The respective energy expenditure is given by $e_{edge} = t_{train} \cdot p^{ML}_{edge}$, where p^{ML}_{edge} stands for the average power consumption for training. Using [21] for CPU-GPU power benchmarking, we obtain an average value of p^{ML}_{edge}=50 Watts.

$$\mathrm{e}_{edge} = \begin{cases} 0, & \text{CL,FL} \\ \frac{p^{ML}_{edge} \cdot d}{v^{ML}_{edge} \cdot r_{sample}}, & \text{EL} \end{cases} \quad (4)$$

Network devices: Energy needs in the mobile network are calculated by summing the energy consumption of the mobile network devices *i.e.*, routers, switches *etc.* which are given by [15] (in Joules/bit) in relation to the data exchanged for the UL/DL streams. We denote as e_{mob_j} the average energy consumption per bit for a network element j. The total traffic for each ML scheme, can be obtained from Eq. 2. The mobile network's energy consumption e_{mob} can then be calculated by summing the energy consumption of all devices:

$$\mathrm{e}_{mob} = \begin{cases} 8 \cdot d \cdot \sum_{j=1}^{h_{total}-h_{back}} (e^{UL}_{mob_j}), & \text{CL} \\ 8 \cdot m \cdot z \cdot \sum_{j=1}^{h_{total}-h_{back}} (e^{UL}_{mob_j} + e^{DL}_{mob_j}), & \text{FL} \\ 8 \cdot m \cdot g \cdot \frac{z}{k} \cdot \sum_{j=1}^{h_{top}-h_{back}} (e^{UL}_{mob_j} + e^{DL}_{mob_j}) + 8 \cdot d \cdot \sum_{j=1}^{h_{bottom}} (e^{UL}_{mob_j}), & \text{EL} \end{cases} \quad (5)$$

where h_{back} stands for the number of hops in the backbone network and cloud infrastructure, while 8 appears due to byte to bit conversion.

Cloud server: The computational capacity values of cloud tasks (training and aggregation) are discussed in Sec. II-D. Energy expenditure per task can thus be calculated[5], given an average power expenditure. For the training task (in CL), being an intensive processing task, we assume an average power p^{ML}_{cloud}=384 Watts, based on Google's TPU benchmarking [22]. For the (less-intensive) aggregation task (in EL, FL), we assume p^{AG}_{cloud}=15 Watts, based on measurements for matrix multiplication tasks [23], which are similar in complexity to weighted averaging (aggregation). The cloud energy consumption e_{cloud} then becomes:

[5]Inference *i.e.*, applying the trained model on active (new) data also involves resources but may come as a stand-alone task, much later than training which is far more demanding in computations and network resources. Likewise, latency is not modelled in our setup, as training time requirements typically dominate over any latency considerations.

$$e_{\text{cloud}}=\begin{cases} p^{ML}_{cloud} \cdot d/(v^{ML}_{cloud} \cdot r_{sample}), & \text{CL} \\ p^{AG}_{cloud} \cdot z/(v^{AG}_{cloud}), & \text{FL} \\ p^{AG}_{cloud} \cdot z \cdot g/(v^{AG}_{cloud} \cdot k), & \text{EL} \end{cases} \quad (6)$$

III. Numerical Results

Scenario Parameters: To compare the behavior of the ML schemes in question (CL, FL, EL), we have selected an indicative ML task, to be performed over a network of varying characteristics. In specific, we assume the Street View House Numbers (SVHN) dataset, widely used in bibliography *e.g.*, [24], is utilized for image-classification. SVHN contains 531K 32x32 colour training images (of d=1.3 GB size) split in 10 classes, while exhibiting a fixed dataset size to sample ratio r_{sample}=2447 Bytes/sample. The original dataset can be replicated without affecting r_{sample} by adding random Gaussian noise (blurring) to the image vectors. The total replicas are denoted as θ, essentially resulting in a total synthetic dataset of size $d\cdot\theta$. The classification task is assumed to be carried-out using a 3072x512x10 neural network, with a total size m of 6.1 MB. The dataset is divided into equal-sized partitions z *i.e.*, e.e.d. setting. The ML task terminates when all data is depleted, since our previous work in [12] has shown that utilizing the complete dataset once (essentially one epoch) with the above-mentioned model, suffices for over 75% accuracy for FL and over 80% for CL.

In terms of the network topology, we assume a tree-hierarchy as in Fig.1, with a tree height H=7, allowing for a broader range of solutions than [15]. To study various edge node placement scenarios, we vary the tree width W, representing the edge nodes density, as well as the tree layer L, representing the edge nodes' proximity to the clients. The tree's root is always set in the BNG, assuming that an edge node cannot be placed in the backbone network. Based on these settings, the number of hops from the clients to the cloud h_{total}=13. Also, the hops from the edge node to the cloud h_{top} are equal to the total hops from the BNG to the cloud (h_{back}=6), plus the total hops from the BNG to the edge node location (Fig.1), thus h_{top}=6+L. The hops from the edge to the clients are h_{bottom}=h_{total}-h_{top}=7-L, with $L \in [1, H = 7]$.

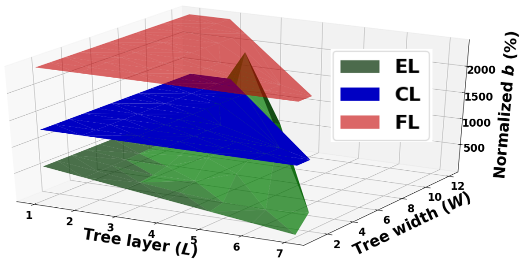

Fig. 2. Normalised traffic b against the edge nodes density (W) and proximity to clients (L)

Effect of tree characteristics and edge node location: Initially, we seek to explore the effect of the network's tree characteristics and edge node location on EL's performance (CL and FL are not affected as there is no interaction with the edge nodes and h_{total}=13). The SVHN dataset is replicated 100 times (θ=100), resulting in a total size d=130GB dataset. We assume z=20K total partitions assigned to the clients, while a fixed number of k=1000 is set for the participating clients per round (*i.e.*, 20 rounds in total). In each cell, we assume q=20 online clients. To more accurately reflect the network usage, b is normalized to the total dataset size d.

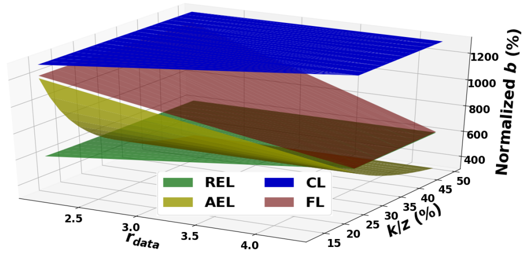

Fig. 3. Normalised traffic b against the per client data to model size ratio r_{data} and the per round clients k as a percentage of the total clients z

The resulted traffic overhead b with respect to both the edge node location (layer L) and tree's width W is depicted in Fig. 2. For CL, a fixed traffic overhead 1300% of the total dataset size is generated, accounting for the fact that the whole dataset traverses from clients to cloud (h_{total}=13). For FL, b is almost doubled, since the ML models traverse both in the UL and in the DL direction. Their behavior is independent of W and governed by the per client data to model size ratio r_{data}, as explored in our earlier research work [12]. Traffic overhead in case of EL, on the other hand, depends both on the edge nodes location (L) and their accompanying density (W). For $L, W < 4$, EL generates less than 50% b compared to CL and less than 25% compared to FL. When $L > 4$ or $W > 4$ *i.e.*, when the edge nodes are placed 4 layers below the BNG (towards the DL direction), EL shows a sudden increase in b, coinciding with FL's b. In that sense, a spatially limited edge network is recommended, closer to the cloud server, for ML-related applications. In regards to the energy footprint for the mobile network e_{mob}, it varies similarly to b, since it is governed by the total traffic that traverses through the network nodes; as shown Table I, its maximum value (for large L,W) is twice the value of its minimum. Therefore if EL is employed towards the cloud, it is twice more energy-efficient for the mobile network operator as opposed to a layer close to the clients; at the same time, an energy cost reduction up to 50% and 75% emerges for the operator compared to the use of CL and FL, respectively. In regards to the energy expenditure for the rest of the network's stakeholders (Table I), we deduce that EL almost halves the total client consumption compared to FL and reduces the cloud's consumption by at least 1600%, when having the consumption offloaded to the edge devices. Interestingly, if the cloud operator offers edge services *i.e.*, we calculate e_{edge} and e_{cloud} together, a 14% reduction of energy emerges when EL is used instead of CL.

Effect of r_{data} and k: We now study how EL's consumption is affected, in comparison to the other ML schemes by 1) the ML task's attributes (captured by the per client data size to model size ratio r_{data}) and 2) by the data assignment procedure's specifics (dictated by the participating clients

TABLE I
ENERGY EXPENDITURE PER NETWORK STAKEHOLDER

	EL	CL	FL
e_{client} (KJ)	970	970	2200
e_{mob} (KJ)	[6-12]	14	25
e_{edge} (KJ)	400	0	0
e_{cloud} (KJ)	(0-30]	500	30

per round k). We select a synthetic SVHN dataset with θ=10, d=13GB, m=6.1MB. Drawing on the results of our previous experiment, we select two indicative variants for EL, a Regional-EL scheme (REL) closer to the BNG with (L,W)=(1,3) and an Access-EL scheme (AEL) closer to the BS/clients with (L,W)=(7,2). Also, we assume e.e.d. settings, therefore r_{data_i} is equal for all clients, $i \in [1, z]$ and denoted as r_{data}, for simplicity. As depicted in Fig. 3, r_{data} shapes the traffic overhead for all ML schemes. In specific, for r_{data}<2 *i.e.*, the client holds data of size less than 2 times the model size, the bandwidth expenditure of CL, FL and AEL reaches the 1300% of the total dataset. REL on the other hand, is proven more efficient, utilizing less than half of b, compared to the rest. An increase of r_{data} *i.e.*, the client holding considerable data in relation to the model, favors both FL and AEL, while CL and REL are not affected. FL and AEL gain a 50% reduction for r_{data}=4 (thus reaching REL's performance). These results are in line with our previous work on r_{data}'s effect on CL-FL performance [12]. We also note that AEL's traffic overhead can be reduced by manipulating the participating clients per round k; as k is increased above the value of 25% of the total clients z, AEL's bandwidth expenditure is reduced exponentially. Such a modification, however, has no effect on the other schemes.

IV. Conclusions

We have in-depth analyzed the energy and bandwidth consumption of distributed ML schemes over a network with varying edge nodes configuration. Our simplified yet realistic, measurement-based system model is introduced to calculate the consumption costs of ML schemes and the way the associated costs are distributed to the involved network stakeholders. Based on our model, we compare the consumption of two distributed ML schemes (EL, FL) against a centralized one (CL), under various cases for the edge nodes network location, followed by a numerical parameter investigation using an actual ML dataset. Our analysis suggests that both the traffic overhead and the energy footprint rapidly increase for EL, when edge nodes are placed closer the clients, in a dense-tree topology. Also, increasing the data to model size ratio or the participating clients per round, exponentially reduces the overhead for EL, as opposed to FL (linearly), CL (constant). Our work points to interesting future directions: the investigation (via simulation) of network topology's effect on the resulted ML accuracy; the impact of non-uniform distribution of clients across the network's cells or the extension of the linear energy consumption model.

Acknowledgements

This paper is part of the 5G-IANA project, co-funded by the EU under the H2020 Research and Innovation Programme (grant agreement No 101016427).

References

[1] D. Artuñedo Guillen *et al.*, "Edge computing for 5G networks- 5G PPP white paper," Mar. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4555780

[2] M. M. Amiri and D. Gündüz, "Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air," *IEEE Transactions on Signal Processing*, vol. 68, pp. 2155–2169, 2020.

[3] S. Hosseinalipour *et al.*, "From federated to fog learning: Distributed machine learning over heterogeneous wireless networks," *IEEE Communications Magazine*, vol. 58, no. 12, pp. 41–47, 2020.

[4] "Ai, machine learning among top cloud workloads," https://www.datanami.com/2017/10/26/ai-machine-learning-among-top-cloud-workloads/, (Accessed on 04/10/2021).

[5] L. Valerio, A. Passarella, and M. Conti, "Accuracy vs. traffic trade-off of learning iot data patterns at the edge with hypothesis transfer learning," in *IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow*, 2016, pp. 1–6.

[6] F. Liang *et al.*, "Toward edge-based deep learning in industrial internet of things," *IEEE Internet of Things Journal*, vol. 7, pp. 4329–4341, 2020.

[7] B. D. Rouhani, A. Mirhoseini, and F. Koushanfar, "Delight: Adding energy dimension to deep neural networks," in *Procs. of the International Symposium on Low Power Electronics and Design*, 2016, pp. 112–117.

[8] Q. Zeng *et al.*, "Energy-efficient radio resource allocation for federated edge learning," in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2020, pp. 1–6.

[9] N. H. Tran *et al.*, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE Conference on Computer Communications (INFOCOM)*, 2019, pp. 1387–1395.

[10] Z. Xu, L. Li, and W. Zou, "Exploring federated learning on battery-powered devices," in *Proceedings of the ACM Turing Celebration Conference-China*, 2019, pp. 1–6.

[11] A. Zavodovski *et al.*, "Exec: Elastic extensible edge cloud," in *Proceedings of the 2Nd International Workshop on Edge Systems, Analytics and Networking*, 2019, pp. 24–29.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.