On the Advantages of P2P ML on Mobile Devices

Robert Basmadjian
Clausthal University of Technology
Clausthal-Zellerfeld, Germany
robert.basmadjian@tu-clausthal.de

Karim Boubouh
UM6P
Benguerir, Morocco
karim.boubouh@um6p.ma

Amine Boussetta
UM6P
Benguerir, Morocco
amine.boussetta@um6p.ma

Rachid Guerraoui
EPFL
Lausanne, Switzerland
rachid.guerraoui@epfl.ch

Alexandre Maurer
UM6P
Benguerir, Morocco
alexandre.maurer@um6p.ma

ABSTRACT

Many fields make use nowadays of machine learning (ML) enhanced applications for cost optimization, scheduling or forecasting, including the energy sector. However, these very ML algorithms consume a significant amount of energy, sometimes going against the whole purpose of their employment. To this day, solutions for an energy-efficient execution of these algorithms have not been addressed adequately. In this paper, we demonstrate the advantage of executing ML algorithms on mobile devices (ARM) over a standard server machine (RISC), from the perspective of energy. To do so, we first propose a novel methodology to quantify the amount of energy consumed by an ML algorithm. Then, we compare the energy consumption of existing algorithms running on mobile devices and server machines. To motivate running ML algorithms on mobile devices, we also propose a new peer-to-peer personalized ML algorithm (P3) that shows better convergence properties than related works, and provably converging to a ball centered at a critical point of a non-convex cost function, under mild assumptions. Most importantly, we show that running the P3 algorithm on mobile devices is extremely energy-efficient, consuming 2700x, 200x and 20x less energy than centralized learning algorithms for 10, 100, and 300 peers respectively. Finally, unlike centralized learning algorithms, the proposed P2P algorithm can generate personalized models, and does not have issues of single-point-of-failure nor data privacy. Thus, we give evidence on the supremacy of our proposed P3 algorithm over the other state-of-the-art centralized ML ones.

CCS CONCEPTS

• **Computing methodologies → Machine learning**.

KEYWORDS

Personalized models, peer-to-peer machine learning, energy-efficiency, ARM and RISC processors.

e-Energy '22, June 28–July 1, 2022, Virtual Event, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9397-3/22/06…$15.00
https://doi.org/10.1145/3538637.3538863

ACM Reference Format:
Robert Basmadjian, Karim Boubouh, Amine Boussetta, Rachid Guerraoui, and Alexandre Maurer. 2022. On the Advantages of P2P ML on Mobile Devices. In *The Thirteenth ACM International Conference on Future Energy Systems (e-Energy '22), June 28–July 1, 2022, Virtual Event, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3538637.3538863

1 INTRODUCTION

Machine learning (ML) has gained a significant popularity among industries, especially in the last decade where the volume and the velocity of data generation reached outstanding levels. As a matter of fact, companies like Google, Microsoft, Facebook or Amazon are constantly using ML models to improve their services and consequently, increase their profits. These ML models are often trained in big data centers, most of the time relying on centralized schemes where a server machine maintains a common model while communicating with other worker machines that perform the computation locally. The *Parameter Server* [1] architecture introduced by Microsoft and the *Federated Learning* paradigm [2] introduced by Google are two notorious examples of centralized learning.

While centralized schemes, coupled with the celebrated *Stochastic Gradient Descent* optimization algorithm [3] (or its variants), show a decent performance in large scale applications, many drawbacks can be named at this point. First, these architectures suffer from a single point of failure (SPOF), which is the server. Indeed, solutions [1, 4] involving replication have been proposed to tackle this problem. However, they come at the expense of time and communication complexity, as well as require additional resources (i.e., storage). Moreover, the server machine constitutes a communication bottleneck since all workers (or a fraction of them in the asynchronous case) are communicating with it in the same time. Most importantly, server machines are famous for their huge energy consumption. In fact, these servers are mainly responsible for the increase in energy consumption of data centers [5–8], as the energy footprint associated with training ML models is now rated in terawatts (e.g., Google 15.4 TWh and Microsoft 10.8 TWh of consumed energy in 2020 [9]).

Decentralized architectures on the other hand solve most of the limitations listed above. Peer-to-peer (P2P) ML is a classical example of decentralized learning where a group of devices (peers) collaboratively learn an ML model, without the need for a central authority/orchestrator (i.e., a server). This very fact comes with many advantages: first, peers can fail without hampering the global execution of the P2P ML training (in other words, no SPOF). Second, peers no longer need to trust a central authority/orchestrator and only rely on a network of similar peers, which is an advantage in terms of data privacy. Finally, since servers are not present in this scheme, energy consumption is decreased and will only depend on the type of the devices (e.g., computers, smartphones, tablets) inside the network.

At this point, the reader may ask the following question: is there a way to measure the energy consumption of ML algorithms running on both aforementioned architectures? There have been some attempts in the literature to estimate the energy consumption of centralized ML algorithms [10–12]. However, it is still lacking in the literature a generic methodology to quantify the amount of energy for those algorithms. Furthermore, the energy consumed by P2P ML algorithms has not yet been adequately addressed.

The major contribution of this paper is two-folded: we propose a generic methodology that can be used to analyze the energy consumed by the centralized and decentralized ML algorithms. Furthermore, using the proposed methodology, we give evidence that executing ML algorithms in the P2P fashion is much more energy-efficient on ARM-based processors of mobile devices than in the traditional way of running ML algorithms on RISC-based processors of servers. To the best of our knowledge, this is the first attempt to evaluate the energy consumption of P2P ML algorithms based on resource-limited devices like mobile phones. Next, we propose P3, a novel P2P personalized ML algorithm that can run on mobile devices, saving thus huge amounts of energy compared to centralized schemes without any performance loss. Essentially, P3 allows collaboration between peers, in order to improve upon locally pre-trained models. It is equipped with a filtering scheme in the collaborative learning phase, so that each peer only aggregates information (gradients, in our case) from similar peers in its neighborhood, by adjusting the range of acceptance, which is based on distance evaluation.

Theoretically, we prove that our algorithm converges to a ball centered at a critical point of a non-convex cost function, as described in Section 2.1. Unlike standard works, we assume that peers produce biased estimate of gradients, following the fact that we do not assume any specific partitioning of the data among peers, which is usually heterogeneous in real applications. We also show the existence of a critical iteration step during the pre-training phase, after which collaboration is no longer beneficial to peers (see Section 3.2). Our experimental results show that the P3 algorithm consumes significantly less energy when running on ARM-based processors, compared to the case where P3 and any other centralized learning (CL) algorithms are executed on RISC-based ones. We show that P3 executed on mobile devices is extremely energy-efficient, consuming 2700x, 200x and 20x less energy than CL algorithms for 10, 100, and 300 peers, respectively. These savings are achieved while maintaining decent convergence properties and solving the traditional problems of CL algorithms, such as SPoF and data privacy.

The major contributions of our work are summarized as follows:

- We propose a new and generic methodology for evaluating the energy consumption of ML algorithms running on servers and Android mobile devices.
- Motivated by the results of our energy analysis, we design a new Personalized P2P (P3) algorithm with a parameterizable personalization degree and prove its convergence properties for non-convex objectives.
- We empirically show the advantage of P2P algorithms (e.g., P3) against centralized schemes in terms of energy consumption without altering convergence properties.
- We conduct extensive experiments on a spectrum of Non-IID data partitioning and network settings to validate the superiority of our proposed algorithm in terms of convergence properties.

The rest of the paper is organized as follows. We first introduce the preliminary concepts and the model setting in Section 2. Section 3 describes our algorithm and its theoretical guarantees. In Section 4, we describe the proposed methodology for our energy analysis. Section 5 presents a selection of experiments demonstrating the superiority of the P3 algorithm against a classical state-of-the-art centralized algorithm in terms of convergence properties and energy requirements. Finally, Section 6 discusses related works, and Section 7 concludes the paper. For space limitations, we defer all the proofs as well as additional experiments to an appendix.

2 PRELIMINARIES

We consider a set of users aiming to train a machine learning model based on their local data while taking into account the models of other anonymous users over the network. For this, users join a decentralized P2P network graph playing the role of a semantic overlay connecting similar users sharing similar tasks and objectives, without revealing their identity or sharing their private data. Each user learns a personalized model that reflects her raw data and personal learning objective, while benefiting from the network of similar users to improve her model with other users' data in a privacy-preserving way.

2.1 Problem Setting

Formally, we consider a set of n users organized in an undirected connected graph $G = (V, E)$. The vertex set V contains a set of n peers, while E denotes the set of weighted edges between these peers. We associate with G a symmetric non-negative similarity matrix $W \in \mathbb{R}^{n \times n}$ to describe the similarity between peers, as suggested in [13, 14]. Given two peers i and j, W_{ij} is the weight of the edge $(i, j) \in E$ representing the similarity between the objectives of i and j. W_{ij} tends to be large (resp. small) if i and j have similar objectives (resp. dissimilar). Furthermore, if i is not linked to j, the similarity is equal to zero (i.e., $W_{ij} = 0$). We also set $W_{ij} = 0$ if $i = j$. In static network graphs, these pairwise similarity weights can be derived from user profiles (e.g., diabetes type, etc.), computed directly from the local datasets, or learned jointly with the model during training [15]. However, in dynamic networks, similarity weights can be learned while training and can be used to improve the set of neighbors of the peer [16]. Finally, we assume that each peer i only communicates with its neighborhood, denoted by $\mathcal{N}_i = \{j : W_{ij} > 0\}$.

Each peer $i \in [\![n]\!]$ holds a local dataset $S_i = \{x_i^j, y_i^j\}_{j=1}^{m_i}$ composed of $m_i = |S_i|$ training samples, independently drawn from the data distribution of the personalized ML problem of the peer. We consider a feature space $\mathcal{X}_i$ and a label space $\mathcal{Y}_i$ defining a personalized supervised learning task, with an unknown probability distribution $P(\mathcal{X}_i, \mathcal{Y}_i)$, and a loss function l. Ideally, a peer would like to minimize the expected risk, defined as follows:

$$R(w) = \int_{\mathcal{X}\times\mathcal{Y}} l(w, x^j, y^j) dP(x^j, y^j)$$

Since $P(x, y)$ is unknown, the empirical risk is usually used as an unbiased estimate of the expected risk. It simply consists in summing the losses over the peer's local data:

$$R_{m_i}(w) = \frac{1}{m_i} \sum_{j=1}^{m_i} l(w, x^j, y^j) \quad (1)$$

2.2 P2P Personalized Learning

Training models in isolation, relying only on local data, can result in either (1) learning weak models, if the peer has little data, or (2) learning models that generalize very poorly, in the case of peers holding data that is not representative of the global data distribution (i.e., non-IID data). Furthermore, peers can have limited processing capabilities (e.g., IoT devices) and be unwilling to share their raw data due to privacy or communication constraints. Our goal is to allow peers to participate in a decentralized collaborative learning scheme. Here, peers can improve upon their locally trained models and benefit from the data of other peers, while maintaining accurate and personalized models with respect to their local data.

While the empirical risk is a good estimate of the true expected risk when data points are independent and identically distributed (i.i.d.), and most importantly, of adequate size, it is generally not the case in real environments. As a matter of fact, peers often have a limited number of non-i.i.d. data points. In our work, we allow peers to collaborate after a pre-training, in order to eventually improve their local models, by completing their information on the true data distribution, using the neighbors' gradients. Our cost function is defined as follows for each peer:

$$F_i(w) = \mu R_{m_i}(w) + (1-\mu)\frac{1}{|\mathcal{N}_i|} \sum_{j\in\mathcal{N}_i} \Big(W_{ij} R_{m_j}(w)\Big) \quad (2)$$

where $\mu \in [0, 1]$ is a personalization parameter to control the effect of the network updates over the locally computed updates. Large values of μ prevent the model from diverging too much from the local data, while small values enable greater contribution from the network. Interestingly, (2) include two extremes case: when $\mu \longrightarrow 1$, peers learn purely local models, and when $\mu \longrightarrow 0$, peers learn a global model.

3 P3 ALGORITHM AND ITS THEORETICAL GUARANTEES

In this section, we present P3, a new collaborative learning algorithm for personalized machine learning. The P3 algorithm uses semi-asynchronous communication to exchange updates within the neighborhood of peers without having a global scope of the network. Note that P3 does not wait for all messages to finish before performing an update. This allows peers to perform updates if a neighbor disconnects, or even in the worst scenario where no messages arrive. Finally, we assume that each peer becomes active when its local clock ticks following a Poisson distribution.

P3 Algorithm

```
Input : Network graph $G$; aggregation rule $\mathbb{A}$; learning rate $\gamma$
Output : $w_i$ optimal personalized model for every peer $i \in G$.
1:  upon event < $p2p.Init$ > do
2:      for each peer $i \in G$ do
3:          $w_i^0 \longleftarrow$ locally trained model $w_i^{loc}$
4:          broadcastUpdate(0);
5:  end event
6:  procedure $broadcastUpdate(t)$    ▷ Trigger collaboration at epoch $t \in [T]$
7:      $\mathcal{D}_i \longleftarrow$ Draw $s$ random samples without replacement from $S_i$;
8:      $v_i^t = \nabla\ell(w_i^t; \mathcal{D}_i)$
9:      trigger < $p2p.Broadcast|neighbors$, $[v_i^t]$ >;
10: upon event < $p2p.Receive|neighbor$, $[v_j^t]$ > do
11:     $V^t \longleftarrow V^t \cup \{v_j^t\}$
12:     if enough gradients received for epoch $t$ then
13:         collaborativeUpdate($V^t$);
14:         broadcastUpdate($t+1$);
15: end event
16: procedure $collaborativeUpdate(V^t)$
17:     for each gradient $v_j^t \in V^t$ do
18:         if $\left\|v_i^t - v_j^t\right\|^2 < \sigma$ then
19:             $accepted \longleftarrow accepted \cup \overline{\{v_j^t\}}$
20:     $v \longleftarrow \mu_i v_i^t + (1-\mu_i)\mathbb{A}(accepted)$    ▷ $\mu_i$ is a personalization parameter
21:     $w_k^{t+1} \longleftarrow w_i^t - \gamma v$
```

3.1 P3 Algorithm

The P3 algorithm consists of two phases: a local learning phase where each peer learns a model w_i^{loc} locally, and a collaboration phase where peers collaborate to enhance their locally trained models.

3.1.1 Local Training Phase. During the local training, peers learn an initial model w_i^{loc} representative of their local data. However, this step can lead to overfitting if the model is overtrained, thus preventing the peer from generalizing its model using the shared knowledge of the network graph, as discussed in Section 3.2.2. Furthermore, peers can suffer from computation and energy constraints even if they had enough data points to learn an accurate model, as local training is computationally expensive and consumes more energy compared to collaborative training (see Table 3).

3.1.2 Collaborative Training Phase. Once a peer i completes its local training phase, it can start collaborating with its neighbors to improve its local model, and compensate for any lack of data or computation power. At each timestamp t, a peer i wakes up and performs the following consecutive steps:

- **Communication step:** Peer i samples a mini-batch of size s from its local dataset S_i and broadcasts its gradient v_i^t to its set of neighbors $\mathcal{N}_i$.

- **Update step:** Upon receiving enough gradients, each peer i evaluates the received gradients and only selects the close ones, by computing the norm squared $\|v_i^t - v_j^t\|^2$ and making sure it is less than a given value σ. Updates that satisfy this condition will be accepted and used to calculate the next model update as follows:

$$v \longleftarrow \mu_i v_i^t + (1-\mu_i)\mathbb{A}(accepted)$$

where the value of μ_i is a personalization parameter, *accepted* is the set of accepted gradient updates, and $\mathbb{A}$ is the aggregation rule used. In our context, we use simple averaging; however, the algorithm can support any aggregation rule.

3.2 Theoretical Guarantees

Similarly to previous works, we prove that the parameter vector sequence progresses towards a region near a critical point of the cost function. We use the following standard assumptions in our analysis.

3.2.1 *Assumptions.*

Assumption 1. *(Smoothness) F is $L-$Smooth:*

$$\forall w', w, \left\|\nabla F(w') - \nabla F(w)\right\| \leq L\left\|w' - w\right\|$$

Assumption 2. *(Biased gradient) The computed gradients $G(w,\xi)$ have a close norm and point in the same direction as the true gradient $\nabla F(w)$:*

$$\langle \nabla F(w)^T, \mathbb{E}\, G(w,\xi)\rangle \geq \rho\, \|\nabla F(w)\|^2$$

Assumption 3. *(Bounded second moment) For all $w \in \mathbb{R}^d$, there exist $(M,N) \in \mathbb{R}^2$ such that:*

$$\mathbb{E}\, \|G(w,\xi)\|^2 \leq M + N\, \|\nabla F(w)\|^2$$

3.2.2 *Convergence Analysis.* As opposed to standard distributed machine learning [17–19], personalized machine learning allows each peer to learn a different model. We therefore conduct a peer-wise analysis, and show that each peer converges near a critical point of its personalized cost function. We first prove an upper bound on the expected error of the aggregated vector in Lemma 1.

Lemma 1. *Let σ be the threshold set by a peer i, and let $AG(w_k)$ be the aggregated gradient at w_k. For each $w_k \in \mathbb{R}^d$, we have:*

$$\mathbb{E}\, \|AG(w_k) - \nabla F(w_k)\|^2 \leq U + V\, \|\nabla F(w_k)\|^2$$

where $U = 2((1-\mu)^2\sigma^2 + M)$ and $V = 2(1+N-\rho)$.

Next, we show in Lemma 2 that this aggregated vector is indeed pointing in the same direction as the true gradient of the personalized cost function. For convenience, we define the subspace $\mathcal{W} \subset \mathbb{R}^d$ as follows:

$$\mathcal{W} := \left\{ w \in \mathbb{R}^d \,\middle|\, (1-V)\, \|\nabla F(w)\|^2 - U \geq 0 \right\}$$

Lemma 2. *Let Assumptions 2 and 3 hold. When $w \in \mathcal{W}$, the expected aggregated vector $E[AG(w)]$ points in the same direction as the true gradient $\nabla F(w)$, and we have:*

$$\langle \mathbb{E}[AG(w)], \nabla F(w)\rangle \geq \frac{1}{2}\left((1-V)\, \|\nabla F(w)\|^2 - U\right)$$

We now state our main theorem, proving convergence for general (non-convex) cost functions.

Theorem 4 (non-convex convergence). *Let Assumptions 1, 2 and 3 hold. If $\gamma < \frac{1}{L}$ and $V < 1$, then*

$$\min_{k\in[T]} \mathbb{E}\, \|\nabla F(w_k)\|^2 \leq \frac{2(F(w^{loc}) - F(w_*))}{T\gamma(1-V)} + \frac{U}{1-V}$$

Collaboration initial point. Note that, in Theorem 4, we use w^{loc} as a starting point, which may be different from the initial point w_0 of the algorithm, depending on whether peers proceed to a local training before collaboration or not. In fact, when a peer pre-trains the model using its local data for a certain number of iterations, then its parameter vector sequence $\{w_k\}_{k>0}$ is already progressing towards a critical point of the empirical risk minimization subproblem, and therefore: $F(w_0)-F(w_*) \geq F(w^{loc})-F(w_*)$. However, it should be noted that, in order to benefit from collaboration with neighbors, the pre-training must stop before learning "too much" on the peer's local data. This statement is supported formally in our results: progress towards a critical point (positive scalar product in Lemma 2) is expected only if the parameter vector w_k is inside the subspace $\mathcal{W}$, which means that the gradient norm is still big enough to be improved ($\|\nabla F(w)\|^2 \geq \frac{U}{1-V}$). In other words, the number of pre-training iterations E is upper-bounded by E_{max}, defined as follows:

$$E_{max} = \max\left\{ k \,\middle|\, \|\nabla F(w_k)\|^2 \leq \frac{U}{1-V} \right\}$$

Personalization degree. The parameter μ and the number of pre-training iterations E are crucial when setting the type of our algorithm. As a matter of fact, we can switch to a standard collaborative learning by setting $\mu = 1$ for every peer and bypassing the pre-training step. When we set $\mu = 0$ and allow the pre-training, each peer learns a first model, then totally relies on its close neighbors to improve it. μ values anywhere between 0 and 1 offer an intermediary personalization degree. The filtering parameter σ has also an important role in the personalization process. When $\sigma = 0$, the peer does not accept any gradients, and only uses its own gradient estimation to update the parameter vector. Any other value for σ will eventually allow a certain number of neighbors to be accounted for during the aggregation step.

remark 1. *Peers are free to choose the parameter σ_i for the filtering step, but we set a constant σ for every peer, to simplify the analysis. Although we consider an honest setting involving only correct peers, our work can easily be extended to a Byzantine setting. In fact, our filtering component is already defending against basic attacks, such as crash failures, random values or extreme values, since vectors that are far from a peer's estimated gradient are discarded, depending on the σ value. When dealing with Byzantine attacks, filtering schemes based on norms can easily be fooled [20]. We can modify our algorithm by simply replacing averaging by a robust aggregation rule, in the aggregation step [21–24].*

4 METHODOLOGY

In this section, we describe our proposed methodology for computing the energy consumption of the experiments (i.e., ML algorithms)

Table 1: Hardware and software characteristics of the considered server for the single-instance use case.

Component	Technology	Characteristics
CPU	Intel Xeon W-2123[2]	1.2 GHz of min and 3.6 GHz of max speeds
RAM	DDR$_4$ 1600	Capacity of 32 GB
OS	Linux	Ubuntu 20.04 LTS

presented in Section 5.2. To this end, two different hardware configurations are considered: server and mobile devices. For the first case, a single server executes the corresponding algorithm of peer-to-peer learning (e.g., P3) and centralized learning (e.g., CL) separately. Note that the peer-to-peer environment of the P3 algorithm is realized by simulating it on this single server instance. For the second case, the corresponding P3 algorithm is run on multiple mobile devices, thus realizing the real peer-to-peer environment. Figure 1 illustrates the hardware (e.g., server, mobile devices) as well as software (e.g., P3 and CL algorithms) setup of the two aforementioned cases. Our goal is to highlight the advantage of P3 compared to centralized learning from an energy perspective, as it can be fully deployed on mobile devices.

REMARK 2. *The methodology presented in this section is a general approach that can be applied directly, or easily extended to evaluate the energy consumption of any machine learning algorithm running on a CPU for both servers and mobile phones.*

4.1 Server

In this setting, the single-server instance is used to execute the corresponding peer-to-peer (e.g., P3) and centralized (e.g., CL) learning algorithms under study.

4.1.1 Configurations and Dataset. For our experiments, the underlying server has the hardware and software characteristics given in Table 1. The processor of the server is equipped with 4 physical cores, whereas the implementation of the corresponding algorithms is realized using Python 3.8, as well as the PyTorch 1.9, Numpy 1.19 and Scikit-learn 1.0 libraries. The algorithms under study generated different models by considering the MNIST[1] (Modified National Institute of Standards and Technology) dataset. Such a dataset consists of 60k and 10k sample points, for training and testing purposes, respectively. To study the impact of the dataset size on the energy consumption of the P3 and CL algorithms, the size of the MNIST dataset is increased by a factor of 4, 16 and 32, with respect to the original size (e.g., 60MB). Additionally, the cases of 10, 100 and 300 peers were considered for the P3 algorithm, whereas the number of iterations (e.g., epochs) of the algorithms are set to 2 and 10.

4.1.2 Power Monitoring. As mentioned above, the CPU of the corresponding server is equipped with 4 physical cores operated by the DVFS technology [25, 26] , such that each core can operate between minimum and maximum frequencies. To identify the optimal configuration of the CPU's frequency from the energy consumption perspective, the following steps were carried out: (1) using the Linux operating system's "userspace" governor, the specific core of the CPU was configured to operate in the three different frequencies MIN (1.2 GHz), MID (2.4 GHz) and MAX (3.6 GHz); (2) the *powerstat*[3] software package was used to monitor system- and power-relevant information such as the average speed (e.g., frequency), utilization, as well as power-demand of the CPU. This software package started and ended with the execution of the corresponding algorithms under study. Hence, the aforementioned monitored values were written onto a log file once the corresponding algorithm terminated. To reduce the overhead of the *powerstat* software package, it was executed on a core different from the one on which the algorithms were running.

4.1.3 Pinning, Shielding and Consumption Calculations. The experiments were carried out by disabling the virtualization technology of the processor. Furthermore, each of the algorithms under study was pinned to a specific core of the underlying quad-core CPU. In order to ensure that the algorithms were not disturbed by background routines of the operating system as well as other running applications, the core on which the algorithms were pinned was also shielded. Consequently, it is ensured that only the corresponding algorithm ran on that specific core of the CPU. Finally, to ensure that the energy consumption of the algorithms under study is captured adequately: first, the frequency of the 4 cores of the CPU was set to the minimum (e.g., 1.2 GHz); then, the server was kept for 10 minutes at the idle state, and the mean power of the whole CPU was calculated. Afterwards, the algorithms under study were executed on a specific core by pinning and shielding, as explained above. The frequency of that core was set to the aforementioned three options of MIN, MID and MAX, whereas the frequency of the other three cores was set to the minimum of 1.2 GHz. As a matter of fact, the *powerstat* software package reported an average frequency of the CPU (e.g., 4 cores) of 1.2, 1.38 and 1.77 GHz for MIN, MID and MAX settings, respectively. Subsequently, we obtain the mean power demand of the corresponding algorithm for each setting. This is done by calculating the difference between (1) the average demand of the CPU of the corresponding setting (e.g., MIN, MID, and MAX) and (2) the average demand of the CPU at the idle state. Finally, the energy consumption (Joules) of the algorithms under study is calculated by multiplying the execution time (seconds) by the mean power demand (Watt).

4.1.4 Annotation-based Power Profiling. For a fine-grain monitoring of the energy consumption of the corresponding algorithms, a power-profiler software feature was developed based on the profiling library[4] of Python. The corresponding profiler can be integrated directly inside specific places of the algorithms, by means of annotations. To this end, at the highest level, two power-profiling annotations were added for local training and collaborative learning phases, for the case of the P3 algorithm. Furthermore, additional annotations were added inside each of the aforementioned two phases, in order to obtain further insights about the frequency and the total amount of time for each method called in each phase. Consequently, in this way, an in-depth overview can be obtained on the most energy-consuming methods and phases of the P3 algorithm.

[1]http://yann.lecun.com/exdb/mnist/

[3]https://manpages.ubuntu.com/manpages/xenial/man8/powerstat.8.html
[4]https://docs.python.org/3/library/profile.html

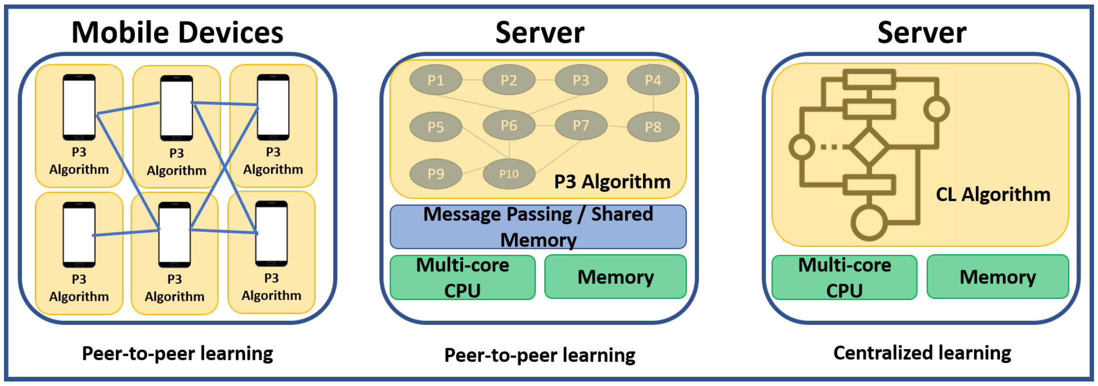


Figure 1: Graphical presentation of the proposed methodology for the single and multiple instances while executing peer-to-peer (P3 algorithm) as well as centralized learning (CL algorithm).

4.2 Mobile Devices

Under this setting, the P3 algorithm was realized in a real peer-to-peer environment using multiple mobile devices, as shown on the left box in Figure 1.

4.2.1 Configurations and Dataset. It was assumed that all the nodes (e.g., mobile phones) have similar hardware and software (e.g., operating system version) characteristics, which are given in Table 2. The P3 algorithm was implemented using Python 3.8 and Numpy 1.19 libraries. Regarding the dataset, we use the same source (e.g., MNIST) and the same configuration as the single server case (see Section 4.1.1).

4.2.2 Power and Energy Monitoring. Each node participating in the peer-to-peer learning runs the P3 algorithm locally. For the purpose of monitoring and collecting the energy consumption of the corresponding algorithm, an open-source Android-based monitoring system was developed. The main objective of this tool is to collect, based on the configured time interval (e.g., every second), relevant KPIs (key performance indicators) of the selected mobile application. To this end, the monitoring system can log, for instance: the voltage, the current, the utilization of the CPU, the frequency of the CPU, to name it few. All those KPIs are then used to compute the energy consumption of the P3 algorithm. More details on the monitoring system can be found in Appendix B.1

4.2.3 Energy Consumption Estimation. To appropriately calculate the energy consumption of the algorithm under study, we first disable all the unnecessary components such as GSM, GPS and Bluetooth. Then, we run the monitoring application for 10 minutes long under these conditions to calculate the mean power of the Android device at the idle state. Afterward, the algorithm is executed and monitored to get the power usage during execution. Finally, we subtract the idle power to calculate the actual energy consumption of the algorithm.

Table 2: Hardware and software characteristics of the considered mobile devices.

Component	Technology	Characteristics
CPU	Qualcomm SDM710 Snapdragon 710	1.7 GHz of min and 2.2 GHz of max speeds
RAM	DDR4	Capacity of 4 GB
OS	Android	Version 11

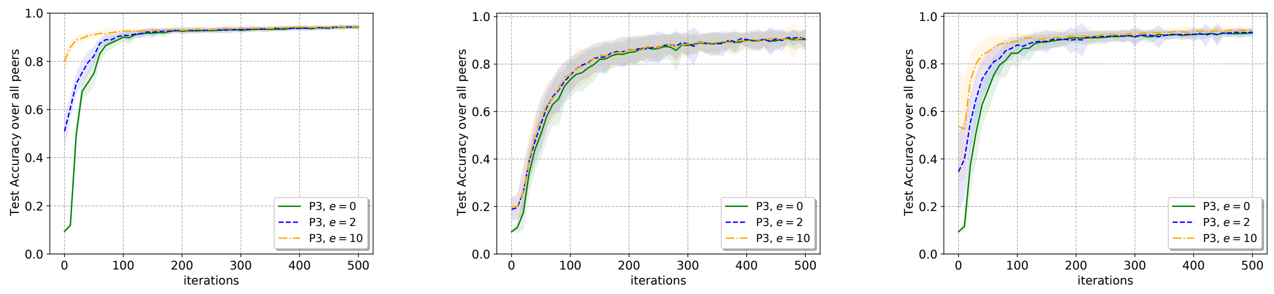
5 EVALUATION

In this section, we present an exhaustive set of experiments to illustrate the practical significance of our P3 algorithm in terms of convergence and energy efficiency. We evaluate P3 under several configurations, and compare it with two other state-of-the-art algorithms for reference, such as FedAvg [2] in a federated learning scheme and Model Propagation [13] in a P2P scheme.

5.1 Performance Evaluation

We evaluate the performance of P3 on server instances and Android mobile phones. In the following, we present the experimental setup, the considered metrics, and the obtained results.

5.1.1 Experimental Setup. We consider a set of 100 peers connected in a P2P network graph built using a random network generator following the Erdős–Rényi model. We randomly sample the number of neighbors for each peer from a uniform distribution defined by the graph density parameter $\rho \in [0, 1]$. The weight W_{ij} between peer i and j is given by the random network generator or by using the auxiliary information of the peers. The personalization parameter μ_i was set to 0.5, which gave the best results on a held-out set of random problem instances. μ_i can be increased if the peer has more confidence in its local data. The value of σ can be either (1) set manually to control the computation complexity and the model personalization threshold, (2) estimated during the local training by calculating the median of the norm squared between model updates, or (3) estimated in the collaborative phase by taking the median of the received gradients in the first collaborative round. In our experiments, we manually set the value of σ to 0.3 in the IID partitioning, 5 in the balanced non-IID partitioning, and 1 in the unbalanced non-IID partitioning. In all conducted experiments for P3, we use two local epochs $e = 2$ for the local learning phase, unless stated otherwise.



(a) IID data partitioning **(b) Balanced non-IID partitioning** **(c) Unbalanced non-IID partitioning**

Figure 2: Training a feed-forward neural network on the MNIST dataset, partitioned in different configurations over $n = 100$ peers. Each experiments is done with different local training epochs e to demonstrate its effect on the convergence rate.

We choose image classification as a learning task performed by each peer. For this, we train a deep neural network using the *MNIST* dataset. *MNIST* consists of 10 categories, including digits ranging from 0 to 9, with a total of 70,000 data samples (60,000 for training and 10,000 for testing). We divide the dataset between peers in three different ways:

- **IID data partitioning:** The data is shuffled, and then evenly partitioned between 100 peers, each receiving 600 samples.
- **Balanced non-IID partitioning:** The data is first sorted by digit label. We divide it into 200 shards of size 300, and assign 2 shards to each of the 100 peers. Each peer receiving 600 samples containing only two categories of digits.
- **Unbalanced non-IID partitioning:** The data is first sorted by digit label; we divide it into 200 shards of size 300, and assign a random number of shards to each of the 100 peers.

All peers are assigned the same test set to assess the quality of the trained models. We use the mean test accuracy over all peers for model evaluation, unless stated otherwise. The experiments in this section were conducted on a server with the characteristics described in Table 1, and an Android device with the characteristics described in Table 2. The source code was written in Python 3.8, and the machine learning models were built using PyTorch 1.9 for the server and Numpy 1.19 for Android. The source code of our implementation is available on https://anonymous.4open.science/r/P3-86D3.

The focus of this set of experiments is to demonstrate that P3 can learn optimal personalized models under both IID and non-IID settings. Furthermore, we compare P3 against various algorithms that learn either global or personalized models, and we confirm its advantage compared to traditional personalized schemes in terms of final accuracy of the model.

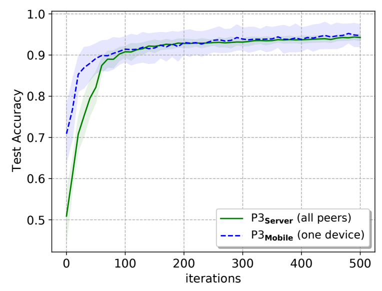
5.1.2 Experimental Results.

Convergence rate. Figure 2a demonstrates that P3 converges to an optimal model after a number of collaborative rounds. We observe that the number of local training epochs e performed by peers before joining the collaboration greatly affects the convergence speed. However, performing too many local training epochs will result in overfitting to the local data, and poor generalization over the network. Furthermore, as local training is computationally expensive and energy-consuming, weak devices can skip the local learning step and still converge to the same model (e.g., P3, $e = 0$) at the expense of convergence time.

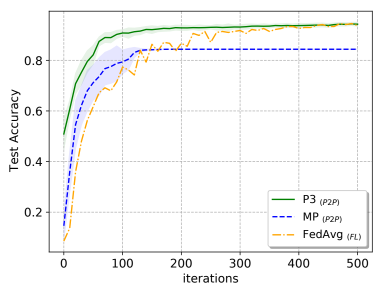
Convergence under heterogeneous partitioning. In a realistic P2P setting, peers are likely to hold non-IID data. In this set of experiments, we show that P3 converges even in the extreme case of non-IIDness, where the peer’s data scope is limited to only two label classes. Figure 2 tracks the mean test accuracy of the trained models under different heterogeneous partitioning. We observe in Figure 2b that P3 converges even if the data is extremely heterogeneous, and achieves an accuracy comparable to the IID setting (Figure 2a) but with a slower convergence rate. However, Figure 2c shows that in unbalanced non-IID partitioning (where peers may have more than 2 categories of samples), the convergence rate is comparable with the IID setting.

Convergence on Android devices. Figure 3a shows the accuracy of the model running on the mobile device ($P3_{Mobile}$) in conjunction with the rest of the 100 peers ($P3_{Server}$). We observe that $P3_{Mobile}$ converges similarly to all the other 99 peers running on a server instance. The fluctuation in accuracy (blue area) is a result of random samples and neighbors attributed to the mobile device in each execution. Furthermore, the time required by the mobile device to finish one iteration is greater than peers running on computers; however, it is much more energy-efficient (See section 5.2.2).

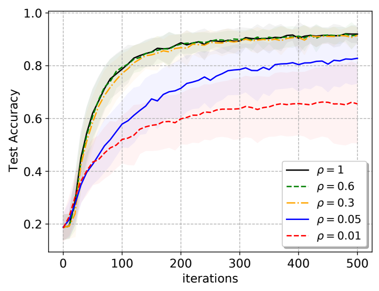
Comparing P3 to prior works. In this experiment, we study the convergence behavior of P3 with two different schemes, a federated learning scheme using FedAvg[2] and a P2P scheme using Model Propagation[13]. In the federated learning scheme, we rely on a server to aggregate the received updates from all the clients and use the aggregated models to learn a single global model. Every client performs one local training epoch per iteration similar to the computation performed by peers in P3. In contrast, Model Propagation runs in a P2P network, and each peer learns its own personalized model similarly to P3. Figure 3b investigates the convergence behavior of the three algorithms. We observe that P3 has a better starting accuracy, thanks to the local training phase, and converges faster compared to FedAvg. In comparison to Model Propagation (MP), P3 converges to a better model under the same conditions.



(a) P3 running on a mobile device



(b) P3 vs. FedAvg and Model Propagation



(c) P3 under different values of graph density.

Figure 3: Convergence behavior of P3. (a) P3 while running on a mobile device compared with the full set of peers. (b) P3 compared to FedAvg and Model Propagation (MP). P3 achieves good results compared to both schemes. (c) P3 under different values of graph density. P3 converges to optimal models if the graph is not extremely sparse.

Effect of network density. To study the effect of network density, we consider an Erdős–Rényi graph generator with a graph density parameter $\rho_{\in[0,1]}$ to control the density of the generated graph. Small values of ρ result in sparse graphs (i.e., few neighbors), while increasing the values of ρ results in more dense graphs (i.e., more neighbors). In a denser graph, peers will have to preform more computation (implying a higher energy consumption). Figure 3c illustrates the effect of network density on the performance of P3. If the graph is not extremely sparse, P3 can still converge to optimal models with less computation overhead. However, in extremely sparse graphs (e.g., $\rho = 0.01$) P3 tend to perform poorly.

5.2 Energy Analysis

In this section, different energy consumption comparisons are given. In Section 5.2.1, we compare centralized (e.g., CL) and peer-to-peer (e.g., P3) learning algorithms by considering a single server setting. We present the results of the analysis and identify the most energy-efficient execution of those two algorithms under this setting. In Section 5.2.2, we go one step further and compare single-server against multiple mobile devices settings of our P3 algorithm.

5.2.1 *P3 versus CL on a Server Setting.*

Optimal Consumption. The main objective of this analysis is to determine the most energy-efficient setup for the P3 and CL algorithms. To achieve this, we consider the methodology of Section 4.1. With the three considered frequencies MIN, MID, and MAX of the underlying CPU, the computed average power of the considered algorithms was 4, 6.5 and 19.5 Watts, respectively.

Figure 4 shows the energy consumption results of the P3 and CL algorithms by considering the aforementioned three frequency ranges of the single server. Note that configuring the core of the CPU at the middle frequency results in the most energy-efficient processing both for our P3 and CL algorithms. All other experiments showed the same results. However, due to space limitations, we only give the results for the case of 2 epochs. Consequently, we conjecture that the most energy-efficient way of executing the P3 and CL algorithms is to set the frequency of the CPU at the middle value (e.g., 2.4 GHz in our case).

Head-to-head. Figure 5 illustrates the comparison of the most energy-efficient execution of our P3 and CL algorithms, by considering a dataset 32 times larger than MNIST (1.87 GB) for 10 epochs. For the peer-to-peer environment of our P3 algorithm, we consider the cases of 10, 100 and 300 peers. Note that this was the maximum limit on the number of peers that can be simulated on the considered single-server instance.

The CL algorithm has a constant energy consumption, due to the fact that there is no notion of peers in the case of centralized learning. More precisely, the CL algorithm generates one generic model by considering the whole dataset. On the other hand, for the case of our P3 algorithm, one can see that for number of peers between 10 and 100, both algorithms have comparable energy consumption, with the P3 algorithm consuming 1.2 and 1.4 times more than the CL algorithm. For larger number of peers (i.e., 300), the gap between the two algorithms in terms of energy consumption becomes larger. This is because our P3 algorithm consists of local and collaborative training phases.

To analyse the energy consumed by each of those two phases, we monitored them using the power profiler of Section 4.1.4. Table 3 gives the results of the carried-out analysis. It can be seen that the energy consumed by the local training phase for different number of peers is almost identical. Furthermore, for small number of peers (i.e., 10 and 100), local training is the predominant phase. However, for large number of peers (i.e., 300), both phases have similar energy consumption, such that the energy consumption of the collaborative learning increases with increasing number of peers.

To conclude the comparison between those two algorithms with the single server setting, we believe that our P3 algorithm will have the advantage over the CL algorithm, when considering larger sizes (e.g., 10 GB and above) of the corresponding dataset even in the single-server setting. The reason is that, for the CL algorithm, the time required to generate one generic model increases exponentially with the size of data. This is not the case for our P3 algorithm, which generates models based on a subset of the dataset.

Discussion. Unlike CL, our P3 algorithm can be run on multiple servers, where each one can be considered as a peer. Furthermore, under this setting, it can be assumed that the number of peers do

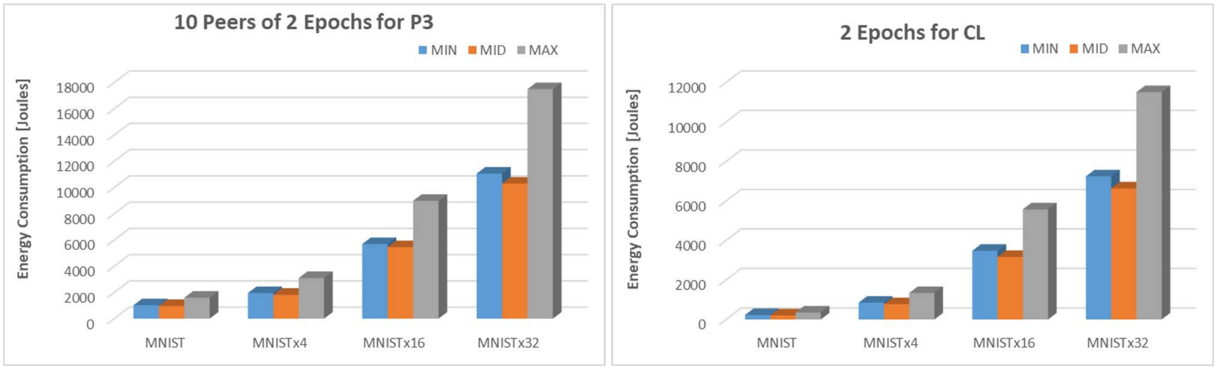


Figure 4: Energy consumption of the P3 and CL algorithms, considering minimum (1.2 GHz), middle (2.4 GHz) and maximum (3.6 GHz) frequencies of the single server.

Table 3: Energy consumption in Joules of the two components of local and collaborative training of our P3 algorithm.

Number of Peers	Local Training [J]	Collab Training [J]
10	37792	771
100	40064	7070
300	48455	30979

not exceed 20. For instance, a group of companies or hospitals can execute our P3 algorithm collaboratively. It is shown in Figure 5 that both our P3 and CL algorithms have a similar energy consumption. However, our P3 algorithm does not have the well-known drawbacks of centralized learning algorithms such as (1) single point of failure, (2) data security issues and (3) data privacy issues. Moreover, the proposed P3 algorithm has the advantage of deriving personalized models for each peer, which is not possible in the case of any centralized machine-learning algorithm. Considering all this, we believe that the main use case of our P3 algorithm is its execution on mobile devices.

5.2.2 Server versus Mobile Devices Settings. Figure 6 shows the results of our comparison between our P3 algorithm, both with server and mobile device settings, and the CL algorithm with the server setting. For all three considered scenarios, the size of the dataset was about 1.87 GB. Additionally, for our P3 algorithm, we considered settings with 10, 100, and 300 peers. The results for the P3 algorithm with the mobile device setting were realized using the configuration of Section 4.2.1 and the developed android-based monitoring system of Section 4.2.2.

One can see that our P3 algorithm (P3-mobile), when running on ARM-based processors of mobile devices, is extremely energy-efficient compared to the two other scenarios (P3-server and CL-server) when running on RISC-based processors of the server. To understand the significant gain in energy-efficiency of our P3 algorithm when running on mobile devices, Figure 7 gives the total execution times of the three considered scenarios. Note that the execution times for the CL algorithm are the same for 10, 100, and 300 peers. The reason is that, in the CL algorithm, there is no notion of peers. Comparing the execution times of the three considered scenarios, we can see that the P3 algorithm, when running on mobile devices, has execution times close to those of the two other scenarios running on a server setting. However, the major gain is in the power demand, where the P3 algorithm on mobile devices

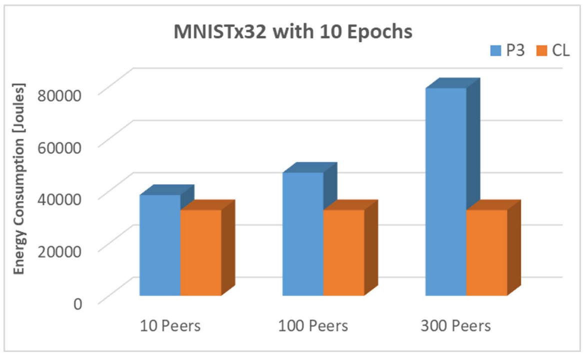


Figure 5: Energy consumption comparison between the P3 and CL algorithms for the MNISTx32 dataset and 10 epochs.

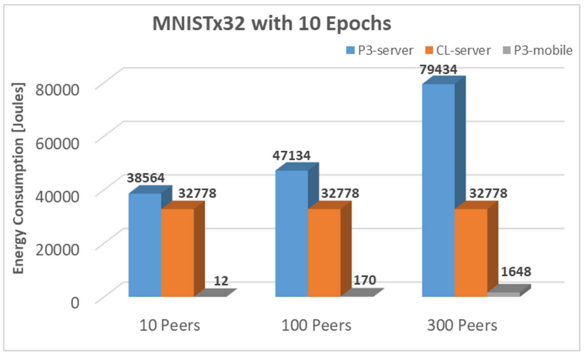


Figure 6: Energy consumption comparison between the P3 and CL algorithms running on server setting, and the P3 algorithm running on mobile devices setting, for the MNISTx32 dataset and 10 epochs.

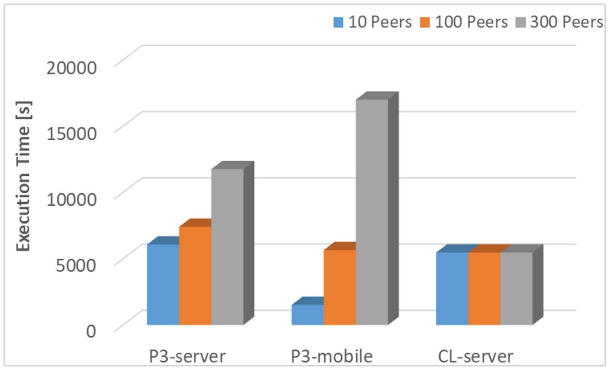


Figure 7: Execution times (in seconds) of the P3 and CL algorithms running on the server setting, and P3 algorithm running on mobile devices setting, for the MNISTx32 dataset and 10 epochs.

has an average demand of 796 μW, 300 μW and 320 μW, for 10, 100, and 300 peers, respectively. On the other hand, both P3 and CL algorithms, when running on the single server setting, have an average demand of 6 W.

This leads us to the conclusion that peer-to-peer machine-learning algorithms, like the one we proposed in this paper, can take advantage of being executed on ARM-based processors. In addition to the advantages with respect to centralized machine learning algorithms, our results show that peer-to-peer algorithms consume significantly less energy than the centralized ones, that need to be executed on RISC-based processors. In our case, P3-mobile requires 2700x, 200x and 20x less energy than CL-server for the cases of 10, 100 and 300 peers, respectively.

6 RELATED WORK

Most work in decentralized machine learning has focused on distributed optimization techniques to learn one global model that minimizes the local losses of all peers [27–32]. In this paper, we consider a P2P setting where peers have total control over the models they want to learn according to their local datasets and personalized objective. Considerable work [13–16, 33] has been published in this area of research. For instance, [13] relies on alternative direction method of multipliers [34] to solve the decentralized objective function. The authors in [14] tackled the privacy issue related to gradient exchange and proposed a differentially private algorithm to learn personalized models under strong privacy requirements. [16] suggested a Robust algorithm to fight against malicious peers, and extended his work to support dynamic networks where peers can change their neighbors if they are malicious or have no beneficial information. To achieve lower communication costs, [15] proposed an algorithm to simultaneously learn the network graph with the models. Nevertheless, none of these works analyzed the energy requirements of their proposed algorithms.

As a matter of fact, machine-learning algorithms are traditionally used to generate forecasting models for demand [35–37] and generation [38–40] in energy field. However, the topic of estimating the energy consumption of machine-learning algorithms has not yet been studied adequately in the literature [41]. The authors in [10] proposed a methodology based on Python programming language to measure the power demand of deep neural networks. Similar to our approach, the proposed methodology is based on annotation markers to identify the energy consumption of the different layers of the network. DeLight was proposed in [12], where the authors took energy consumption as one of the optimization criteria while inferring the deep neural network. In [11] the NeuralPower was proposed, which is a collection of regression-based estimation models of energy consumption for the different layers of the network. The corresponding estimation models were built by considering real power demand values using nvidia-smi software tool. Unlike this approach, we do not provide power and energy prediction models: in our experiments, we obtained real power values monitored directly at the CPU level.

The main goal of our paper is to demonstrate the advantage of peer-to-peer learning when running on mobile devices. Hence, unlike the above-mentioned contributions, we propose a methodology to calculate the energy consumption of the machine-learning algorithms under study (i.e. P3 and CL), both for server and mobile devices settings. The proposed methodology serves to carry out a comparison between the two algorithms and identify the most energy-efficient setting. To the best of our knowledge, this is the first attempt in the literature to provide such a comparison and analysis. Additionally, as one of the contributions of this paper, we demonstrate through carried-out experiments that the energy consumption of the peer-to-peer algorithm is significantly lower, when running on mobile devices, than that of the server setting.

7 CONCLUSION

Decentralized algorithms are usually considered for their advantages w.r.t. scalability, high availability (i.e., resilience to SPOFs), and strong privacy guarantees. In this paper, we highlight another major advantage over centralized architectures, namely: *energy efficiency*. To support our claim, we introduced a novel methodology for estimating the energy requirements of ML algorithms running on CPUs. To our best knowledge, this is the first work that proposes a methodology for evaluating the energy consumption of P2P algorithms. Motivated by the results of our energy analysis, we designed P3, a new P2P algorithm to learn personalized models based on every peer's local data and computation capabilities (energy requirements). We demonstrate, both theoretically and practically, that P3 converges to optimal models under realistic assumptions. Interestingly, we show that running P3 on mobile devices is extremely energy-efficient, consuming 2700x, 200x and 20x less energy than classical centralized algorithms for a network of 10, 100, and 300 peers, respectively, without altering the convergence properties (convergence rate and statistical error).

As future work, we plan to consider the presence of Byzantine peers and support dynamic networks, where peers can join or leave the network, or even disconnect from irrelevant neighbors. We can also consider using GPUs instead of CPUs in the experiments involving the server, since GPUs are notoriously known for their energy efficiency.

REFERENCES

[1] M. Li, L. Zhou, Z. Yang, A. Li, F. Xia, D. G. Andersen, and A. Smola, "Parameter server for distributed machine learning," in *Big learning NIPS workshop*, vol. 6, 2013, p. 2.

[2] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics (AISTATS)*, 2017.

[3] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[4] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, 2014, pp. 583–598.

[5] R. Basmadjian, "Flexibility-based energy and demand management in data centers: a case study for cloud computing," *Energies*, vol. 12, no. 17, p. 3301, 2019.

[6] R. Basmadjian, P. Bouvry, G. Costa, L. Gyarmati, D. Kliazovich, S. Lafond, L. Laurent, H. Meer, J.-M. Pierson, R. Pries *et al.*, "Green data centers," *Large-Scale Distributed Systems and Energy Efficiency: A Holistic View*, pp. 159–196, 2015.

[7] Y. Liu, X. Wei, J. Xiao, Z. Liu, Y. Xu, and Y. Tian, "Energy consumption and emission mitigation prediction based on data center traffic and PUE for global data centers," *Global Energy Interconnection*, vol. 3, no. 3, pp. 272–282, Jun. 2020.

[8] M. Avgerinou, P. Bertoldi, and L. Castellazzi, "Trends in data centre energy consumption under the european code of conduct for data centre energy efficiency," *Energies*, vol. 10, no. 10, p. 1470, Sep. 2017.

[9] D. Patterson, J. Gonzalez, U. Hölzle, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, and J. Dean, "The carbon footprint of machine learning training will plateau, then shrink," *arXiv preprint arXiv:2204.05149*, 2022.

[10] C. F. Rodrigues, G. Riley, and M. Lujan, "Fine-grained energy profiling for deep convolutional neural networks on the jetson TX1," in *2017 IEEE International Symposium on Workload Characterization (IISWC)*. IEEE, Oct. 2017.

[11] E. Cai, D.-C. Juan, D. Stamoulis, and D. Marculescu, "NeuralPower: Predict and deploy energy-efficient convolutional neural networks," Oct. 2017.

[12] B. D. Rouhani, A. Mirhoseini, and F. Koushanfar, "Delight: Adding energy dimension to deep neural networks," in *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*, ser. ISLPED '16, 2016, p. 112–117.

[13] P. Vanhaesebrouck, A. Bellet, and M. Tommasi, "Decentralized collaborative learning of personalized models over networks," in *Artificial Intelligence and Statistics (AISTATS)*, 2017.

[14] A. Bellet, R. Guerraoui, M. Taziki, and M. Tommasi, "Personalized and private peer-to-peer machine learning," in *Artificial Intelligence and Statistics (AISTATS)*, 2018.

[15] V. Zantedeschi, A. Bellet, and M. Tommasi, "Fully decentralized joint learning of personalized models and collaboration graphs," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

[16] K. Boubouh, A. Boussetta, Y. Benkaouz, and R. Guerraoui, "Robust p2p personalized learning," in *2020 International Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2020, pp. 299–308.

[17] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018.

[18] S. Shalev-Shwartz and T. Zhang, "Stochastic dual coordinate ascent methods for regularized loss," *J. Mach. Learn. Res.*, vol. 14, pp. 567–599, 2013.

[19] R. M. Gower, N. Loizou, X. Qian, A. Sailanbayev, E. Shulgin, and P. Richtárik, "SGD: General analysis and improved rates," in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, 09–15 Jun 2019, pp. 5200–5209.

[20] E. M. El Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in Byzantium," in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., vol. 80, 2018, pp. 3521–3530.

[21] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," 2018.

[22] C. Xie, O. Koyejo, and I. Gupta, "Phocas: dimensional byzantine-resilient stochastic gradient descent," 2018.

[23] A. Boussetta, E.-M. El-Mhamdi, R. Guerraoui, A. Maurer, and S. Rouault, "AKSEL: Fast Byzantine SGD," in *24th International Conference on Principles of Distributed Systems (OPODIS 2020)*, vol. 184, 2021, pp. 8:1–8:16.

[24] C. Xie, O. Koyejo, and I. Gupta, "Generalized byzantine-tolerant sgd," *arXiv preprint arXiv:1802.10116*, 2018.

[25] R. Basmadjian, F. Niedermeier, and H. de Meer, "Modelling performance and power consumption of utilisation-based dvfs using m/m/1 queues," in *Proceedings of the Seventh International Conference on Future Energy Systems*, 2016, pp. 1–11.

[26] R. Basmadjian and H. de Meer, "Modelling and analysing conservative governor of dvfs-enabled processors," in *Proceedings of the Ninth International Conference on Future Energy Systems*, 2018, pp. 519–525.

[27] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: Convergence analysis and network scaling," *IEEE Transactions on Automatic Control (TACON)*, pp. 592–606, 2012.

[28] Z. Jiang, A. Balu, C. Hegde, and S. Sarkar, "Collaborative deep learning in fixed topology networks," in *International Conference on Neural Information Processing Systems (NIPS)*, 2017, pp. 5906–5916.

[29] H. Tang, X. Lian, M. Yan, C. Zhang, and J. Liu, "D2: Decentralized training over decentralized data," in *International Conference on Machine Learning (ICML)*, 2018.

[30] A. Nedic and A. E. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control (TACON)*, vol. 54, 2009.

[31] S. S. Ram, A. Nedic, and V. V. Veeravalli, "Distributed stochastic subgradient projection algorithms for convex optimization," *Journal of Optimization Theory and Applications (JOTA)*, pp. 516–545, 2010.

[32] E. Wei and A. E. Ozdaglar, "Distributed alternating direction method of multipliers," *IEEE Conference on Decision and Control (CDC)*, pp. 5445–5450, 2012.

[33] I. Almeida and J. M. F. Xavier, "Djam: Distributed jacobi asynchronous method for learning personal models," *IEEE Signal Processing Letters (SPL)*, vol. 25, pp. 1389–1392, 2018.

[34] S. P. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, pp. 1–122, 2011.

[35] S. Bouktif, A. Fiaz, A. Ouni, and M. Serhani, "Optimal deep learning LSTM model for electric load forecasting using feature selection and genetic algorithm: Comparison with machine learning approaches," *Energies*, Jun. 2018.

[36] N. Shabbir, R. Ahmadiahangar, L. Kutt, and A. Rosin, "Comparison of machine learning based methods for residential load forecasting," in *2019 Electric Power Quality and Supply Reliability Conference (PQ) & 2019 Symposium on Electrical Engineering and Mechatronics (SEEM)*. IEEE, Jun. 2019.

[37] S. Jurado, À. Nebot, F. Mugica, and N. Avellana, "Hybrid methodologies for electricity load forecasting: Entropy-based feature selection with machine learning and soft computing techniques," *Energy (Oxf.)*, vol. 86, pp. 276–291, Jun. 2015.

[38] M. N. Rahman, A. Esmailpour, and J. Zhao, "Machine learning with big data an efficient electricity generation forecasting system," *Big Data Research*, 2016.

[39] K. Mahmud, S. Azam, A. Karim, S. Zobaed, B. Shanmugam, and D. Mathur, "Machine learning based pv power generation forecasting in alice springs," *IEEE Access*, vol. 9, pp. 46 117–46 128, 2021.

[40] N. Sharma, P. Sharma, D. Irwin, and P. Shenoy, "Predicting solar generation from weather forecasts using machine learning," in *2011 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. IEEE, Oct. 2011.

[41] E. García-Martín, C. F. Rodrigues, G. Riley, and H. Grahn, "Estimation of energy consumption in machine learning," *Journal of Parallel and Distributed Computing*, vol. 134, pp. 75–88, 2019.

A PROOFS OF THEORETICAL RESULTS

In this section, we present the proofs of our theoretical findings. For convenience, we restate the results, as appearing in the main paper.

A.1 Proof of Lemma 1

LEMMA 3. *Let σ be the threshold set by a peer i and $AG(w_k)$ be the aggregated gradient at w_k. The following holds true for every $w_k \in \mathbb{R}^d$:*

$$\mathbb{E}\left\|AG(w_k) - \nabla F(w_k)\right\|^2 \leq U + V\left\|\nabla F(w_k)\right\|^2$$

where $U = 2((1-\mu)^2\sigma^2 + M)$ and $V = 2(1 + N - \rho)$.

PROOF. We first upper bound the error of a local gradient estimation. Let $G_i(w_k)$ be the estimation of the gradient computed by peer i at round k. We have then:

$$\begin{aligned}\mathbb{E}\left\|G_i(w_k) - \nabla F(w_k)\right\|^2 &= \mathbb{E}\left\|G_i(w_k)\right\|^2 - \langle \mathbb{E}[G_i(w_k)], \nabla F(w_k)\rangle + \left\|\nabla F(w_k)\right\|^2 \\ &\leq M + N\left\|\nabla F(w_k)\right\|^2 - \rho\left\|\nabla F(w_k)\right\|^2 \\ &= M + (1 + N - \rho)\left\|\nabla F(w_k)\right\|^2 \end{aligned} \tag{3}$$

We now derive an upper bound on the aggregated vector $AG(w_k) = \mu G_i(w_k) + (1-\mu)A(w_k)$, where $A(w_k)$ is the average of the accepted gradients. Let p be the number of gradients $A_j(w_k)$ that passed the angular filter. We have then:

$$\begin{aligned}\mathbb{E}\left\|AG(w_k) - \nabla F(w_k)\right\|^2 &= \mathbb{E}\left\|\mu G_i(w_k) + (1-\mu)A(w_k) - \nabla F(w_k)\right\|^2 \\ &= \mathbb{E}\left\|\mu G_i(w_k) + (1-\mu)\left(\frac{1}{p}\sum_{j=1}^{p} A_j(w_k)\right) - \nabla F(w_k)\right\|^2 \\ &= \mathbb{E}\left\|\mu G_i(w_k) - G_i(w_k) + G_i(w_k) + (1-\mu)\left(\frac{1}{p}\sum_{j=1}^{p} A_j(w_k)\right) - \nabla F(w_k)\right\|^2 \\ &= \mathbb{E}\left\|-(1-\mu)G_i(w_k) + (1-\mu)\left(\frac{1}{p}\sum_{j=1}^{p} A_j(w_k)\right) + G_i(w_k) - \nabla F(w_k)\right\|^2 \\ &= \mathbb{E}\left\|(1-\mu)\left(\left(\frac{1}{p}\sum_{j=1}^{p} A_j(w_k)\right) - G_i(w_k)\right) + G_i(w_k) - \nabla F(w_k)\right\|^2 \\ &= \mathbb{E}\left\|(1-\mu)\left(\frac{1}{p}\sum_{j=1}^{p}(A_j(w_k) - G_i(w_k))\right) + G_i(w_k) - \nabla F(w_k)\right\|^2 \end{aligned}$$

Using the inequality $\|a-b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$, we obtain:

$$\begin{aligned}\mathbb{E}\left\|AG(w_k) - \nabla F(w_k)\right\|^2 &\leq 2\,\mathbb{E}\left\|(1-\mu)\left(\frac{1}{p}\sum_{j=1}^{p}(A_j(w_k) - G_i(w_k))\right)\right\|^2 + 2\,\mathbb{E}\left\|G_i(w_k) - \nabla F(w_k)\right\|^2 \\ &\leq 2\left(\frac{1-\mu}{p}\right)^2 \mathbb{E}\left\|\sum_{j=1}^{p}(A_j(w_k) - G_i(w_k))\right\|^2 + 2\,\mathbb{E}\left\|G_i(w_k) - \nabla F(w_k)\right\|^2 \end{aligned}$$

Applying Cauchy-Schwartz in the last inequality gives:

$$\mathbb{E}\left\|AG(w_k) - \nabla F(w_k)\right\|^2 \leq 2\left(\frac{1-\mu}{p}\right)^2 p\sum_{j=1}^{p}(\mathbb{E}\left\|A_j(w_k) - G_i(w_k)\right\|^2) + 2\,\mathbb{E}\left\|G_i(w_k) - \nabla F(w_k)\right\|^2$$

By construction of the algorithm, a peer i only accepts vectors $A_j(w_k)$ such that $\left\|A_j(w_k) - G_i(w_k)\right\|^2 \leq \sigma^2$. Using this fact and Inequality (3), we obtain:

$$\begin{aligned}\mathbb{E}\left\|AG(w_k) - \nabla F(w_k)\right\|^2 &\leq 2(1-\mu)^2\sigma^2 + 2(M + (1 + N - \rho)\left\|\nabla F(w_k)\right\|^2) \\ &= 2((1-\mu)^2\sigma^2 + M) + 2(1 + N - \rho)\left\|\nabla F(w_k)\right\|^2 \end{aligned}$$

which concludes the proof. □

A.2 Proof of Lemma 2

We first define the subspace $\mathcal{W} \subset \mathbb{R}^d$ as follows:

$$\mathcal{W} := \left\{ w \in \mathbb{R}^d \,\middle|\, (1-V)\,\|\nabla F(w)\|^2 - U \geq 0 \right\}$$

Lemma 4. *Let Assumptions 2 and 3 hold. When $w \in \mathcal{W}$, the expected aggregated vector $E[AG(w)]$ points in the same direction as the true gradient $\nabla F(w)$, and we have:*

$$\langle \mathbb{E}[AG(w)], \nabla F(w) \rangle \geq \frac{1}{2}\left((1-V)\,\|\nabla F(w)\|^2 - U \right)$$

Proof. Using Jensen's Inequality, we have for all $w_k \in \mathbb{R}^d$:

$$\begin{aligned}\|\mathbb{E}[AG(w_k)] - \nabla F(w_k)\|^2 &\leq \mathbb{E}\,\|AG(w_k) - \nabla F(w_k)\|^2 \\ &\leq \underbrace{U + V\,\|\nabla F(w_k)\|^2}_{r^2}\end{aligned}$$

where the last inequality comes directly from Lemma 1.

The fact that $\|\mathbb{E}[AG(w_k)] - \nabla F(w_k)\|^2 \leq r^2$ means that $\mathbb{E}[AG(w_k)]$ belongs to a ball centered at $\nabla F(w_k)$ with radius r. We have then:

$$\|\mathbb{E}[AG(w_k)] - \nabla F(w_k)\|^2 = \|\mathbb{E}[AG(w_k)]\|^2 - 2\langle \mathbb{E}[AG(w_k)], \nabla F(w_k)\rangle + \|\nabla F(w_k)\|^2 \leq r^2$$

$$2\langle \mathbb{E}[AG(w_k)], \nabla F(w_k)\rangle \geq \|\mathbb{E}[AG(w_k)]\|^2 + \|\nabla F(w_k)\|^2 - r^2 \tag{4}$$

Since $w_k \in \mathcal{W}$, which means that $U \leq (1-V)\,\|\nabla F(w_k)\|^2$, we also have:

$$\frac{r}{\|\nabla F(w_k)\|} \leq 1$$

Let us define $\theta \in [0, \frac{\pi}{2}]$ such that $\sin\theta = \frac{r}{\|\nabla F(w_k)\|}$. Furthermore, using the triangle inequality, we have $\|\mathbb{E}[AG(w_k]\| \geq \|\nabla F((w_k)\| - r$. Therefore, $\|\mathbb{E}[AG(w_k]\| \geq (1-\sin\theta)\,\|\nabla F((w_k)\|$. Using this in (4), we obtain:

$$\begin{aligned}2\langle \mathbb{E}[AG(w_k)], F(w_k)\rangle &\geq \|\mathbb{E}[AG(w_k)]\|^2 + \|\nabla F(w_k)\|^2 - r^2 \\ &\geq (1-\sin\theta)^2\,\|\nabla F(w_k)\|^2 + \|\nabla F(w_k)\|^2 - \sin^2\theta\,\|\nabla F(w_k)\|^2 \\ &= 2(1-\sin\theta)^2\,\|\nabla F(w_k)\|^2 .\end{aligned}$$

and therefore:

$$\begin{aligned}\langle \mathbb{E}[AG(w_k)], \nabla F(w_k)\rangle &\geq (1-\sin\theta)^2\,\|\nabla F(w_k)\|^2 \\ &\geq (1-\sin\theta)\,\|\nabla F(w_k)\|^2 \text{ (since } 0 \leq 1-\sin\theta \leq 1)\end{aligned}$$

Substituting $\sin\theta = \frac{r}{\|\nabla F(w_k)\|}$ above, we obtain:

$$\begin{aligned}\langle \mathbb{E}[AG(w_k)], \nabla F(w_k)\rangle &\geq \|\nabla F(w_k)\|^2 - r\,\|\nabla F(w_k)\| \\ &\geq \frac{1}{2}\left(\|\nabla F(w_k)\|^2 - r^2\right)\end{aligned}$$

Recall that $r^2 = U + V\,\|\nabla F(w_k)\|^2$. Substituting this above concludes the proof. □

A.3 Proof of the main theorem

Theorem 5 (non-convex convergence). *Let Assumptions 1, 2 and 3 hold. If $\gamma < \frac{1}{L}$ and $V < 1$, then*

$$\min_{k\in[T]} \mathbb{E}\,\|\nabla F(w_k)\|^2 \leq \frac{2(F(w^{loc}) - F(w_*))}{T\gamma(1-V)} + \frac{U}{1-V}$$

Proof. We proceed by upper bounding the norm of the gradient, depending on the parameter vector w's membership in the subspace $\mathcal{W}$. Trivially, when $w \notin \mathcal{W}$, we have, by construction of $\mathcal{W}$:

$$\forall w \notin \mathcal{W}, \quad \|\nabla F(w)\|^2 < \frac{U}{1-V} \tag{5}$$

We now analyze the case where $w \in \mathcal{W}$. Under Assumption 1, we have:

$$F(w_{k+1}) = F(w_k - \gamma AG(w_k)) \leq F(w_k) - \gamma\langle F(w_k), AG(w_k)\rangle + \gamma^2\frac{L}{2}\,\|AG(w_k)\|^2 \tag{6}$$

Using the fact that $\|F(w_k) - AG(w_k)\|^2 = \|F(w_k)\|^2 - 2\langle F(w_k), AG(w_k)\rangle + \|AG(w_k)\|^2$, (6) becomes

$$\begin{aligned} F(w_{k+1}) &\leq F(w_k) - \gamma(1-\gamma L)\langle F(w_k), AG(w_k)\rangle \\ &+\gamma^2\frac{L}{2}\left(\|AG(w_k) - F(w_k)\|^2 - \|F(w_k)\|^2\right). \end{aligned} \tag{7}$$

By taking the expectation $\mathbb{E}[\cdot]$ on both sides in (7), we obtain:

$$\begin{aligned} \mathbb{E}[F(w_{k+1})] &\leq F(w_k) - \gamma(1-\gamma L)\langle F(w_k), \mathbb{E}[AG(w_k)]\rangle \\ &+\gamma^2\frac{L}{2}\left(\mathbb{E}\left[\|AG(w_k) - F(w_k)\|^2\right] - \|F(w_k)\|^2\right) \end{aligned} \tag{8}$$

Using Lemma 1 and the fact that $\gamma < \frac{1}{L}$, we obtain:

$$\begin{aligned} \mathbb{E}[F(w_{k+1})] &\leq F(w_k) - \frac{1}{2}\gamma(1-\gamma L)((1-V)\|F(w_k)\|^2 - U) \\ &+\gamma^2\frac{L}{2}\left(\mathbb{E}\left[\|AG(w_k) - F(w_k)\|^2\right] - \|F(w_k)\|^2\right) \end{aligned} \tag{9}$$

Using Lemma 2 in (9), we also have:

$$\begin{aligned} \mathbb{E}[F(w_{k+1})] &\leq F(w_k) - \frac{1}{2}\gamma(1-\gamma L)\left((1-V)\|F(w_k)\|^2 - U\right) \\ &+\gamma^2\frac{L}{2}\left(\left(U + V\|F(w_k)\|^2\right) - \|F(w_k)\|^2\right) \end{aligned} \tag{10}$$

Rearranging the terms gives:

$$\mathbb{E}[F(w_{k+1})] \leq F(w_k) - \frac{\gamma}{2}(1-V)\|F(w_k)\|^2 + \frac{\gamma U}{2} \tag{11}$$

which is equivalent to:

$$\|\nabla F(w_k)\|^2 \leq \frac{2(F(w_k) - \mathbb{E}[F(w_{k+1})])}{\gamma(1-V)} + \frac{U}{1-V} \tag{12}$$

Let $\mathbb{E}$ denote the total expectation taken with respect to the joint distribution of all random variables from iteration 1 to T. We have then:

$$\mathbb{E}\|\nabla F(w_k)\|^2 \leq \frac{2(\mathbb{E}[F(w_k)] - \mathbb{E}[F(w_{k+1})])}{\gamma(1-V)} + \frac{U}{1-V} \tag{13}$$

By summing both sides through $[1, \cdots, T]$, we get:

$$\sum_{k=1}^{T}\mathbb{E}\|\nabla F(w_k)\|^2 \leq \frac{2(\mathbb{E}[F(w^{loc})] - \mathbb{E}[F(w_{T+1})])}{\gamma(1-V)} + T\frac{U}{1-V} \tag{14}$$

Dividing the last inequality by T gives:

$$\mathbb{E}\left[\frac{1}{T}\sum_{k=1}^{T}\|\nabla F(w_k)\|^2\right] \leq \frac{2(F(w^{loc}) - \mathbb{E}[F(w_{T+1})])}{T\gamma(1-V)} + \frac{U}{1-V} \tag{15}$$

And using the fact that $\mathbb{E}[F(w_{T+1})] > F(w_*)$, we obtain:

$$\mathbb{E}\left[\frac{1}{T}\sum_{k=1}^{T}\|\nabla F(w_k)\|^2\right] \leq \frac{2(F(w^{loc}) - F(w_*))}{T\gamma(1-V)} + \frac{U}{1-V}, \tag{16}$$

which also means that

$$\min_{k\in[T]}\mathbb{E}\left[\|\nabla F(w_k)\|^2\right] \leq \frac{2(F(w^{loc}) - F(w_*))}{T\gamma(1-V)} + \frac{U}{1-V}. \tag{17}$$

Finally, by combining (5) and (17), we obtain, $\forall w \in \mathbb{R}^d$:

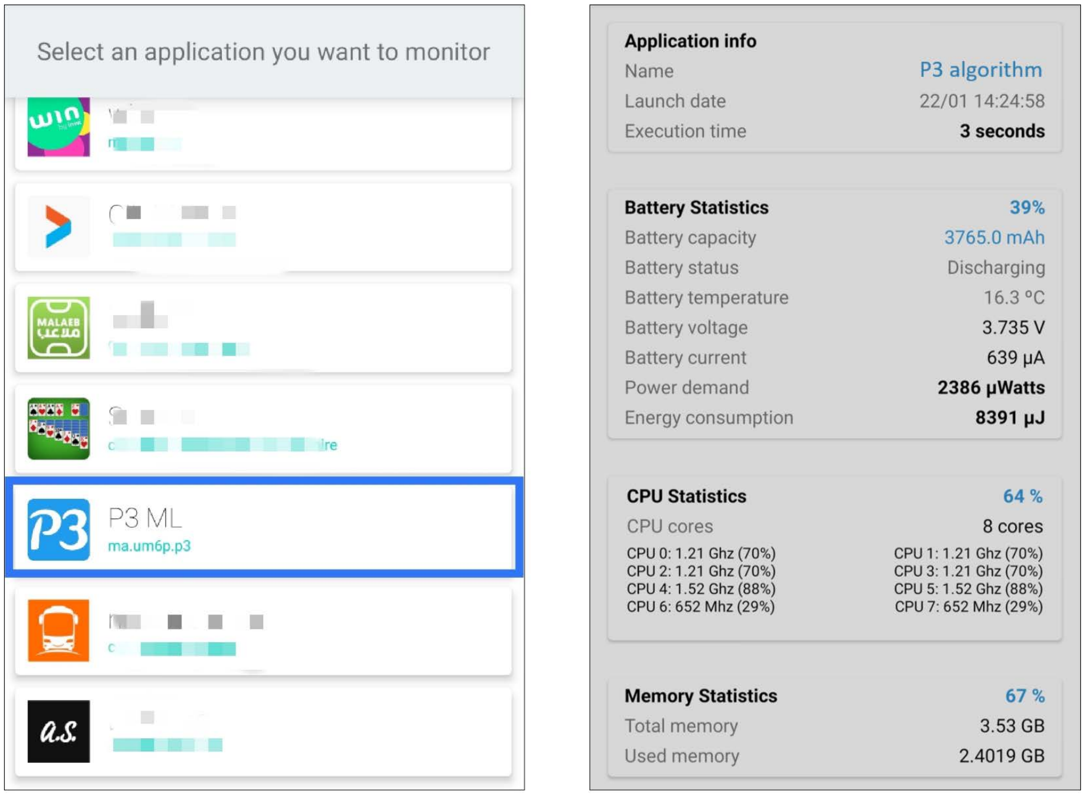
$$\begin{aligned} \min_{1<i<T}\mathbb{E}\left[\|\nabla F(w_i)\|^2\right] &\leq \max\left\{\frac{2(F(w^{loc}) - F(w_*))}{T\gamma(1-V)} + \frac{U}{1-V}, \frac{U}{1-V}\right\} \qquad (18) \\ &= \frac{2(F(w^{loc}) - F(w_*))}{T\gamma(1-V)} + \frac{U}{1-V} \qquad (19) \end{aligned}$$

which concludes the proof. □

B ADDITIONAL DETAILS ON EXPERIMENTS

B.1 Energy Monitoring Application

Figure 8 presents screenshots of the implemented Android-based monitoring system. In the left picture, we can select which application to monitor. Once an application is selected, the monitoring application switches to a new information screen, we can find the name of the monitored application (e.g., the P3 algorithm in our case), the start time and the execution time. In the battery statistics part, one can find information related to the voltage, current, power and energy consumption. Finally, the CPU and memory statistics part illustrates the overall utilization of the CPU and memory occupation, respectively. The source code of the Power Monitor is available on: https://anonymous.4open.science/r/PowerManager-7CB3/


Select an application you want to monitor

P3 ML
ma.um6p.p3

Application info	
Name	P3 algorithm
Launch date	22/01 14:24:58
Execution time	**3 seconds**
Battery Statistics	**39%**
Battery capacity	3765.0 mAh
Battery status	Discharging
Battery temperature	16.3 °C
Battery voltage	3.735 V
Battery current	639 μA
Power demand	**2386 μWatts**
Energy consumption	**8391 μJ**
CPU Statistics	**64 %**
CPU cores	8 cores
CPU 0: 1.21 Ghz (70%)	CPU 1: 1.21 Ghz (70%)
CPU 2: 1.21 Ghz (70%)	CPU 3: 1.21 Ghz (70%)
CPU 4: 1.52 Ghz (88%)	CPU 5: 1.52 Ghz (88%)
CPU 6: 652 Mhz (29%)	CPU 7: 652 Mhz (29%)
Memory Statistics	**67 %**
Total memory	3.53 GB
Used memory	2.4019 GB

Figure 8: The developed Android monitoring application.

B.2 Energy Consumption of FedAvg

In this experiment, we want to compare the energy consumption of our P3 algorithm with a federated learning (FL) scheme using the FedAvg algorithm. FL has a hybrid architecture, as it relies on workers (i.e., mobile devices in our case) to do the actual training of the models locally and a server parameter to aggregate the received updates.

For a given number of iteration, every worker trains the model on its local data for a given number of epochs. The higher the number of train epochs, the more energy required from the worker to finish the task. In our experiment, each worker trains only for one epoch per iteration.

We present in our experiment a lighter variant of FedAvg named e-FedAvg. Here, workers only draw random samples from the local data similar to the $broadcastUpdate()$ of P3 and train the received model for one epoch.

Figure 9 shows the results of our comparison between our P3 algorithm, both versions of the FL scheme (i.e., FedAvg and e-FedAvg) and the CL scheme. We considered settings with 10 peers using MNISTX32. We observed that P3 consumes X591 times less energy than FedAvg with one epoch. In contrast, e-FedAvg was much more efficient. Yet, P3 was X9 times more efficient than e-FedAvg.

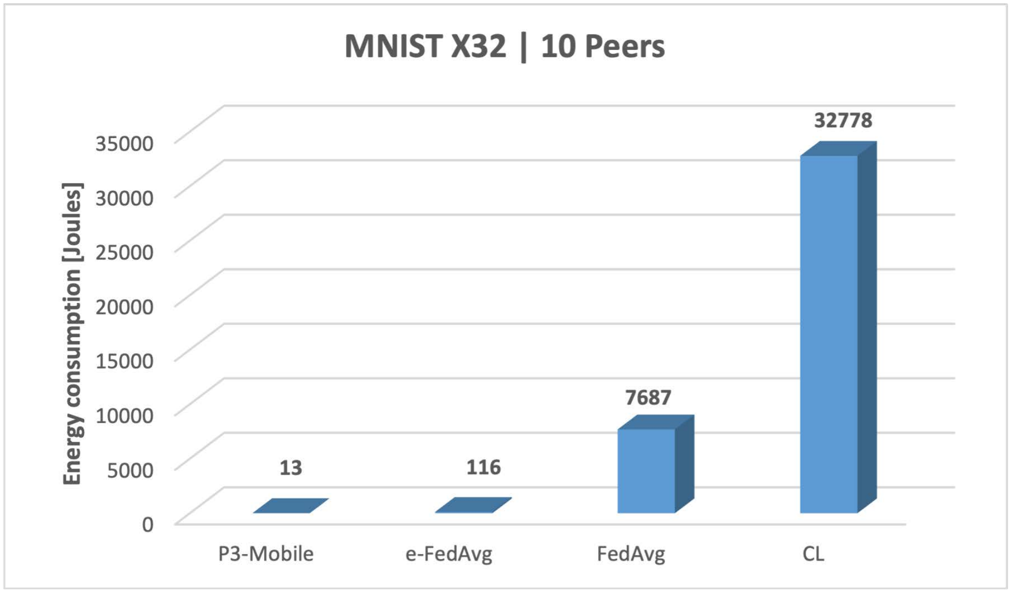


Figure 9: Energy consumption comparison between our P3 algorithm, FedAvg, e-FedAvg (lighter version of FedAvg) and CL. P3 and the workers in the federated learning (FL) scheme are running on mobile devices, while the aggregator for FL and CL are running on a server.