University of Novi Sad

Faculty of Sciences

Department of Mathematics and Informatics

Nevena Đilas

Green Federated Learning Training Strategy

Master's thesis

Mentor

Dr Dušan Jakovetić

Novi Sad, 2024

I dedicate this thesis to my little brother Vladan in heavens, with love.

I would like to express my sincere gratitude to my mentor Dr Dušan Jakovetić for his advice, help, guidance and support during the writing of this thesis. Additionally, I would like to thank members of the board for their valuable feedback and for reviewing this thesis.

I am grateful to my family for their unconditional support, patience, understanding and encouragement during my studies.

I would like to thank all my friends and colleagues for brightening my days and studying together.

Last, but not least, I would like to thank all the professors that selflessly shared their knowledge with me and help me get to where I am today.

Contents

1. Introduction

Machine learning models have grown larger and more complex over the years. Innovative new technologies and methods for training these models have emerged. Federated learning is a method for training large amount of decentralised data – data stored on client devices. It is the model training method that is gaining popularity and recognition in the recent years due to its many advantages, mostly for privacy benefits.

However, there are environmental concerns regarding federated learning and other computationally expensive deep learning methods because the consequence of increased amount of computing is the large carbon footprint. Since technological innovations result in an increase of greenhouse gases responsible for global warming, there is a need to add the carbon footprint as an evaluation criterion for machine learning models. Our goal is to quantify the carbon emissions of federated learning, compare them with emissions of centralised learning, explore factors that contribute to these emissions and find ways to optimise them so that we can minimise these emissions. In this thesis we propose a strategy for minimising carbon footprint of federated learning.

First, we learn about federated learning, its pipeline and settings. We'll see how FedAVG, the fundamental federated leaning algorithm, works and how its updates for the model differ from the centralised learning updates.

Secondly, we will find out about the impact of technology on climate change. We will give formulas to quatify the amount of greenhouse gases that federated learning an centralised learning emmit.

Finally, we investigate factors which affect the emissions of greenhouse gases, find ways to predict the carbon footprint of federated learning acording to these factors and propose a way to optimise them. We explore hyperparameter optimisation methods, some of them specificaly designed for federated learning (the FedEx method) in our search for a strategy for optimising model's learning. We present conclusions and guidelines for greener federated learning.

2. Federated Learning

As neural network models grew larger, so did the need for computational power used to train such large models. Today, deep learning for state-of-the-art models is mostly performed in large data centres. Several problems emerged with this model training method, like the cost of maintenance of servers and the fact that the data from the users of the model was gathered and stored on these servers putting the users' data privacy in question. Therefore, decentralised learning methods were introduced.

Federated learning (FL) [6,8,9] is a decentralised learning method for training deep learning models. It is conducted by multiple devices that collaboratively learn a machine learning model without sharing their private data under the supervision of a central server. The model is trained on a large number of client devices like smartphones (if the model is for personal use) or in case of institutions, like private hospitals, that cannot share the data among each other, the model is trained in each institution on their data and then sent to the central server. The central server then aggregates the local models sent from clients and updates the global model. In later learning rounds, the central server sends a global model with training tasks back to the clients.

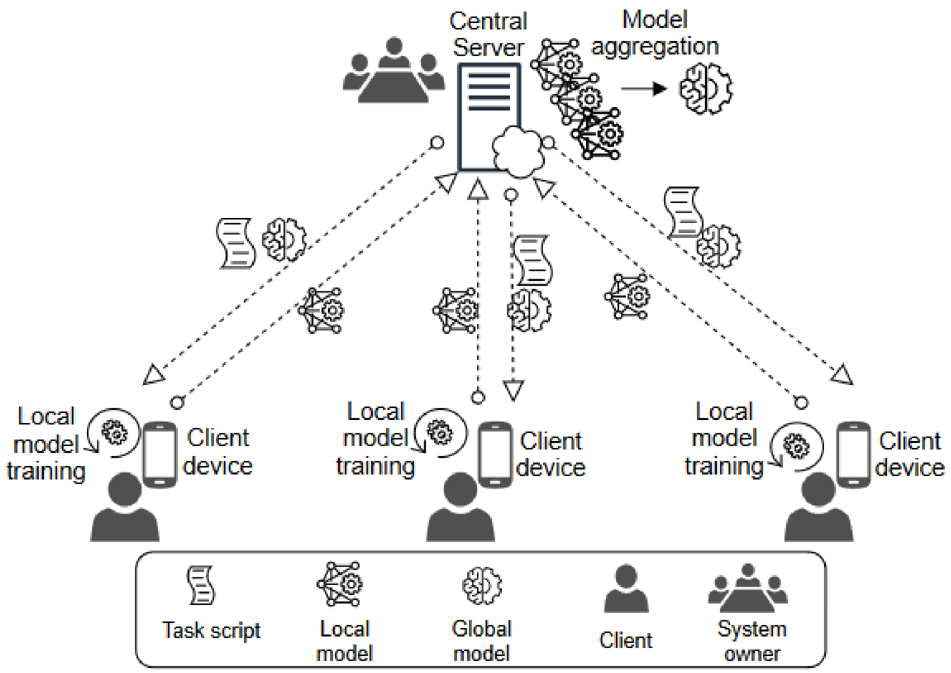


Figure 1: Overview of Federated Learning Process [8]

For federated learning there are two types of hardware components [8] (system nodes) needed: central server and client device. There are three stakeholders in a federated learning system: learning coordinator (owner of the system), contributor client (contributes data and trains a local model) and user client (uses the model). We can see that a contributor client can also be a user client.

In centralised deep learning [6,17], the objective of training a dataset containing n samples $(x_i, y_i), 1 \leq i \leq n$ is:

$$\min_{w \in \mathbb{R}^d} f(w)$$

$$\text{where } f(w) \overset{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} f_i(w),\ w \text{ – weight}$$

Here, $f_i(w) = l(x_i, y_i, w)$ is the loss on the prediction on sample (x_i, y_i). A typical iterative methods for solving problems like this are gradient descent or stochastic gradient descent (SGD). In each new iteration weight is updated as follows:

$$w_{t+1} = w_t - \eta \Delta f(w_t) \text{ – gradient descent}$$

$$w_{t+1} = w_t - \eta \Delta f(w_t; x_k, y_k) \text{ – stochastic gradient descent}$$

where η is a learning rate that controls the step size and (x_k, y_k) is a randomly chosen small subset (mini-batch) of training samples that SGD uses at each step to compute a gradient, instead of computing gradient for all training samples.

Federated learning has the same objective, but it is decentralised [6,17]. Suppose that n training samples are distributed to K clients, where P_k is the set of indices of data points on client k, and $n_k = |P_k|$. The training objective can be written as:

$$\min_{w \in \mathbb{R}^d} f(w)$$

$$\text{where } f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \text{ and } F_k(w) \overset{\text{def}}{=} \frac{1}{n_k} \sum_{i \in P_k} f_i(w).$$

We see that if P_k is a partition that was formed by distributing the training examples over the clients uniformly at random, the expectation of $F_k(w)$ over the set of examples assigned to a fixed client k is the objective function $f(w)$; $E_{P_k}[F_k(w)] = f(w)$. This is the identically and independently distributed (IID) case that is an assumption typically made by distributed optimisation algorithms. However, this is not always the case. Function F_k could be an arbitrarily bad approximation of f. This happens in non-IID settings that are very common in practice.

Federated learning settings [1] are *cross-silo* and *cross-device* settings. These settings refer to the nature of the data distribution and the characteristics of the participating clients. In *cross-silo* settings the data is IID. Data distribution of each client is the same as global data distribution. Participating clients of a *cross-silo* federated learning scenario are typically organisations or entities that have a collaborative interest in training a shared machine learning model, for example private hospitals. This case where the participants are organisations is ilustrated in Figure 2. *Cross-device* settings involve collaboration among individual client devices, each contributing its local data and that data is non-IID. These devices are typicaly smartphones, but they can be personal laptops, tablets or desktop computers as ilustrated in Figure 3.

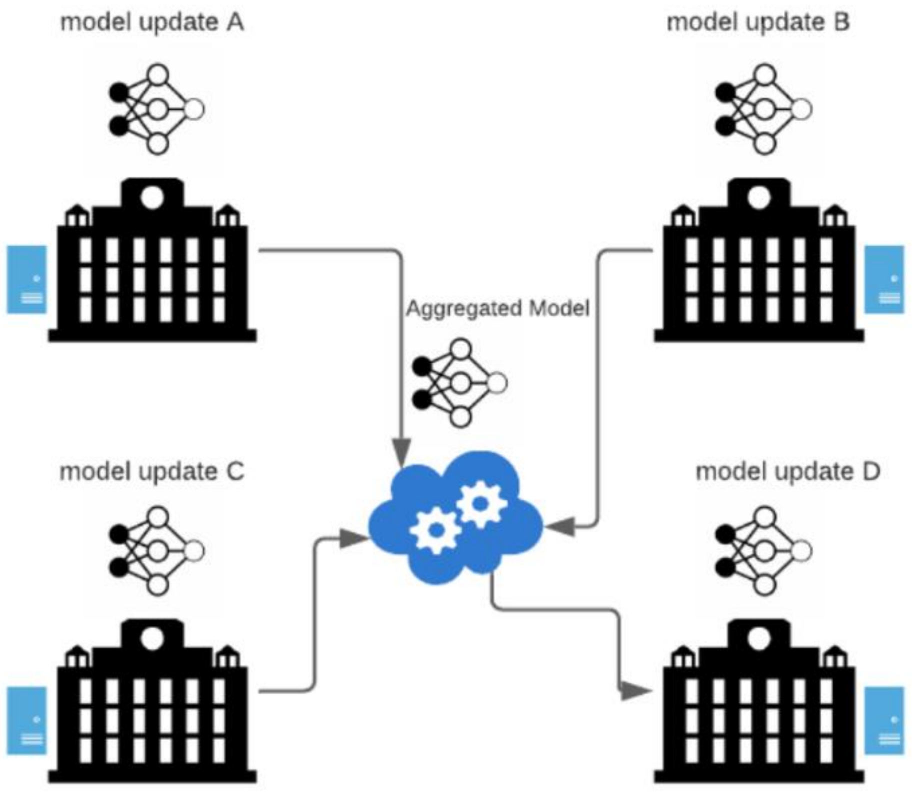


Figure 2: Federated Learning Across Organisations [8]

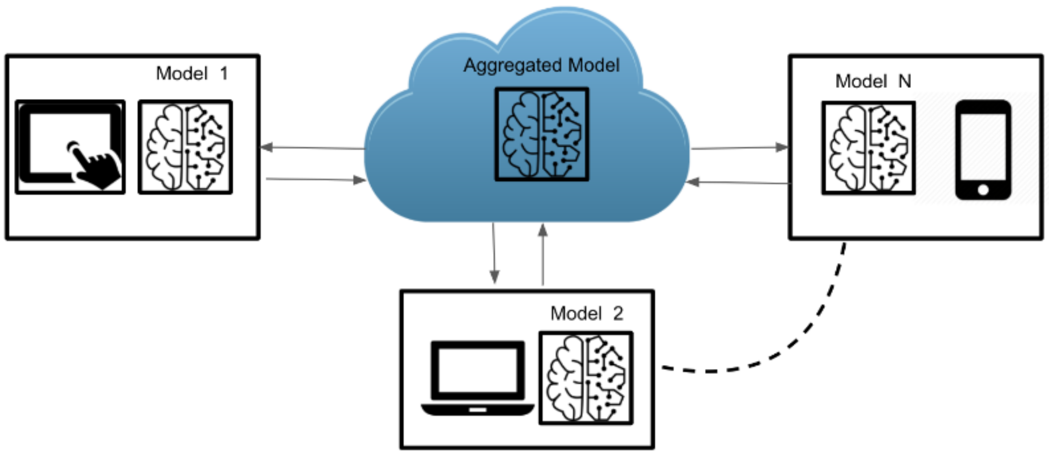


Figure 3: Federated Learning Across Devices [8]

2.1 Synchronous and Asynchronous Federated Learning

There exist *synchronous* and *asynchronous* federated learning [7].

Synchronous FL [15,7], the classical federated learning mostly found in research papers, consists of rounds. At the beginning of a round, a subset of existing clients is selected, each of which downloads the current model. Then, each client in this subset computes an updated model based on their local data. The model updates from the contributor clients are then sent to the server and the server aggregates these models (most commonly by averaging) to construct an improved global model.

Standard synchronous FL can be described as follows: At the initialisation step server prepares the initial global model w_G^0, alongside training parameters like learning rate, batch size, and round (iteration) count. In round $t \geq 0$, the server distributes w_G^t, the current global model, to a subset K of contributor clients. Contributor clients independently update the model using their respective local data. We denote the updated local models $w_1^t, w_2^t, \dots, w_k^t$, $k \in [1, K]$. These local models, w_k^t, are subsequently transmitted back to the server. Upon gathering the client updates, central server computes the global update by aggregating them, computing a weighted average of the local models: $w_G^{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_k^t$, where the training samples on node k amount to n_k and the total of training samples is represented as n.

In practice, not all clients send back the update due to the bad internet connection or similar problems. In those cases, a benchmark $K_{min} < K$ of minimal number of client updates received for updating the model needs to be introduced . Also, because of the connection issues and availability of client devices there are ways to compress the updates so that they would be easier to upload and download. For that reason there exist two general classes of approaches, structured updates and sketched updates. More about them can be found in [broj].

Asynchronous FL [7] is a variant of federated learning where contributor clients can independently and asynchronously update the global model without waiting for a predefined synchronisation point.

The principal steps of asynchronous federated learning are outlined as follows: Similarly to synchronous FL, at the initialisation the initial global model w_G^0 is distributed over all the selected K clients. Clients then update their local model by using their local data. The moment the server receives the first updated local model, it updates the global model: $w_G^1 = \frac{w_G^0 + w_1^1}{2}$. This process repeats. The clients update their local model based on the most recent global model they received. Because of the heterogeneity in computing capabilities among client devices, the completion of local model training ($\{w_1^t, w_2^{t+1}, w_3^{t+2}, \dots, w_k^{t+k}\}$) ,where $t \geq 0$ is the iteration, does not occur concurrently. Central server aggregates the newly collected local model with the latest global model using the following equation: $w_G^{t+k} = \frac{w_G^{t+k-1} + w_k^{t+k}}{2}$.

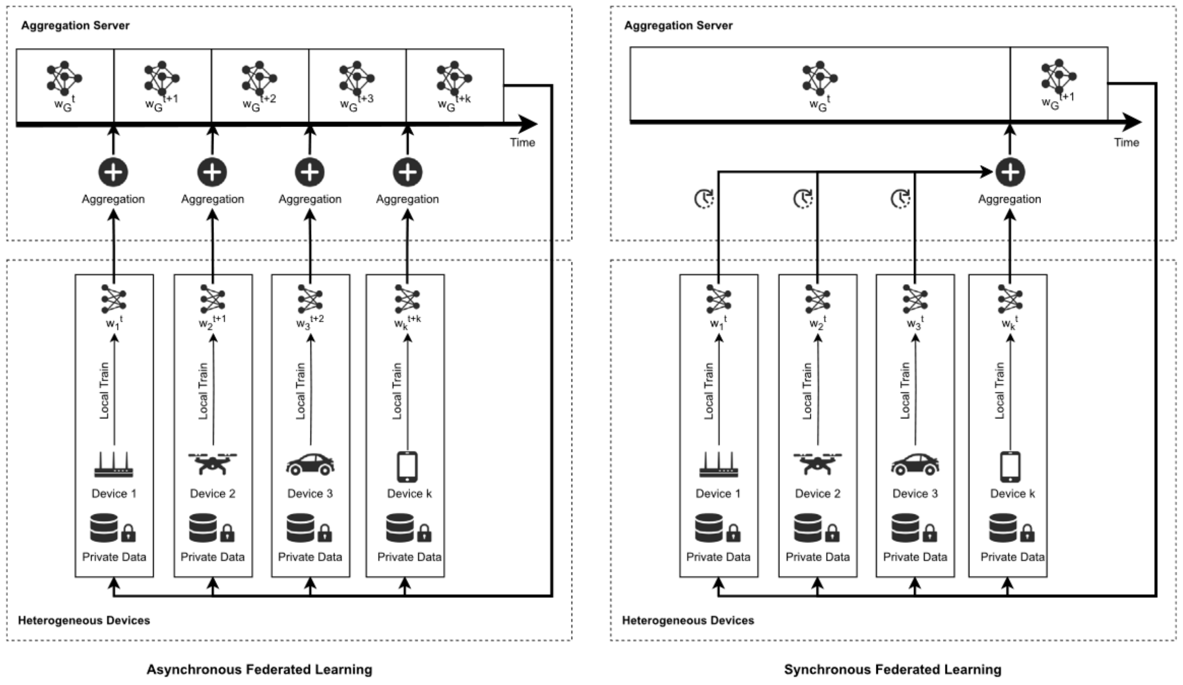


Figure 4: Representation of Training Process for Asynchronous and Synchronous FL [7]

2.2 Fundamental Federated Learning Algorithms

Two fundamental federated learning algorithms (approaches) are FedSGD and FedAVG [6]. They are made for classical (synchronous) federated learning. Many initial successful applications of deep learning have relied on variants of stochastic gradient descent (SGD) for optimisation. Later on, in 2016, Google has introduced FedAVG that became the leading basic algorithm for federated learning.

Let's denote with C fraction of clients that perform computation in each round. C controls the global batch size. For learning rate η, total number of samples n, total number of clients K, number of samples on a client k: n_k in a round t the central server is broadcasting current model w_t to each client and each client k is computing gradient on its local data $g_k = \nabla F_k(w_k)$.

For Federated SGD (FedSGD) we take $C = 1$ that corresponds to full-batch (non-stochastic) gradient descent. First approach would be that each client submits its local gradient g_k to the central server. Then the server aggregates the gradients to generate a new model:

$$w_{t+1} \leftarrow w_t - \eta \nabla f(w_t) = w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k$$

Second approach is equivalent to the first, but the difference is that each client updates its local model $w_{t+1}^{k} \leftarrow w_t - \eta g_k$ and then sends the updated local model to the server. Central server performs the aggregation of these models:

$$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^{k}$$

Federated Averaging (FedAVG) is based on adding more computations on top of FedSGD. In FedAVG the amount of computation is controlled with these three parameters:

- C, fraction of selected clients
- E, number of training passes each client makes over its local dataset on each round (epochs)
- B, local minibatch size used for client updates

In each round t every client k that was selected, $k \epsilon C$, after getting the global model and computing the local gradient g_k computes for E epochs: $w_{t+1} \leftarrow w_t - \eta g_k$. Number of local updates per round for a client with nk local examples is given by: $u_k = E \frac{n_k}{B}$.

For $C = 1$, $E = 1$, $B = \infty$ (the full local dataset is treated as a minibatch) we get FedSGD.

Algorithm of FedAVG is given by this pseudo-code:

Algorithm 1 [6]: FedAVG

The K clients are indexed by k, $\mathcal{P}_k$ is the set of indexes of data points on client k, B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

```
Server executes:
initialise $w_0$
for each round $t = 1,2,\ldots$ do
    $m \leftarrow \max(C \cdot K, 1)$
    $S_t \leftarrow$ (random set of $m$ clients)
    for each client $k \in S_t$ in parallel do
        $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
    $m_t \leftarrow \sum_{k \in S_t} n_k$
    $w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$
ClientUpdate$(k, w)$:
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size $B$)
for each local epoch $i$ from $1$ to $E$ do
    for batch $b \in \mathcal{B}$ do
        $w \leftarrow w - \eta \nabla l(w; b)$
return $w$ to the server
```

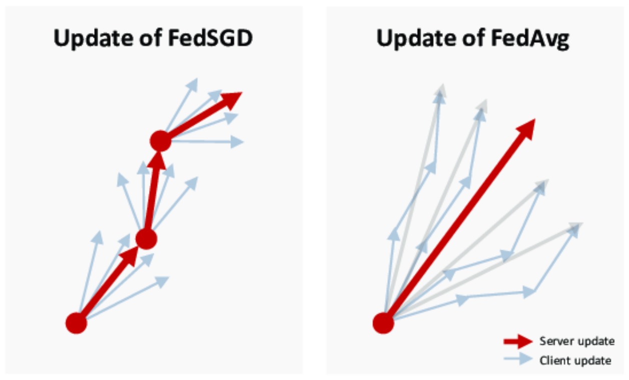


Figure 5: Updates of FedSGD and FedAVG [33]

Figure 5 illustrates the differences between FedSGD and FedAVG updates [33]. Gray arrows represent each clients' gradient and the red arrows represent a global model update on the central server in one communication round. In FedSGD, after each client performs one step of SGD, it sends the update to the server, while FedAvg allows each client to perform multiple SGD steps before averaging.

Federated Averaging is considered as basic and most used algorithm for federated learning nowadays. It is simple and eficient. However, many more algorithms for federated learning besides FedSGD and FedAVG have been developed and even more will follow as FL becomes a leading new architecture.

Federated learning, because of its privacy benefits and the fact that it is using computational power of smaller devices for training, is gaining popularity and is expected to be a leading technology in the future. Due to different countries' data regulations, for example European General Data Protection Regulation (GDPR) from 2020, moving data across national borders becomes subject to data sovereignty law [1], centralised data training becomes infeasible and federated learning is the reasonable solution for such a problem.

3. Green Federated Learning

We have seen the benefits of federated learning, how it proposes an innovative approach for analysing and learning from the data spread across thousands of devices and protects privacy of data owners, the users. However, when reviewing an architecture, it's not only privacy concerns, model precision, computational cost that matters, but also the effect that it has on the environment. We need to be aware of the impact that training state-of-the-art models has on climate change and try to minimise the said negative impact in order to protect our planet from global warming.

Climate change is one of the biggest pressing global issues humanity is facing today. It has a significant impact on human communities, ecosystems and biodiversity. Climate change is a long-term shift in temperatures and weather patterns. Historically, natural changes in the sun's activity and large volcanic eruptions have caused shifts in the Earth's temperatures and weather patterns.[1] However, over the past 200 years, these natural factors haven't considerably influenced global temperatures. Today, climate change is primarily driven by human activities. It's caused mostly by harmful greenhouse gases emitted by burning fossil fuels such as coal, oil, and gas. These gases are also increased in the atmosphere because of agriculture and cutting down forests [31].

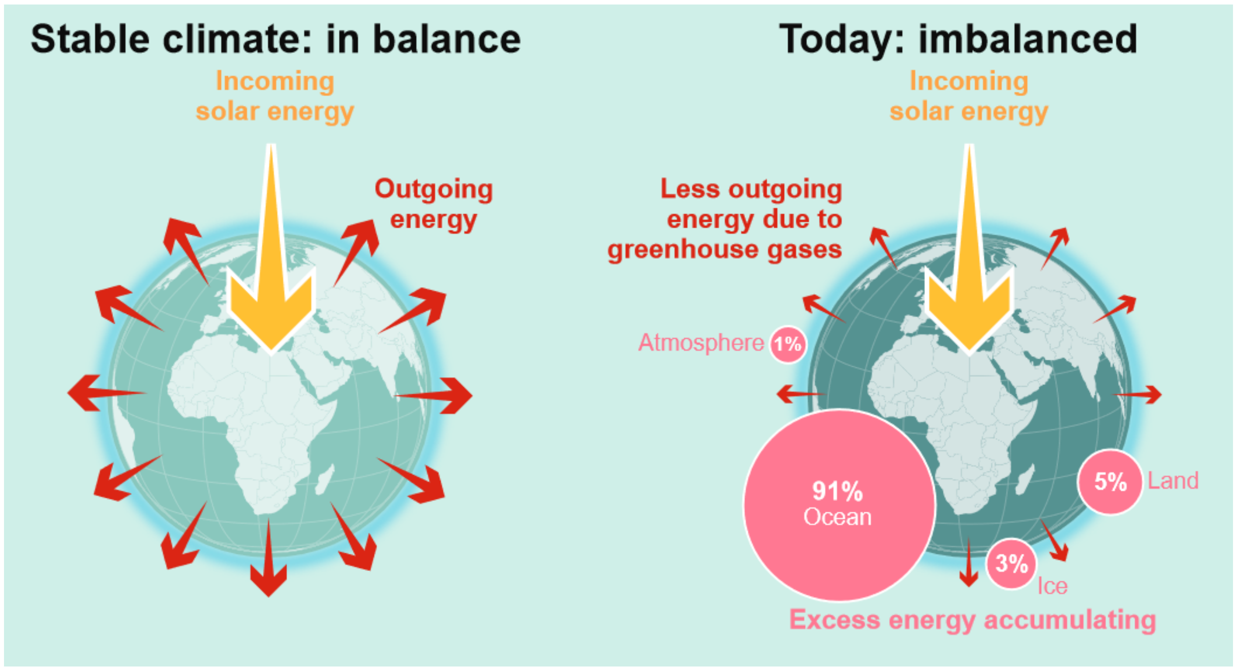
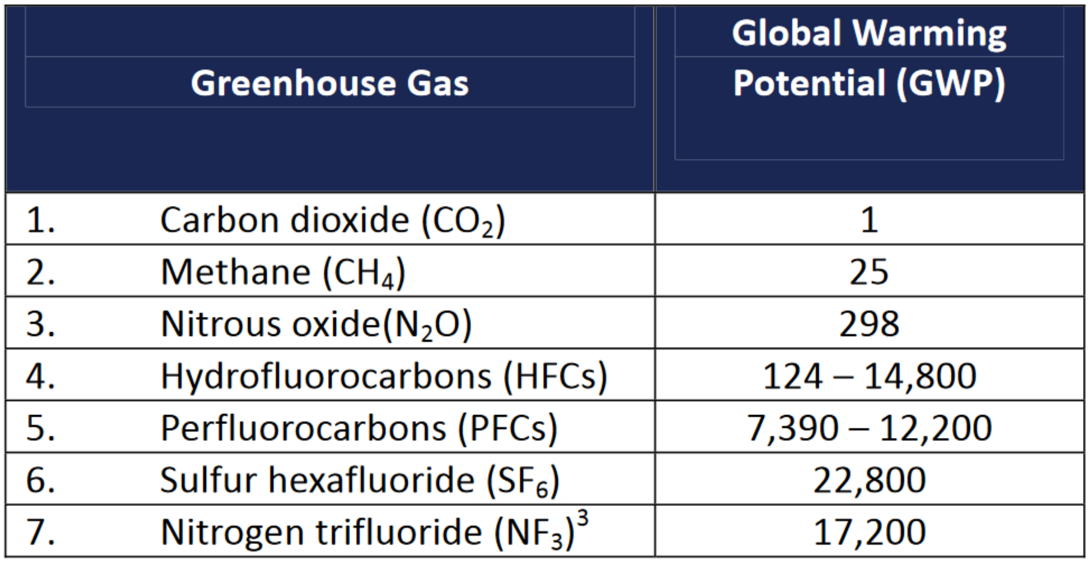


Figure 6: Climate Change Impact on Earth's Energy Balance [31]

[1] https://www.un.org/en/climatechange/science/mythbusters

Greenhouse gases are gases like carbon dioxide, methane and nitrous oxide that reside in Earth's atmosphere. They absorb and re-emit heat, warming the Earth by making it more difficult for heat to be released into outer space, as shown in Figure 3. Their increase is causing global warming. Concentrations of greenhouse gases in the atmosphere today are at levels not seen in the last 800000 years [1]. ICCC, Intergovernmental Panel on Climate Change, has established an international treaty for controlling the release of greenhouse gases from human activities – Kyoto Protocol [27]. The gases controlled under the treaty are shown in Figure 7.



Greenhouse Gas		Global Warming Potential (GWP)
1.	Carbon dioxide (CO_2)	1
2.	Methane (CH_4)	25
3.	Nitrous oxide(N_2O)	298
4.	Hydrofluorocarbons (HFCs)	124 – 14,800
5.	Perfluorocarbons (PFCs)	7,390 – 12,200
6.	Sulfur hexafluoride (SF_6)	22,800
7.	Nitrogen trifluoride (NF_3)[3]	17,200

Figure 7: Greenhouse Gases from Kyoto Protocol [27]

Since there are different greenhouse gases, their potential for warming the Earth's surface is different. Some of them last in the atmosphere for a longer period of time than the others and some absorb more heat than the others. Also, they don't exist in the same quantities in the atmosphere. Global Warming Potential (GWP) of greenhouse gases indicates the amount of warming a gas causes over a period of 100 years. Carbon dioxide, the most common and well known greenhouse gas, has a GWP of 1. The GWP for all other gases is the number of times bigger for the warming they cause compared to CO_2.

Since there is a need to address and all these gases in the same unit of measurement, we introduce CO_2e. Carbon dioxide equivalent, CO_2e, is a common unit that for any quantity and type of greenhouse gas signifies the amount of CO_2 which would have the equivalent global warming impact. For example, if 1kg of methane is emitted, this can be expressed as 25kg of CO_2e. This follows from the fact that GWP of methane is 25 (from table in Figure 6) and then we calculate the carbon dioxide equivalent as: $1kg\ \mathrm{CH_4}\ * 25 = 25kg\ \mathrm{CO_2e}$

CO_2e is a very useful term because it allows “bundles” of greenhouse gases to be expressed as a single number and therefore different bundles of gases can be easily compared. This is the unit in which we will express all greenhouse emissions moving forward.

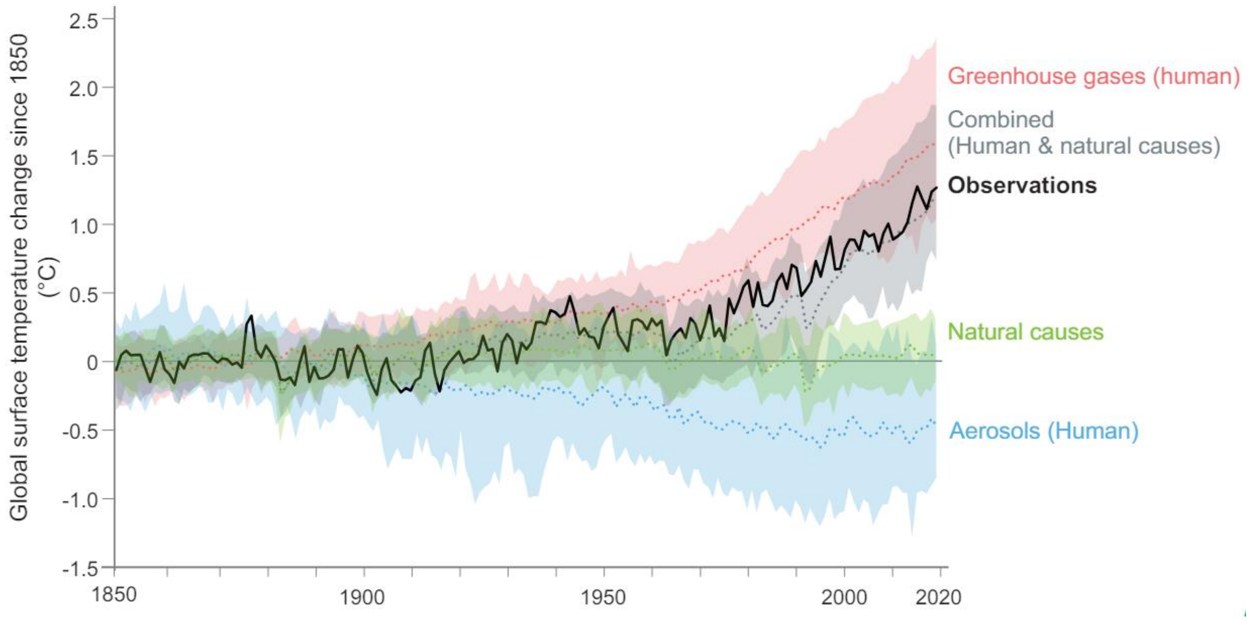


Figure 8: Different Causes of Climate Change with Respect to Temperature [31]

In Figure 8 we can observe the causes of climate change with respect to change of temperature from year 1850 to 2019. Colour shadings in the graph represent climate model simulations that can only reproduce observed change in global temperature (black) when they include human-caused emissions. This figure illustrates how global temperatures change using climate model simulations that incorporate: only greenhouse gases (red band), only aerosols (air pollutants) and other human drivers (blue band), only natural causes (green band) and the case when all causes are included (grey band). It is very clear from the graph that humans are the ones responsible for warming the climate.

The largest source of greenhouse gases, besides transportation and industrial activity, is electricity production because of emissions of power plants. Of course, electricity production can be made greener by using renewable energy sources like wind power, solar power, bioenergy (organic matter burned as a fuel) and hydroelectric energy. However, most electricity is to this day produced in power plants that burn fossil fuels and produce harmful gases. That is why it is of essential importance to minimise the carbon footprint of complex machine learning models whose training requires large amounts of electrical power.

Deep learning algorithms keep growing in complexity requiring a large amount of computational resources and energy. Today’s state-of-the-art models are trained on large amounts of data using a significant amount of electric energy to train the model. The data is processed and stored in extensive data centres that currently account for about 1% – 1.5% of global electricity use, according to the International Energy Agency[2]. Also, the amount of computing needed for training biggest deep neural networks has been exponentially increasing and it has grown by more than 300000 times from 2012 to 2018 [1]. Data centres use around 200TWh (terawatt-hours) per year and that is more electricity consumption than some countries make in a year. Nowadays even the smaller deep learning models that are not trained in data centres for tasks like making a speech recogniser that can be found in research papers could produce more than 0.1 tonnes of CO_2e with the consumer-grade hardware. To further emphasize the magnitude of these emissions, an average human being is responsible for 5 tonnes of CO_2e per year on average, whereas training an NLP (Natural Language Search) model may produce 284 tonnes of carbon emissions. The pollution from production of numerous new deep learning models is increasing and this is an issue that needs to be addressed.

Trend of "the bigger, the better" doesn't tend to stop when it comes to creating new AI. There is a race between big companies who want to produce the most innovative new technology and sometimes the solution becomes making data centres larger, so they can store more data, and make the models more complex. The bigger the data centre, the more significant impact it has on the environment. The largest problem in data centres is cooling. Besides the electricity, cooling the hardware in a datacentre may require water and air. There has already been research about reusing the waste heat produced in data centres for district/plant/water heating, absorption cooling, direct power generation etc. [32] while cooling the data centre in the process. Cooling takes around 40% of the total consumption of energy for model training in a centralised setting.

In this competition for the best performing models of modern age, accuracy is one of the most important metrics that evaluates the model. However, it is also important to think about the environment and climate change while training state-of-the-art models. Therefore, energy efficiency and carbon footprint are proposed as evaluation criteria for research along-side accuracy, convergence, speed and related measures [2]. Optimising the efficiency of the model also lowers the hardware requirements which leads to cost reduction and enables the academics (not just big corporations) to develop innovative deep learning models.

Big corporations tend to deal with climate change by purchasing carbon offsets [5] to cover the impact on the environment carbon emissions that they produced made. However, doing this is not mandatory. Microsoft Azure21 and Google cloud20 (cloud computing platforms) purchased carbon credits to offset the electrical energy they spent, but Amazon's AWS22 covered only 50% of its power usage with renewable energy. It is still not proven that buying credits is as effective at preventing climate change as using less energy.

[2] https://www.techtarget.com/searchdatacenter/feature/Assess-the-environmental-impact-of-data-centers

Federated learning pipeline, as opposed to centralised learning that is conducted in data centres, consists of more elements – central server, internet network and client's end-devices. It is a decentralised learning method for training the data scattered across hundreds, thousands or even millions of devices. It operates differently than centralised learning architectures whose data is in one place and its energy consumption is harder to compute. Its CO_2-equivalent (CO_2e) emissions result from both hardware training and communication between server and clients. Since client devices vary and can be from many different countries that have different energy sources which may be more or less green, it is extremely hard to approximately deduce each client device's carbon footprint. It is important to notice that, since FL isn't centralised, this architecture does not have cooling problems because its data isn't in a data centre that needs to be cooled down. For this same reason, the money cost for training a model in a federated setting is significantly lower than in a centralised setting.

There have been several novel research papers that are contributing to the greener federated learning. They are adopting a data-driven approach to quantify the carbon emissions of federated learning by measuring FL tasks running on millions of phones. After the experiments that were conducted, the authors have presented the challenges, guidelines and lessons that were learnt from studying the trade-off between energy efficiency, performance and time needed to train FL models.

3.1 Estimation of CO_2e Emissions

An important finding is that the computations on client devices and communication between clients and the server are responsible for the majority of carbon emissions of federated learning (~97%) [2]. Server-side computation emissions are almost negligible (~1-2%). Clients' computation contributes to almost half of the CO_2e emitted (~46-50%). Upload contributes to around 27-29% and download to 22-24% of FL's emissions.

In the research paper "A First Look into the Carbon Footprint of Federated Learning" [1], authors have proposed one of the first models to quantify the carbon footprint of federated and centralised learning. They give us formulas to compute the electric energy consumed by the learning model and then convert that consumed energy into CO_2e emissions based on the geographical location of the hardware.

In federated learning, the total training energy consumption of N clients in the pool with hardware power e, for R rounds is defined by:

$$T_{FL}(e, N, R) = \sum_{j=1}^{R} \sum_{i=1}^{N} 1_{\{Clt_{i,j}\}} t_i e_{clt,i}$$

where $1_{\{Clt_{i,j}\}}$is the indicator function that indicates if participating client i is chosen for training at round j, $e_{clt,i}$ is energy power of one client i (coming from device's CPU and GPU), t_i is time per round.

Estimated energy required for transfering model parameters between the server and clients can be partitioned in two parts – energy consumed by routers and energy consumed by hardware when downloading and uploading the model parameters. Energy power of the router is reported on "The Power Consumption Database". Download and upload speed can be measured with Speedtest. The communication energy per round can be defined as:

$$C(e, N, R) = \sum_{j=1}^{R} \sum_{i=1}^{N} 1_{\{Clt_{i,j}\}} S \left(\frac{1}{D} + \frac{1}{U}\right) (e_r + e_{idle,i})$$

Here, S is the size of the model in Mb, e_r is the power of the router, $e_{idle,i}$ is the power of the hardware of the idile clients during upload and download, D and U are download and upload speed.

The total consumed energy of federated learning is a sum of $T_{FL}(e, N, R)$ and $C(e, N, R)$.

Total training energy in centralised setting can be calculated like this:

$$T_{centre} = \mathrm{PUE}(te_{centre})$$

Where t is total training time, e_{centre} is the power of CPUs and GPUs in the centralised training setup and PUE stands for Power Usage Effectiveness ratio. Power Usage Effectiveness ratio varies significantly depending on the company owning the data centre being used to conduct model training. For example, Google's PUE ratio is 1.11, Amazon's is 1.2 and Microsoft's is 1.125. PUE shows data centre's efficiency.

Next, the electricity-specific CO_2e emission factors of countries in which the devices (or data centre in case of centralised training) used to train the model are stationed are obtained from official governmental websites and reports. This emission factor is called the conversion rate factor, denoted c_{rate} and it is expressed in kg/kWh.

Conversion rate factor is lower in countries with carbon-efficient electricity production. For example, France is relying mostly on efficient power plants and France's conversion rate is 0.054. That is not the case with Australia, per se, with a conversion rate of 0.656. Here is the heat world map of electricity to CO_2e conversion rate:

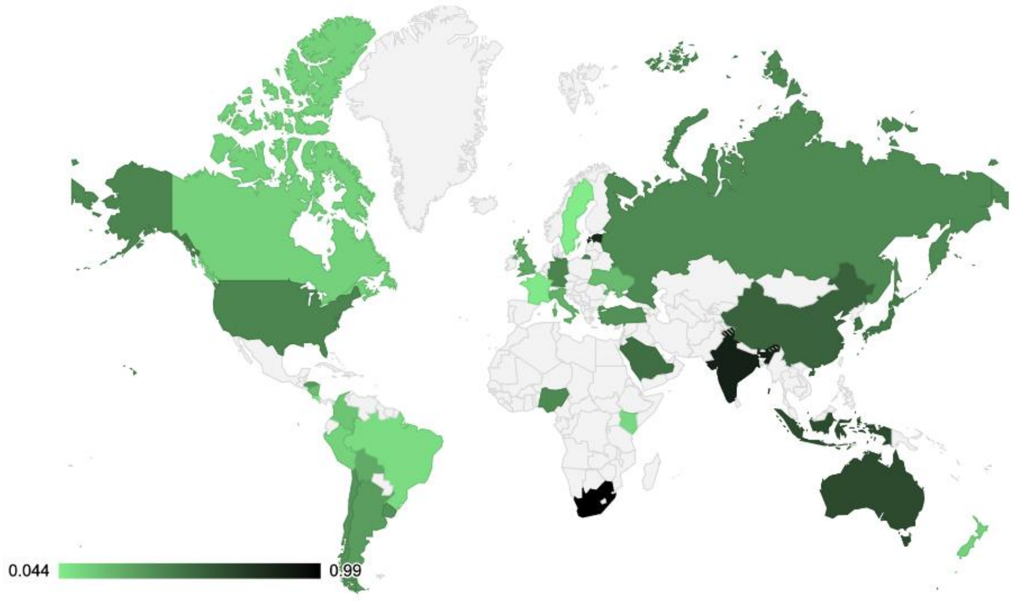


Figure 9: World Heat Map of Electricity to CO$_2$e Conversion Rate (in kg/kWh) [1]

The total amount of carbon gases emitted in kg for federated learning (E_{FL}) and for centralised learning (E_{centre}) is:

$$E_{FL} = c_{rate}\left(T_{FL}(e, N, R) + C(e, N, R)\right)$$

$$E_{centre} = c_{rate}T_{centre}$$

Federated learning can take place on devices from different countries. Therefore, when FL is being conducted on devices from around the world, c_{rate} needs to be adjusted for each client based on the client's physical location.

3.2 Experimental Results of Elected Studies Regarding Carbon Footprint of Federated Learning

The unfortunate results for this cost reducing and privacy boosting learning strategy are that FL can emit up to two orders of magnitude more carbon than centralised training. These are the results of performing federated and centralised learning in a controlled environment on the same datasets using the same neural network architectures. Only in certain settings can FL's carbon footprint be comparable to centralised learning.

It is also important to address that in a non-IID setting of federated learning the total energy consumption will commonly be bigger than in IID case. In non-IID, cross-devices cases, the FL algorithm often needs a larger number of communication rounds to reach sufficient model performance, using more electricity in the process. In IID, cross-silo case data distribution in each client is the same as the global data distribution, hence training energy is close to energy that emits centralised training with additional communication cost. This is illustrated in Figure 10 that shows the CO2e emissions of the same federated learning strategy for 5 local epochs used on image recognition datasets CIFAR10 and ImageNet in cases of IID and non-IID distributed data [1]. The carbon emissions of non-IID data of CIFAR10 are significantly higher than in IID case with the same data. Greenhouse gases emissions of non-IID ImageNet data are a bit higher than in IID case, but the difference is negligible.

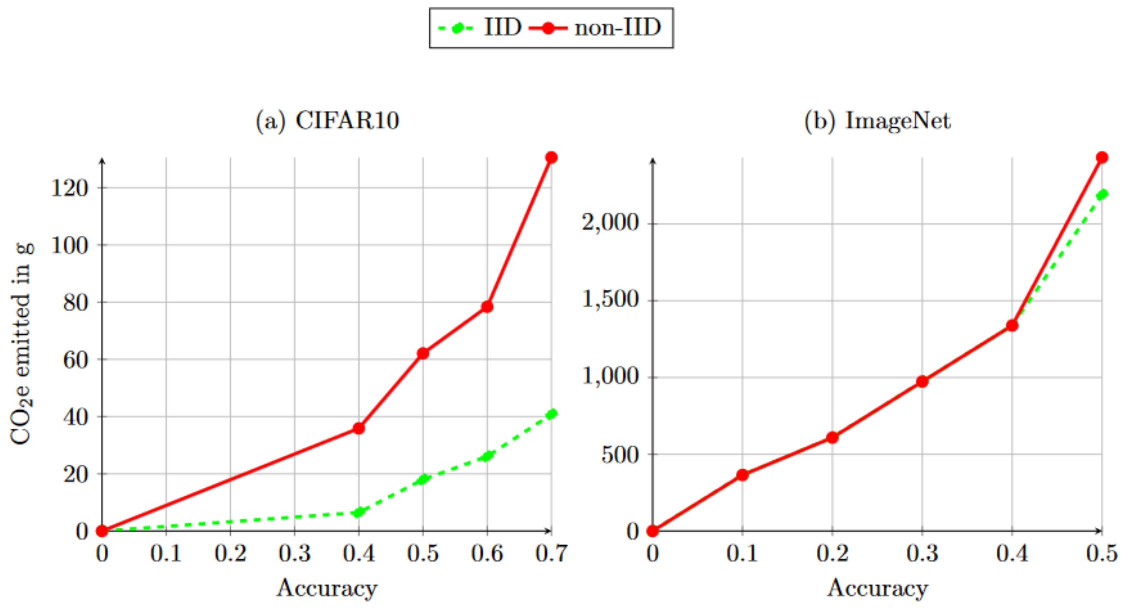


Figure 10: CO2e Emissions for CIFAR10 and ImageNet with Respects to Accuracies [1]

There is also an important relationship between CO2e emissions and accuracy shown in Figure 10. As accuracy increases, so do the harmful emissions. We see that the carbon emissions for additional accuracy gains is increasing exponentially, which is concerning since optimising accuracy is one of main goals of model training.

Important factor for estimation of the carbon footprint of FL is also hardware efficiency. Nowadays, the average client device doesn't have specialised hardware. However, as technology is improving, we can soon expect that smartphones, tablets and other devices might get hardware updates and be able to perform better in machine learning training. Federated learning can benefit more from hardware advancement then centralised learning because even though GPUs are becoming more energy-efficient (more computation can be done for less consumed energy then before), the need for energy-consuming cooling remains.

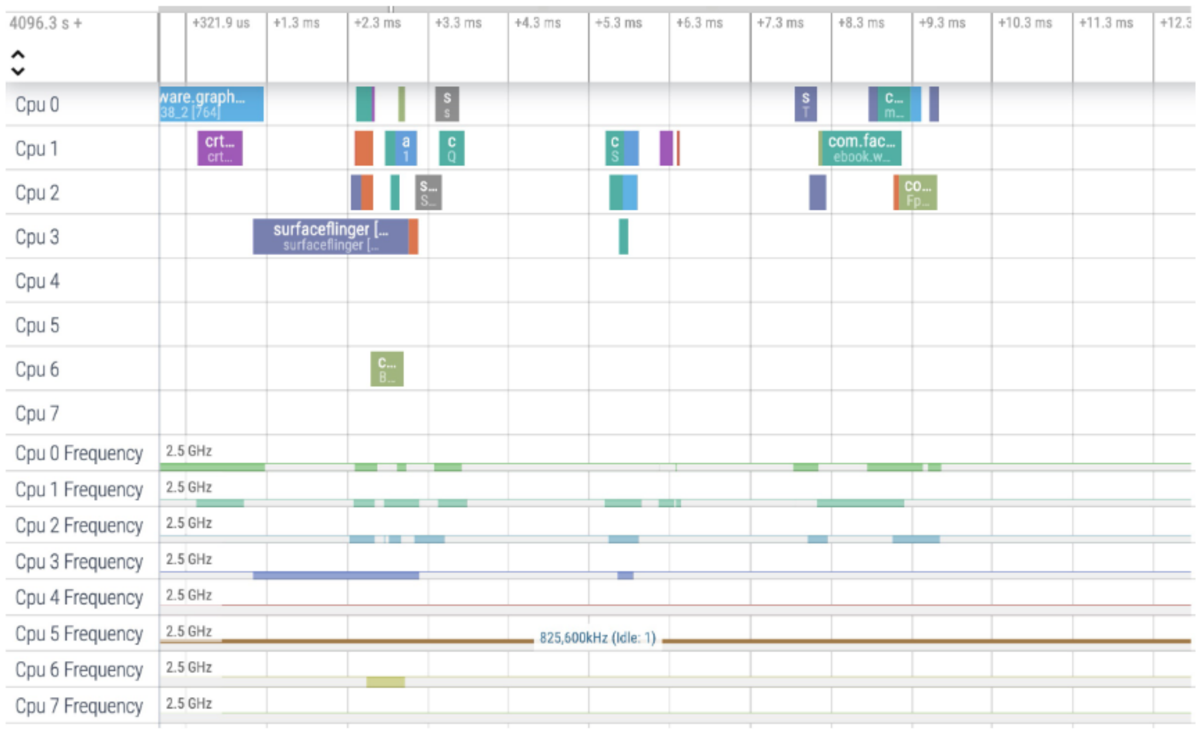


Figure 11: A Snapshot of the 8 Cores of the Phone in Idle State [2]

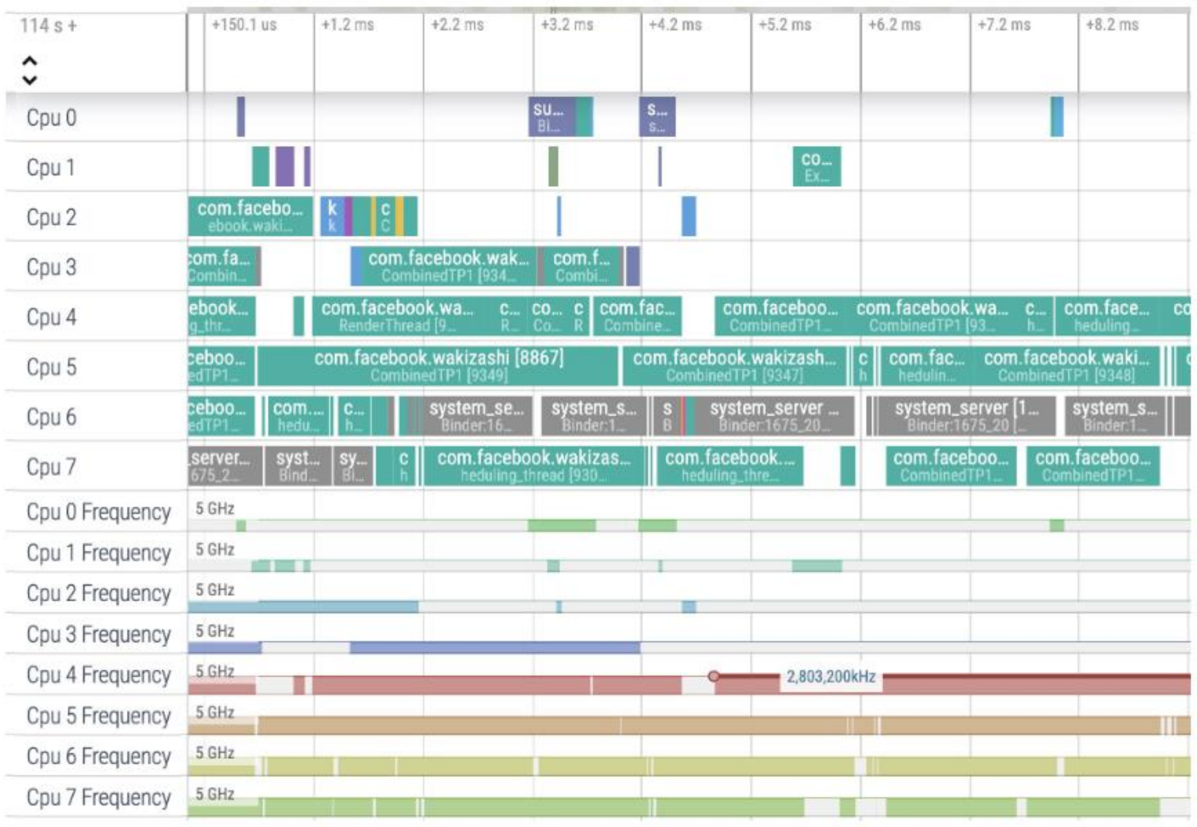
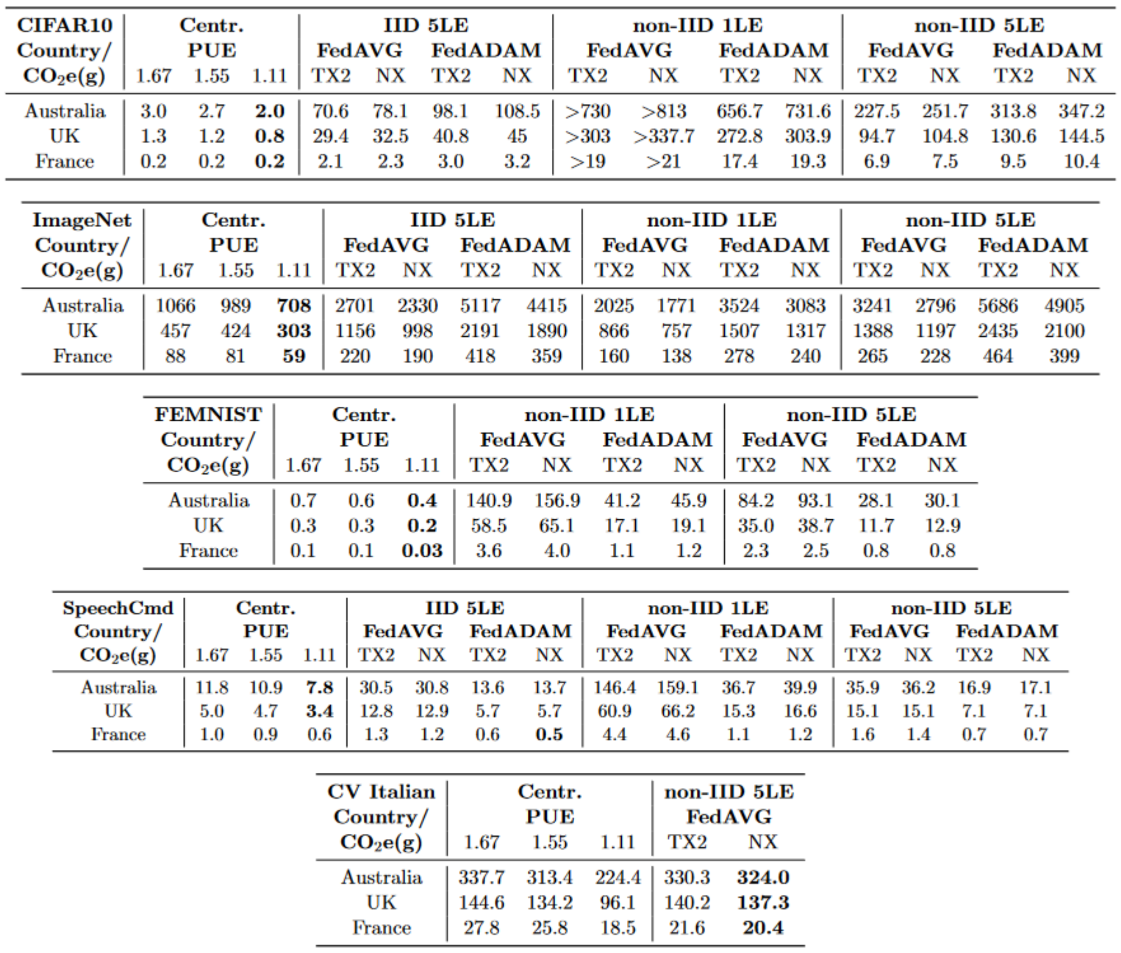


Figure 12: A Snapshot of the 8 Cores of the Phone while It is Performing an FL Task [2]

Figures 11 and 12 are snapshots of 8 cores (a cluster of four smaller 1.8GHz CPUs and a cluster of four bigger 2.8GHz CPUs) of a mobile phone in idle state and while running an FL task, respectively. These snapshots illustrate how computationally demanding FL is for ordinary devices like mobile phones. Cores from 4 to 7 (CPUs of the big cluster) are running the FL task at the maximal frequency of 2.8GHz. These cores in idle state are running at a lower frequency of 0.8GHz.

There have been experimental results and comparisons done with devices with exceptional chips like NVIDIA Tegra X2 and Jetson Xavier NX on well-known public datasets for image classification tasks (CIFAR, ImageNet, FEMNIST), as well as datasets used for keyword spotting from audio records (SpeechCmd, CV Italian) [1].



CIFAR10	Centr.			IID 5LE				non-IID 1LE				non-IID 5LE			
Country/	PUE			FedAVG		FedADAM		FedAVG		FedADAM		FedAVG		FedADAM	
CO_2e(g)	1.67	1.55	1.11	TX2	NX	TX2	NX	TX2	NX	TX2	NX	TX2	NX	TX2	NX
Australia	3.0	2.7	**2.0**	70.6	78.1	98.1	108.5	>730	>813	656.7	731.6	227.5	251.7	313.8	347.2
UK	1.3	1.2	**0.8**	29.4	32.5	40.8	45	>303	>337.7	272.8	303.9	94.7	104.8	130.6	144.5
France	0.2	0.2	**0.2**	2.1	2.3	3.0	3.2	>19	>21	17.4	19.3	6.9	7.5	9.5	10.4

ImageNet	Centr.			IID 5LE				non-IID 1LE				non-IID 5LE			
Country/	PUE			FedAVG		FedADAM		FedAVG		FedADAM		FedAVG		FedADAM	
CO_2e(g)	1.67	1.55	1.11	TX2	NX	TX2	NX	TX2	NX	TX2	NX	TX2	NX	TX2	NX
Australia	1066	989	**708**	2701	2330	5117	4415	2025	1771	3524	3083	3241	2796	5686	4905
UK	457	424	**303**	1156	998	2191	1890	866	757	1507	1317	1388	1197	2435	2100
France	88	81	**59**	220	190	418	359	160	138	278	240	265	228	464	399

FEMNIST	Centr.			non-IID 1LE				non-IID 5LE			
Country/	PUE			FedAVG		FedADAM		FedAVG		FedADAM	
CO_2e(g)	1.67	1.55	1.11	TX2	NX	TX2	NX	TX2	NX	TX2	NX
Australia	0.7	0.6	**0.4**	140.9	156.9	41.2	45.9	84.2	93.1	28.1	30.1
UK	0.3	0.3	**0.2**	58.5	65.1	17.1	19.1	35.0	38.7	11.7	12.9
France	0.1	0.1	**0.03**	3.6	4.0	1.1	1.2	2.3	2.5	0.8	0.8

SpeechCmd	Centr.			IID 5LE				non-IID 1LE				non-IID 5LE			
Country/	PUE			FedAVG		FedADAM		FedAVG		FedADAM		FedAVG		FedADAM	
CO_2e(g)	1.67	1.55	1.11	TX2	NX	TX2	NX	TX2	NX	TX2	NX	TX2	NX	TX2	NX
Australia	11.8	10.9	**7.8**	30.5	30.8	13.6	13.7	146.4	159.1	36.7	39.9	35.9	36.2	16.9	17.1
UK	5.0	4.7	**3.4**	12.8	12.9	5.7	5.7	60.9	66.2	15.3	16.6	15.1	15.1	7.1	7.1
France	1.0	0.9	0.6	1.3	1.2	0.6	**0.5**	4.4	4.6	1.1	1.2	1.6	1.4	0.7	0.7

CV Italian	Centr.			non-IID 5LE	
Country/	PUE			FedAVG	
CO_2e(g)	1.67	1.55	1.11	TX2	NX
Australia	337.7	313.4	224.4	330.3	**324.0**
UK	144.6	134.2	96.1	140.2	**137.3**
France	27.8	25.8	18.5	21.6	**20.4**

Figure 13: CO2e Emissions (expressed in grams, where lower values are better) for Both Centralized Learning and Federated Learning (FL) when they Reach the Target Accuracies across Different Tasks and Setups. The Results Include both FedAVG and FedADAM in both IID and non-IID partitions [1]

In Figure 13 we can see that in most cases CO_2e emissions, that are expressed in grams, are way smaller in a centralised setting. However, for SpeachCmd dataset and task of finding keywords, with PUE that France has, a federated learning strategy FedAdam on NX chips in cross-silo case with 5 local epochs per round has emitted less CO_2e then centralised learning in the same country. Also, FedAVG on NX chips has better performance than centralised learning regardless of the country on Common Voice Italian dataset.

From these results we see that choosing a federated learning strategy is also important for lowering carbon emissions because in most cases, a bit more optimised FedAdam emitted less CO_2e than FedAVG.

Emissions differ for synchronous and asynchronous federated learning as well. A study conducted on carbon emissions of synchronous FedAVG and asynchronous FedBuff show that asynchronous FL finishes training the model faster, but does that at the cost of higher CO_2e emissions.

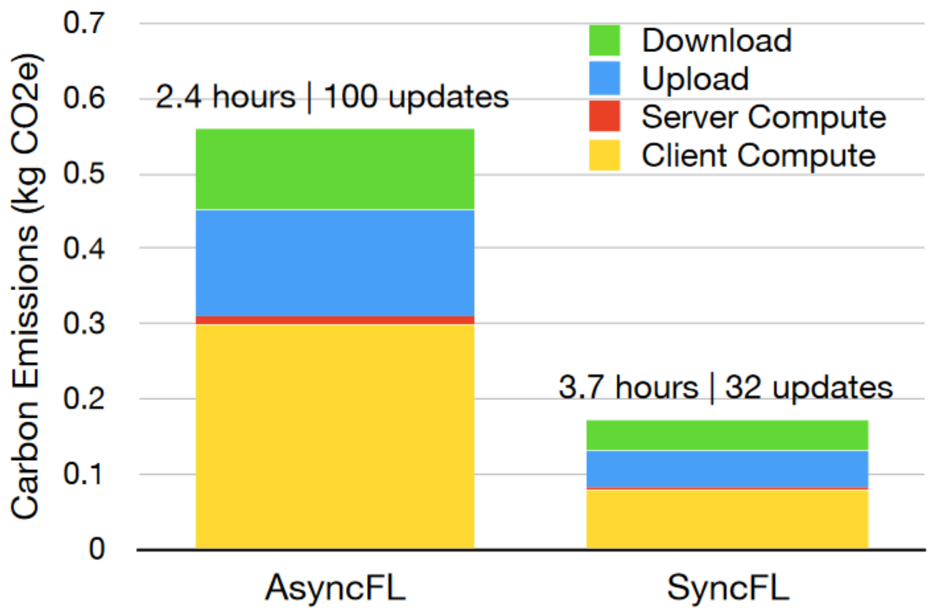


Figure 14: Carbon Emissions of Asynchronous and Synchronous FL (expressed in kilograms of CO_2e) with Respect to Time and Number of Updates [2]

In Figure 14 we see the differences between carbon emissions of asynchronous and synchronous federated learning of a language modelling task. This model was trained to predict the next word with inputted previous words in the sentence and make predictions while the characters of the word that is being predicted are being inputted in the device (in other words, autocorrect). It is trained on a public dataset – Reddit FL benchmark dataset that consists of Reddit posts that are in textual form. This dataset is a good representative of a real-world data distribution for mobile keyboard predictions. It consists of millions of users with 34 samples per user on average. Therefore, a study that has been done on this dataset measures emissions of greenhouse gases at a scale for an industrial FL system [2].

To better explain the reasons behind harmful carbon emissions depicted in the graph and differences between the emissions of asynchronous and synchronous federated learning, we will first introduce three new concepts – aggregation goal, concurrency and perplexity [2]. Aggregation goal represents the smallest number of client responses that must be received in order for server to aggregate them and update the model. Concurrency denotes the maximal number of clients that can train simultaneously. Perplexity is the normalised inverse probability of sequences. In this case sequences are sentences that consist of words and the probability of a sequence of words $S = w_1, \ldots, w_T$ autoregressively is $p(S) = \prod_{i=1}^{T} p(w_i|w_{<i})$. Therefore, perplexity is calculated as:

$$Perplexity(w_0, w_1, \ldots, w_i) = \left(\prod_{j=0}^{i-1} p(w_{j+1}|w_{\leq j}) \right)^{-\frac{1}{i}}$$

Perplexity measures the degree of uncertainty when a model generates a new token (token can be a character or a word or other segments of text) averaged over the length of a sequence. Since the goal is to be more certain what the next word will be, we want perplexity to be small.

For the results of experiment shown in Figure 14 stopping criteria was model reaching the target perplexity set to 175 (or lower).

Synchronous FL consists of rounds and at the beginning of each round server distributes the same model to the number of devices equal to concurrency. At the end of the round server updates the model if it receives at least as many updates as the set aggregation goal. We see that aggregation goal is less or equal to the concurrency. The process of distributing model to more devices then you are expecting to get an update from is called “over-selection” and is done because the users may drop out during the round because of lack of or bad internet connection, device shutting down etc. Since the server is waiting for most of the devices to finish calculating the update and then computing the next version of the model, waiting time of the server is significant, but the number of computations done by all the participants in this process is significantly lower than of asynchronous FL doing the same training.

Asynchronous FL selects a new device for training as soon as the server receives a client’s response. In this case, the number of devices training at any given time equals to concurrency. The devices are constantly being assigned new training tasks which leads to high client computation CO_2e emissions. With FedBuff algorithm the server model is not updated immediately upon receiving every client update. Updates are stored in a buffer and FedBuff chooses to update the global model when there are enough updates to reach the previously set aggregation goal. Many clients chosen earlier that failed to compute their update in time for the latest global model update are still training on old downloaded global model, doing additional computations that will not be used and wasting energy resources. To limit the client training time and avoid too big energy consumptions, 4-minute timeout is imposed in conducted experiments. There are more uploads and downloads than in synchronous training because of the constant new assignments given to the available contributor client devices. Server also does more computations than in the synchronous case, emitting more CO_2e.

In Figure 13, training time is the fixed parameter. Carbon emissions (in kg of CO_2e) and perplexity of asynchronous and synchronous federated learning are the measured parameters.

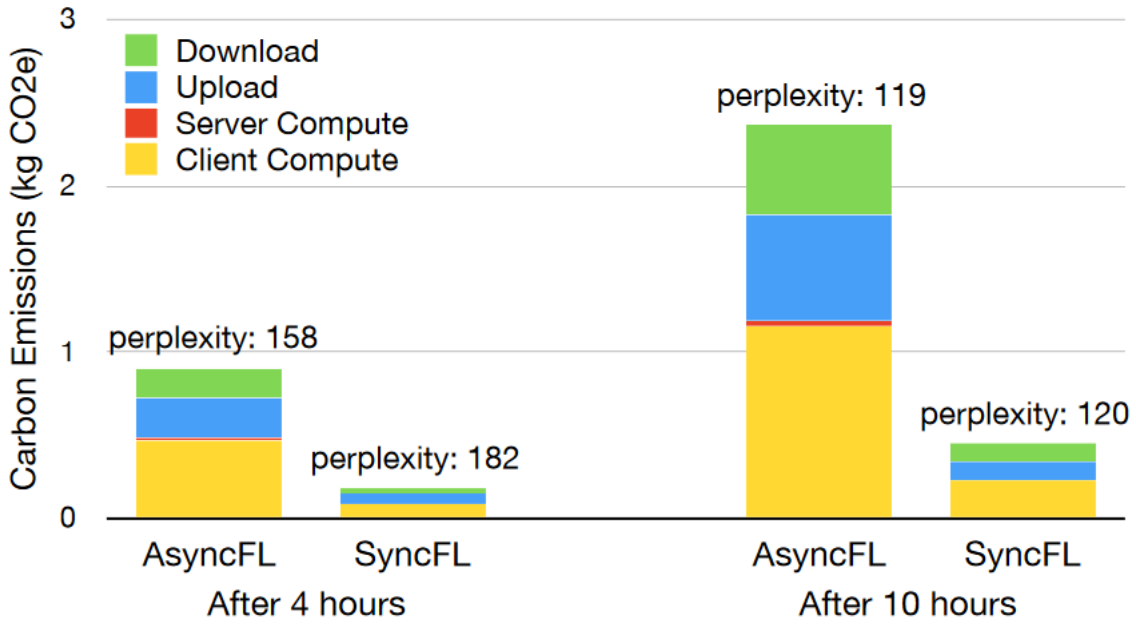


Figure 15: Carbon Emissions of Asynchronous and Synchronous FL (expressed in kilograms of CO_2e) with Respect to Perplexity for Fixed Time [2]

The perplexity in the experiment from the Figure 15 is computed using 20 held-out contributor clients with enough data samples for evaluation. We see that the asynchronous federated learning advances the model faster than synchronous – it has reached perplexity 158 in 4 hours while the synchronous has perplexity 182 for the same running time (lower perplexity is better). However, asynchronous FL has emitted around 3 times more CO_2e than synchronous in those four hours. In 10 hours, synchronous federated learning catches up to asynchronous with its perplexity 120. So, with enough training time, synchronous FL can reach similar results as asynchronous FL with smaller carbon emissions.

In all the experiments whose results were given in this chapter it is important to notice that most of them are done on public datasets that usually have smaller amount of data than the amount of the data the industrial federated learning is working with nowadays. Also, the experiments were conducted in controlled environments and in some cases, with better hardware than the one available for the real-world FL. Therefore, the carbon footprint of federated learning of state-of-the-art models is much larger. Since the experiments described for synchronous and asynchronous FL were the only ones done with an industrial FL in mind, it is estimated that carbon emissions of at-scale production models of the same tasks are expected to be around 10 times higher.

It is important to make federated learning "greener" so it could compete with centralised learning in terms of CO_2e emissions because of its rising popularity and the significant cost of creating state-of-the-art models with this innovative technology. Carbon footprint of FL needs to be taken into consideration for future implementations.

4. Elected Energy-efficient Training Strategies

Carbon emissions of federated learning are shaped by many factors, as we saw in the experimental results conducted in elected studies. To summarise, we conducted that carbon emissions of federated learning depend on:

- Data distribution among devices (IID and non-IID) and the amount of data
- Hardware efficiency (strength of client devices CPUs and GPUs)
- Geolocation and the "greenness" of electrical energy consumed by devices
- Internet connection (upload, download speed and connectedness of clients to the internet)
- Federated learning tasks (language learning, image classification, etc.)
- Federated learning strategy (FedSGD, FedAVG, FedAdam etc.)
- Synchronicity of FL (synchronous and asynchronous)
- Choice of hyperparameters and parameters (batch sizes, learning rates, number of rounds in synchronous FL, aggregation goals, local epochs, concurrency etc.)

Since our goal is to minimise carbon emissions, it is natural to choose the factors that lower the energy consumption. However, in practice, choosing all low CO_2e emitting factors usually isn't possible, especially those factors that tell us about the shape of the data. Still, we strive to optimise federated learning with respect to carbon emissions as much as possible.

In real-world applications federated setting is usually proposed for processing data that is already distributed among clients. That way their personal information is protected. Because of that, the data distribution between devices cannot be altered. Although the IID, cross-silo case is better for lowering emissions of green gasses, if the data is already distributed in a non-IID fashion among devices for given real-world task, that is the distribution we are working with. Only if we already have a whole dataset in one place and we want to simulate federated approach, it is important to choose IID setting for lowering the carbon emissions. However, with all the data in one place, it is common to use centralised learning. Maybe if our device doesn't have enough computational power to train the model on its own, we could divide the data and send it to more devices so the training can be possible. That is the real-world case in which we can set our data distribution. In industry-scale federated learning training of models this is highly unlikely to happen.

If the data is distributed among institutions that can't share data among themselves, like private hospitals, the data is IID distributed and these institutions would benefit a lot if they upgrade their hardware. Better hardware ensures faster computations with lower electrical energy consumptions which helps lowering the carbon emissions. Acquisition of said hardware is completely in hands of these companies that use models trained with federated learning for their benefit. They can also invest in fast internet.

However, federated learning is mostly used for training the data from smaller personal devices of users like mobile phones, tablets or laptops. Companies that make the model can't control how good the hardware of their clients is, or how good is their internet connection. In cases of multi-national companies whose applications are downloaded all around the world, clients' PUE varies significantly. One of things these companies can do to reduce the carbon emissions is to select client devices from locations with small PUE, good hardware and good internet connection and use only those devices for training the model. This sounds excellent, but it introduces a demographic bias in the model. This bias exists because the pool of selected clients is not a good representation of all users of these companies' services [1]. It is possible that the clients from the greener locations or with better internet connection don't have sufficient amount of data samples for training or they might represent a skewed data distribution. Client devices should be chosen more wisely to be as accurate sample from the distribution of all client data as possible. Unfortunately, it is impossible to avoid losing electrical energy for computations on disconnected devices or losing time for computing the update on phones with not so powerfull CPUs and get a representative model.

Similarly, carbon emissions can be optimised by selecting the clients with the better devices that have more computational power that could be used for model training. However, such selection also induces potential biases. Good news are that the personal devices are getting better every year and internet connection is becoming faster all around the world. This is making federated learning easier to implement as time progresses.

Factors that are mostly in our power are choice of federated learning strategy, deciding between synchronous FL and asynchronous FL and choice of model's hyperparameters and parameters.

Since asynchronous federated learning advances faster than synchronous at the cost of higher carbon emissions, if the time is not the most important factor, we can choose to only work with synchronous FL. If that is not the case, we can count mostly on tuning the parameters to decrease carbon footprint of asynchronous FL.

4.1 Prediction of Carbon Footprint of FL with Respect to Parameters

In the research paper “Green Federated Learning” [2] has been discovered that factors that impact the emissions of greenhouse gases the most are concurrency and time to reach target accuracy. For synchronous FL time to reach target accuracy corresponds to the number of rounds and for asynchronous to the wall-clock time needed for training. Other parameters, such as learning rates, batch sizes, local epochs and aggregation goals also impact the CO_2e emissions because they influence convergence of the model to the target accuracy and that impacts the time to finish the training of the model. Therefore, tuning all these parameters would be the best solution for carbon emissions reduction.

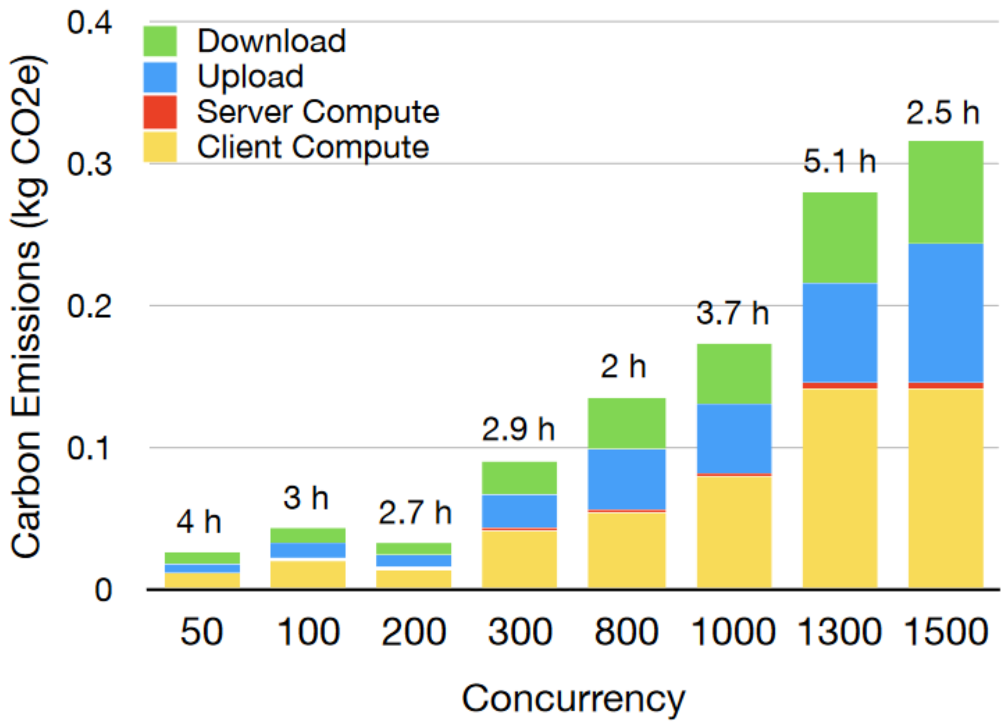


Figure 16:Carbon Emissions (expressed in kilograms of CO_2e) with Respect to Concurrency [2]

In Figure 16 we see that the concurrency has a great impact on carbon emissions. As concurrency increases, so do the CO_2e emissions. On each bar is stated the time the model needs to reach the target accuracy with the said concurrency. Time varies for each concurrency. Also, we can conclude that overall, higher concurrency accelerates model convergence. However, if concurrency is for example, increased 10 times, that increases electrical power usage 10 times, but only reduces convergence time 1.5-2 times. Therefore, changing the concurrency greatly impacts energy consumption and CO_2e emissions, but has a smaller impact on training duration.

Benefits of higher concurrency – faster convergence, better model performance, don't scale linearly. High concurrency is responsible for slightly better and faster convergence.

There is a need to better understand the relationship between carbon emissions, concurrency and time. There are experimental results that validate the claim that the CO_2e emissions have a linear relationship with the product of concurrency and time in which model reaches target accuracy. Figure 17 illustrates how carbon emissions of synchronous federated learning are correlated linearly with the product of rounds it takes to reach target accuracy and concurrency. Similarly, Figure 18 shows the linear correlation of carbon emissions of asynchronous federated learning with the product of time it takes to reach a target accuracy and concurrency. In these Figures, the results were given for carbon emissions of download, upload and client computation. Carbon emissions of server computations are negligible and therefore, they are not shown. The points on scatterplots each represent a different training run of a language learning model described in the last chapter for differences between carbon emissions of asynchronous and synchronous FL. Figures 17 and 18 also show a goodness-of fit measure R^2 for linear regression models, since linear regression is used to find the fitting line for said scatterplots.

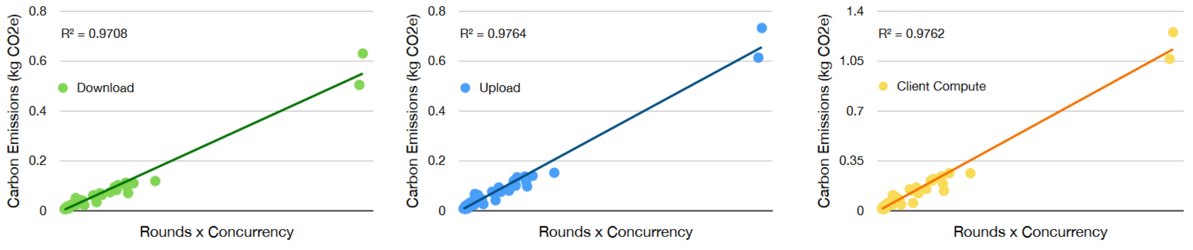


Figure 17: Linear Correlation of Carbon Emissions of Synchronous FL and the Product of Rounds it Takes to Reach a Target Accuracy and Concurrency [2]

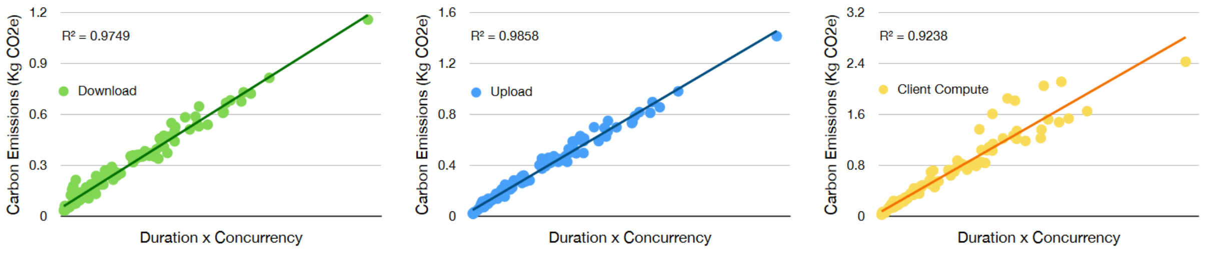


Figure 18: Linear Correlation of Carbon Emissions of Asynchronous FL and the Product of Time Duration it Takes to Reach a Target Accuracy and Concurrency [2]

In Figures 17 and 18 we see that the values of R^2 are near 1, which means that the linear model is a very good representation of the given points. Linear model is a good fit. Because of that, we can conclude that the product of rounds and concurrency, for synchronous FL, and product of duration and concurrency, for asynchronous FL, represent a good way to predict the carbon footprint of federated learning. Carbon footprint is proportional to concurrency and it's also proportional to the time to reach target accuracy (rounds/duration).

Using this knowledge, we can estimate the carbon emmisions of federated learning and tend to minimise it. So, we need to know or estimate the concurrency and number of rounds (or duration), parameters that are proportional to the carbon footprint, for our prediction. Concurrency is a hyperparameter of federated learning and it is set before the training starts. Time needed for the model to reach target accuracy can be estimated with FL simulation tools that are commonly used in the industry. However, time relies on many factors, although mostly on appropriate selection of hyperparameters of the model (that can be dictated by us). For estimation of carbon emissions, besides concurrency and estimated rounds (or duration), we need to know the coefficient of proportionality. Coefficient of proportionality is the slope of the lines in Figures 10 and 11. It depends on numeruous factors, some of them being the infrastructure used for federated learning and nature of the given federated learning task, as well as user population. This coefficient can be estimated by measuring carbon emissions of the task in several settings, creating few data points and doing the linear reggresion for that small amount of points. This might not be the perfect way of estimation with regards to the CO_2e emmited during this tasks, but it is the way it can be estimated in practice.

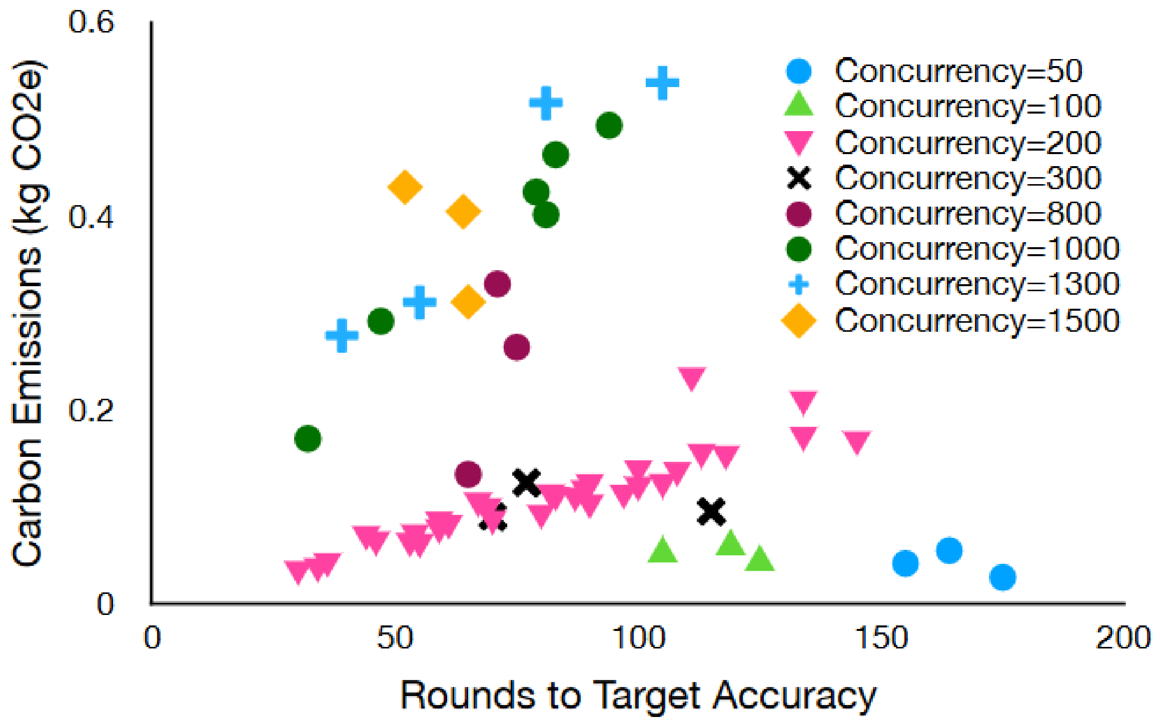


Figure 19: Carbon Emissions of Synchronous FL's Increase with the Product of Number of Rounds it Takes to Reach a Target Accuracy and Concurrency [2]

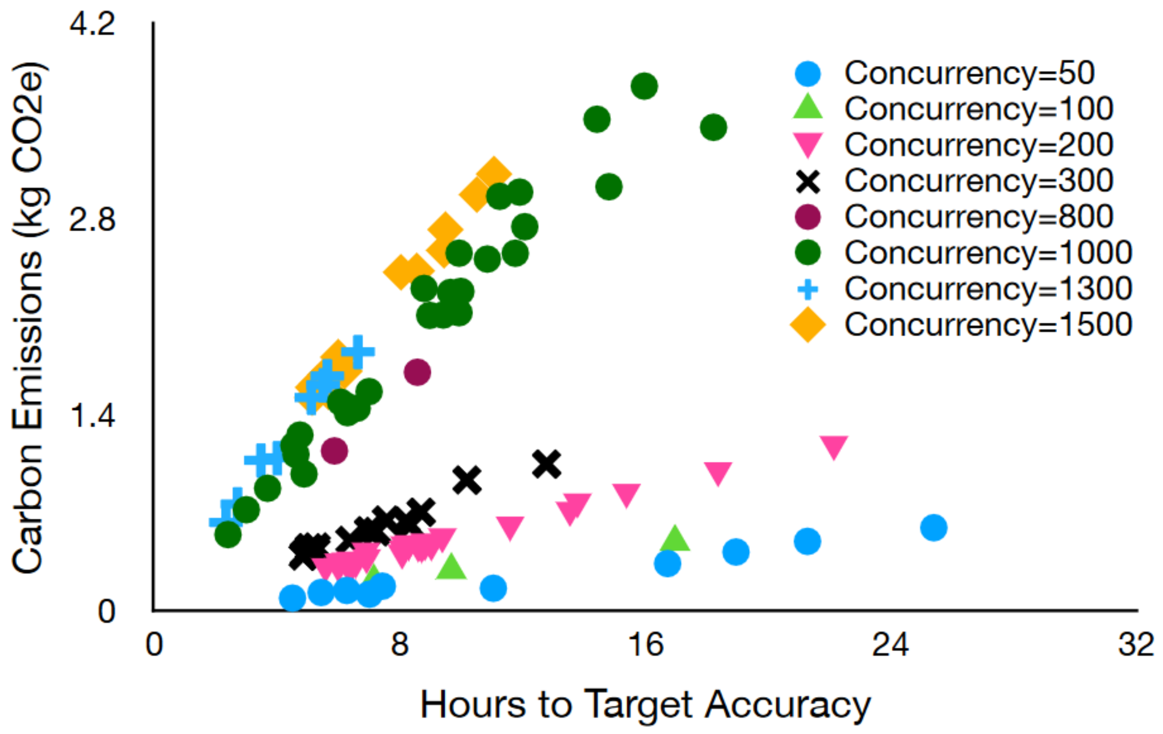


Figure 20: Carbon Emissions of Asynchronous FL's Linear Increase with the Product of Time it Takes to Reach a Target Accuracy and Concurrency [2]

Figures 19 and 20 are scatterplots for training synchronous and asynchronous federated learning, respectively, where each point represents a training run with a set concurrency (represented with the points colour and shape) and different target accuracy. On the X axis is the number of rounds or time needed to reach the target accuracy and on the Y axis are carbon emissions of the training process presented in kg of CO_2e. Here we can see the trade-off between the number of rounds (or time), performance and CO_2e emissions of the federated learning model. As the number of rounds or time to reach the target accuracy increases and the hihger is the concurrency, carbon emissions increase.

Sometimes, high concurrency is needed because of its more robust privacy guarantees. In those cases, hyperparameter tuning becomes esential for reducing the training time and, in return, reducing the carbon footprint.

4.2 Federated Hyperparameter Tuning Strategy

Training time for reaching the target accuracy is influenced by many factors. Some of them can be further optimised. Training time can, for example, be lowered with better choice of hyperparameters such as learning rate or batch size. Concurrency is also a hyperparameter that can be tuned to minimise energy consumption and therefore also the CO_2e emission cost of training the model.

Hyperparameter tuning is one of the main energy-efficient training methods used for lowering emissions of greenhouse gases of deep learning algorithms, as stated in “A Survey on Green Deep Learning” [3]. Acknowledging the importance of hyperparameter and parameter optimisation in federated setting and the benefits shown in literature for optimising concurrency, it is safe to assume that hyperparameter tuning would be beneficial for lowering the carbon emissions in federated learning.

Hyperparameter optimisation is, mathematically, a process of finding a set of hyperparameters to achieve minimum loss or maximum accuracy of an objective neural network [28]. The choice of hyperparameters influences both the structure of the network (for example, the number of hidden layers) and training accuracy (for example, learning rate and batch size). The hyperparameters we want to optimise can be integers, floating points, categorical data, or binary data, with a distribution of the search space. Search space defines all hyperparameter candidates [3]. It contains all the possible values our hyperparameter can take. Hyperparameters can be chosen manually, with respect to our knowledge of search space.

Manual search consists of experimenting with different combinations of hyperparameters and its values for the selected model, performing the training and taking the best model with the best performance [26]. This method takes a lot of time and effort and it isn’t very efficient. It consists of trying manually all the combinations of possible values of hyperparameters in our model and picking the one that performed the best. In centralised setting with a smaller amount of data, this method can be used; however, in federated setting with large amount of data it isn’t preferred.

Except the manual method, there are better and more often used search methods in machine learning and deep learning for optimising hyperparameter value like grid search, random search, Bayesian optimisation or successive halving. [28,30].

Grid search is a straightforward search algorithm similar to manual search in a sense that it goes through the all potential combinations of hyperparameters’ values from the search space, but it does that automatically. It will construct many versions of the model with all possible combinations from the grid of potential discrete hyperparameter values that it had previously constructed and then return the best one.[3] In practice, this algorithm is preferable when we have enough experience with the hyperparameters we want to optimise to enable the definition of a narrow search space and no more than 3 hyperparameters that need to be tuned simultaneously. Although other search methods may have more favourable features, grid search is still widely used method because of its mathematical simplicity [28].

Random search is a search approach in which the values of hyperparameters are selected randomly instead of using a predetermined set of numbers, as in grid search. It employs a randomised search over hyperparameters from certain distributions over possible parameter values [28]. In each iteration, it tests a different set of hyperparameters and records the model's performance. The searching process continues untill the desired accuracy is reached, or until the predetermined fixed number of iterations is exhausted. Eventually, it returnes the combination that yields the best result after several iterations. This method reduces unnecessary computation and is faster than the previously mentioned approaches. Random search, in contrast to grid search, tests only a fixed number of hyperparameter combinations. It navigates the hyperparameter space randomly to find the optimal set. The main advantage of this approach is that, in most cases, it can produce results comparable to grid search, but in a shorter amount of time. Also, because of its randomness, it yields better results when some hyperparameters are not uniformly distributed. However, it's biggest drawback is that it's possible that the outcome is not the ideal hyperparameter combination. The differences between grid search and random search in the exploration of hyperparameters' search space is illustrated in Figure 21.

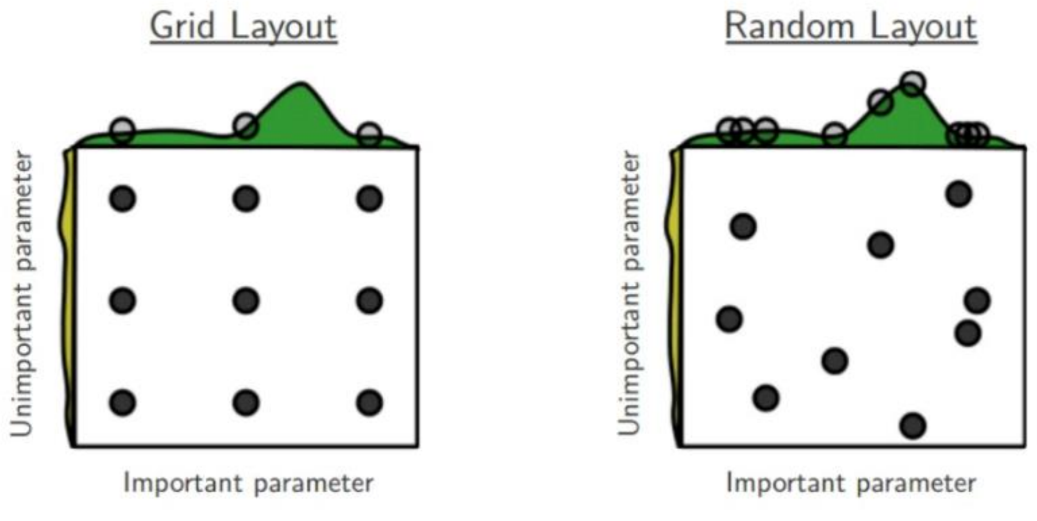


Figure 21: Grid Search and Random Search over the Parameter Search Space[28]

[3]https://www.analyticsvidhya.com/blog/2022/02/a-comprehensive-guide-on-hyperparameter-tuning-and-its-techniques/

Bayesian optimisation is a search method that treats the search for optimal hyperparameters as an optimisation problem. In contrast to grid search and random search methods that often waste resources by evaluating many unsuitable hyperparameter combinations because they don't consider previous results, Bayesian optimisation considers previous evaluation results when selecting the next hyperparameter combination. It uses a probabilistic function to choose the combination that is most likely to yield the best results. Using the probabilistic model to evaluate hyperparameters is computationally inexpensive compared to directly evaluating the objective function. This approach typically finds a good hyperparameter combination in a relatively few iterations. However, unlike grid search or random search, Bayesian optimisation must be computed sequentially, which doesn't allow distributed processing. Consequently, Bayesian optimisation takes longer but uses fewer computational resources. Its drawback is that it can be hard to implement and that it requires an understanding of the underlying probabilistic model.

It is important to note that, since hyperparameter tuning can end up being time-consuming process that is computationally costly, to lower the use of resources we can use early stopping. Early stopping evaluates the trials and decides when to halt them. Even though there is the need to do additional computations while training the neural network for hyperparameter optimisation, studies show [2, 3] that tuning the parameters lowers the overall energy consumption and therefore CO_2e emissions produced by training because it optimises it.

Optimisation method that performs very well in deep neural networks is successive halving algorithm (SHA) [28, 13]. Hyperparameter optimisation in deep neural networks is more likely to be a trade-off between accuracy and computational resources since training a complex network is a very costly process. Therefore, SHA is an algorithm that can outperform earlier mentioned searching algorithms in saving resources for hyperparameter tuning of state-of-the-art models. Since federated learning is used for creating complex models, SHA is a very good candidate for tuning the model. It uses random search as a sampling method and a bandit-based early stopping policy. Instead of evaluating models when they are fully trained to convergence, like earlier methods, SHA evaluates the intermediate results to determine whether to terminate it or not.

SHA uses Multi-Armed Bandit framework to solve the hyperparameter optimisation problem. This framework is a guide towards the solution of the so called „Multi-Armed Bandit Problem"[4]. This problem can be described as a problem of a gambler that faces a row of slot machines (or "one-armed bandits"). Each slot machine provides a different and unknown reward probability. The gambler's goal is to get as much money as possible, so he wants to maximise the total reward over a series of lever pulls. He seeks to find an optimal strategy for allocating pulls among machines, so that the expected cumulative reward is maximised. Also, the number

[4] https://medium.com/@vireshj/efficient-hyperparameter-tuning-with-successive-halving-7f50a57bb160

of times he can pull the lever is finite. So, he needs to find the trade-off between the exploration (pulling the levers of different machines to find how big the reward can be) and exploitation (pulling the lever of the slot machine with the highest expected reward based on the information already gathered). In case of successive halving algorithm, each arm of the bandit (a slot machine) corresponds to a different hyperparameter configuration. Pulling an arm is akin to evaluating a configuration with a specific computational budget. The gamblers reward is the performance metric we are optimising.

The Successive Halving Algorithm (SHA) can be described as follows [28]: we begin by setting an initial finite budget and the number of trials η. The budget could refer to the number of iterations, epochs, the total training time or, in the synchronous federated learning setting – the number of rounds. Initially, all hyperparameter sets are uniformly queried for a portion of this initial budget. This means that each trial is allocated a small fraction of the total budget to start with. Once this initial phase is completed, the performance of all the trials is evaluated. Then the algorithm eliminates the bottom half of the trials, those that performed the worst. The budget for the remaining trials is then doubled, allowing them to continue with a greater allocation of resources. This process of evaluating, halving the trials, and doubling the budget continues iteratively. The cycle is repeated until only one trial remains, which is deemed the best-performing hyperparameter set.

This method efficiently narrows down the most promising hyperparameter sets in deep neural networks by progressively increasing the budget for the better-performing trials, thus maximising the chances of finding an optimal set within a finite budget.

Optimising hyperparameters of complex models in federated setting is far more challenging than in centralised setting because we don't have all the data in one place, which makes the training and validation difficult. Research paper "Federated Hyperparameter Tuning: Challenges, Baselines, and Connections to Weight-Sharing" [13] tackles the challenges federated setting proposes for optimisation of models' hyperparameters.

In federated setting the data on which we are training our model is distributed on a high number of various small devices. These devices are usually phones, tablets or laptops with limited computational and communication capabilities and also, since they are mostly personal devices, their privacy is protected by privacy techniques such as differential privacy that limit the number of times user data can be accessed. Therefore, it isn't possible to access one device many times and the resources it provides are limited. Training and validation become problematic since the devices for training the model can change in each round (in synchronous federated learning). Since applying common hyperparameter optimisation methods to federated learning approaches can be costly because evaluation may require performing additional training steps on devices with limited capabilities and accessibility, there was a need to adjust hyperparameter tuning for federated approach.

Hyperparameter tuning method made for federated learning that tackles mentioned challenges is called FedEx [13]. FedEx utilises a weight-sharing technique widely used in neural architecture search (NAS) and applies it to the federated architecture. Weight-sharing integrates noisy validation signal, it can simultaneously tune and train the model, and it evaluates personalisation as part of training instead of incorporating a costly separate step. FedEx extends capabilities of standard weight-sharing that typically focuses on architectural hyperparameters (for example: choice of layer or activation) so that it can cover crucial settings like those of local stochastic gradient descent (SGD). This method can be applied on any local training-based federated learning approaches such as FedAvg, FedProx, etc. and it is the leading method for hyperparameter tuning of state-of-the-art federated learning models.

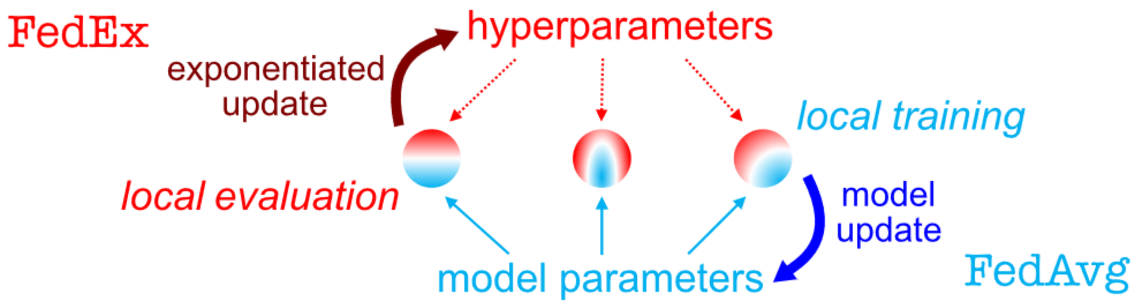


Figure 22: Application of FedEx to FedAVG [14]

In Figure 22 we see where FedEx takes place in case of it being used to optimise hyperparemeters of FedAVG federated learning approach. It alternates standard updates to model weights obtained by aggregating results from local training with exponentiated gradient updates to hyperparameters, calculated by aggregating results from local validation.

We will now show a way to represent and decompose federated learning methods such that the hyperparameter tuning methods can be applied to them. At first, we will propose a global hyperparameter tuning problem. This problem can also be solved using some classical hyperparameter optimisation methods, like random search. However, then we will introduce FedEx algorithm which is made just for federated hyperparameter optimisation and successfully tackles the challenges of federated setting.

In federated learning (FL), the goal is to optimise over a network of heterogeneous clients, $i = 1, \dots, n$, each possessing their own training, validation, and testing sets, denoted as T_i, V_i, and E_i, respectively. Let's denote the average loss over a dataset S for a machine learning model parameterised by $w \in \mathbb{R}^d$ with $L_S(w)$. For hyperparameter optimisation, we consider a class of algorithms Alg_a which are hyperparameterised by $a \in A$. These algorithms use federated access to the training sets T_i to output an element of $\mathbb{R}^d$. "Federated access" here means that each iteration corresponds to a communication round where Alg_a accesses a batch of clients that can perform local training and validation.

Now, we can define the **global hyperparameter optimisation problem** as:

$$\min_{a \in A} \sum_{i=1}^{n} |V_i| L_{V_i}\left(Alg_a\left(\{T_j\}_{j=1}^{n}\right)\right)$$

However, if we want to obtain a device-specific local model, where we take a model trained on all clients and finetune it on each individual client before evaluation, we can transform the global hyperparameter optimisation problem slightly by using decomposition. We assume that $a \in A$ can be represented as $a = (b, c)$ where $b \in B$, $c \in C$ and $A = B \times C$. Here we have $c \in C$ that encodes settings of a local training algorithm we will denote with Loc_c that takes for input initialisation $w \in \mathbb{R}^d$ and training set S and outputs a model $Loc_c(S, w) \in \mathbb{R}^d$. Then $b \in B$ configures an aggregation function, denoted Agg_b, that takes for input initialisation $w \in \mathbb{R}^d$ and outputs of the model $Loc_c(S, w)$ and then it outputs a model parameter. The local training algorithm Loc_c used by class of algorithms Alg_a can be assumed to be the same as the finetuning algorithm. Then we define the **personalised federated learning objective** as the hyperparameter optimisation problem in the personalised setting that can be written as:

$$\min_{a=(b,c) \in A} \sum_{i=1}^{n} |V_i| L_{V_i}\left(Loc_c\left(T_i, Alg_a\left(\{T_j\}_{j=1}^{n}\right)\right)\right)$$

Since FedEx is a method that tunes the local parameters, we will focus on settings where the local training hyperparameters c make up a significant portion of the total hyperparameters $a = (b, c)$. To derive FedEx, we focus on tuning only the local training hyperparameters c in Loc_c. The objective of FedEx then simplifies to finding the optimal initialisation $w \in \mathbb{R}^d$ and the best local hyperparameters $c \in C$, replacing the personalised objective with:

$$\min_{c \in C, w \in \mathbb{R}^d} \sum_{i=1}^{n} |V_i| L_{V_i}\left(Loc_c(T_i, w)\right)$$

Let's now focus on how weight-sharing approach in neural architecture search (NAS) works. We will later use it for our federated hyperparameter optimisation algorithm.

For a set C of network configurations NAS is often posed as the bi-level optimisation:

$$\min_{c \in C} L_{valid}(w, c) \text{ s.t. } w \in \arg\min_{u \in \mathbb{R}^d} L_{train}(u, c)$$

Here, L_{valid} and L_{train} are evaluating a single configuration with the given weights. We can instead of a bi-level optimisation consider solving the following "single-level" empirical risk minimisation (ERM) that is equivalent:

$$\min_{c \in C, w \in \mathbb{R}^d} L_{train}(u, c) + L_{valid}(w, c) = \min_{c \in C, w \in \mathbb{R}^d} L(u, c)$$

Early neural architecture search approaches were expensive due to the need to fully or partially train numerous architectures in a vast search space. This problem can be simplified to a problem of training a single "supernet" that encompasses all architectures in the search space C. We can do that through a "stochastic relaxation", where the loss is the expectation with respect to sampling c from a distribution over C. The shared weights $w \in \mathbb{R}^d$ can be updated using SGD by sampling architecture c and using an unbiased estimate of the gradient $\nabla_w L(w, c)$. We focus on the case where the distribution over C stays fixed (the other case is when the distribution over C is adapted). Then, that distribution is some distribution $\mathcal{D}_\theta$ with parameter $\theta \in \Theta$. The optimisation objective for ERM than transforms into the stochastic relaxation objective:

$$\min_{\theta \in \Theta, w \in \mathbb{R}^d} \mathbb{E}_{c \sim \mathcal{D}_\theta} L(u, c)$$

This objective can be solved with a method that uses exponentiated update $\nabla_w \mathbb{E}_{c \sim \mathcal{D}_\theta} L(w, c)$ with the standard SGD update (explained in Chapter 2 – Federated Learning) to weights w and it is guaranteed to converge, under certain conditions, to the ERM objective. It is also important to note that, since architectural hyperparameters are often discrete decisions, a natural choice of $\mathcal{D}_\theta$ is as a product of categorical distributions over simplices. Therefore, natural update scheme here is exponentiated gradient, where each successive θ is proportional to $\theta \odot \exp(-\eta \tilde{\nabla})$ (where $\odot$ is a Hadamard product – component-wise multiplication for matrices),η is a step-size, and $\tilde{\nabla}$ is an unbiased estimate of $\nabla_\theta \mathbb{E}_{c \sim \mathcal{D}_\theta} L(w, c)$ that can be calculated using the re-parameterisation trick.

When we apply this method to our search for local hyperparameters, for FedEx, we get that our objective is:

$$\min_{\theta \in \Theta, w \in \mathbb{R}^d} \sum_{i=1}^{n} |V_i| \mathbb{E}_{c \in \mathcal{D}_\theta} L_{V_i}\big(Loc_c(T_i, w)\big)$$

Applying NAS is possible because for fixed c personalised objective is equivalent to the classic train-validation split objective for meta-learning with Loc_c as the base-learning training algorithm. It is a single-level function of w and c and this objective is upper-bounded by the original objective, so any solution will be at least as good for the original hyperparameter tuning objective.

Finally, we can present the FedEx algorithm for tuning local hyperparameters in federated learning approaches:

Algorithm 2 [13]: FedEx

```
Input: configurations $c_1, \dots, c_k \in C$, setting $b$ for $Agg_b$, schemes for setting step-size $\eta_t$ and baseline $\lambda_t$, total number of steps $\tau \geq 1$

Initialise $\theta_1 = \frac{\mathbf{1}_k}{k}$ and shared weights $w_1 \in \mathbb{R}^d$

for communication round $t = 1, \dots, \tau$ do
    for client $i = 1, \dots, B$ do
        send $w_t$,$\theta_t$ to client
        sample $c_{ti} \sim \mathcal{D}_{\theta_t}$
        $w_{ti} \leftarrow Loc_{c_{ti}}(T_{ti}, w_t)$
        Send $w_{ti}$, $c_{ti}$, $L_{V_{ti}}(w_{ti})$ to server
    $w_{t+1} \leftarrow Agg_b(w, \{w_{ti}\}_{i=1}^{B})$
    $\nabla_j \leftarrow \frac{\sum_{i=1}^{B} |V_{ti}| \left(L_{V_{ti}}(w_{ti}) - \lambda_t\right) 1_{c_{ti}=c_j}}{\theta_{t[j]} \sum_{i=1}^{B} |V_{ti}|}$, $\forall j$
    $\theta_{t+1} \leftarrow \frac{\theta_{t+1}}{\|\theta_{t+1}\|_1}$
Output: model $w$ and hyperparameter distribution $\theta$
```

Here we see that with $c_1, \dots, c_k \in C$ derived from the distribution $\mathcal{D}_\theta$, step size η_t, baseline λ_t and already optimised parameter $b \in B$, FedEx tunes the parameter (w, θ) of the neural network. Since FedEx focuses primarily on tuning local training hyperparameters, but does not directly handle the hyperparameters of the aggregation step $b \in B$ in federated learning algorithms, this method is advised to be used in combination with some other hyperparameter optimisation method. The recommended methods in literature [13] are successive halving algorithm (SHA) and random search (RS).

Algorithm 3 is the successive halving algorithm (that can be seen as random search for specific settings of η and R) for tuning the federated learning hyperparameters (without FedEx). It is applied to personalised federated learning objective.

Algorithm 3 [13]: Successive halving algorithm (SHA) applied to personalized FL. For the non-personalized objective, replace $L_{V_{ti}}(w_i)$ by $L_{V_{ti}}(w_a)$. For random search (RS) with N samples, set $\eta = N$ and $R = 1$.

Input: distribution D over hyperparameters A, elimination rate $\eta \in N$, elimination rounds $\tau_0 = 0, \tau_1, \dots, \tau_R$ sample set of η^R hyperparameters $H \sim D^{[\eta^R]}$ initialize a model $w_a \in \mathbb{R}^d$ for each $a \in H$

```
for elimination round $r \in [R]$ do
    for setting $a = (b, c) \in H$ do
        for comm. round $t = \tau_{r-1}, \dots, \tau_r$ do
            for client $i = 1, \dots, B$ do
                send $w_a, c$ to client
                $w_i \leftarrow Loc_c(T_{ti}, w_a)$
                Send $w_a, L_{V_{ti}}(w_i)$ to server
            $w_a \leftarrow Agg_b(w_a, \{w_i\}_{i=1}^{B})$
            $s_a \leftarrow \sum_{i=1}^{B}|V_{ti}|L_{V_{ti}}(w_i) / \sum_{i=1}^{B}|V_{ti}|$
    $H \leftarrow \left\{a \in H: s_a \leq \frac{1}{\eta} - quantile(\{s_a: a \in H\})\right\}$
```

Output: remaining $a \in H$ and associated w_a

Tuning federated learning architectures like FedAVG with this algorithm is proven to be successful. However, there are previously mentioned problems like the fact that the validation data is spread across multiple devices making it hard for SHA to evaluate performance consistently. This is a problem which is tackled with FedEx. By using validation data from one round, SHA could potentially make noisy elimination decisions, early-stopping configurations that might actually be good because of a difficult set of clients on a particular round. To make better decisions, we can try using more validation data from previous rounds. There is a method called power decay weighting that can be used for evaluation. This is a method in which where the importance of a round decreases over time. A round is discounted by some constant factor (that we chose in advance) for each time step it is in the past.

These are the experimental results for model performance of using Algorithm 3 for tuning the FedAVG on datasets FEMNIST and CIFAR (image classification datasets) with factors 0, 0.5 and 1 for power decay weighting of the rounds' validation data:

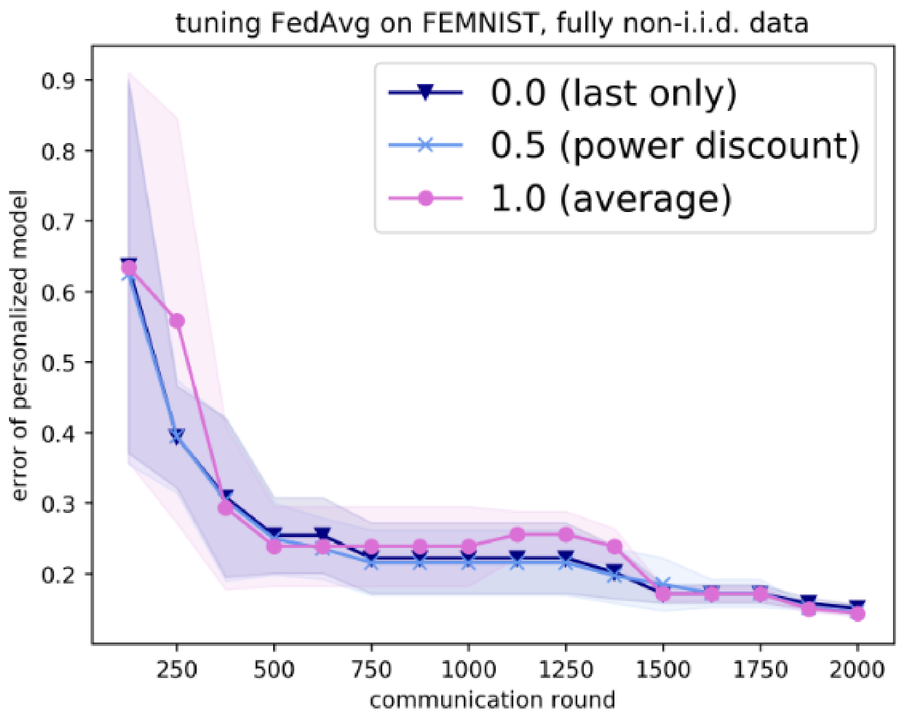


Figure 23: Tuning FedAVG with SHA on FEMNIST [13]

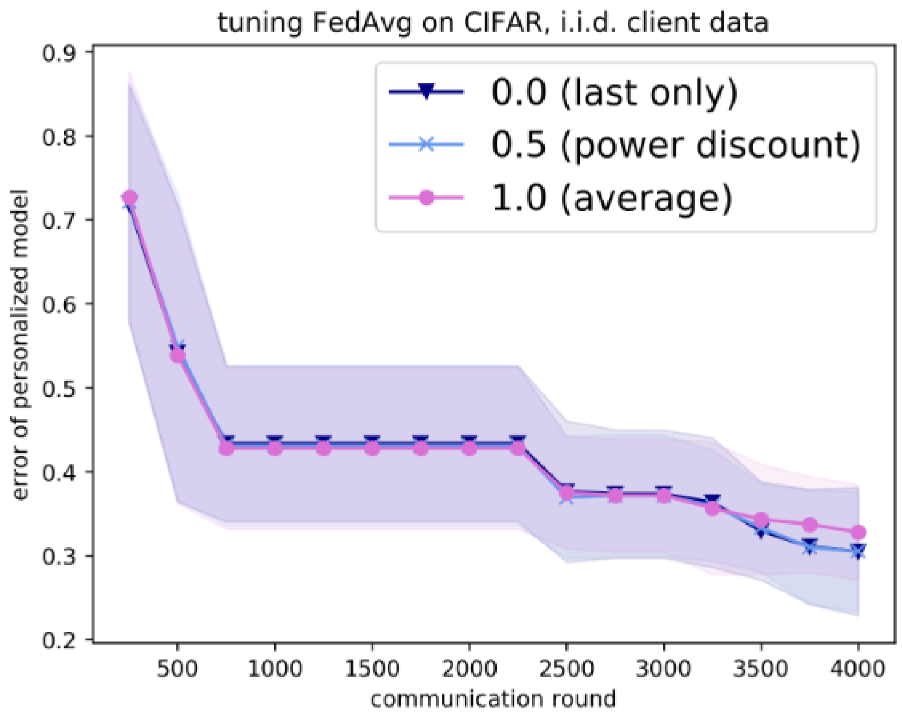


Figure 24: Tuning FedAVG with SHA on CIFAR [13]

In Figures 23 and 24 we see that the performance of SHA does not sufficiently change when we use just the most recent round's validation error (for factor 0) or when we take the average performance (for factor 1) or even when the factor is 0.5 meaning that the performance of previous rounds still counts, but the recent ones are deemed more important. Therefore, we conclude that incorporating more validation data this way than it is used by SHA by default doesn't significantly affect results. However, FedEx uses validation on each round to update a client's hyperparameters distribution used to sample configurations to send to devices, so it can update on each step. It doesn't have to wait for elimination round like SHA.

The best result in literature [13] yields a combination of SHA and FedEx. Specifically, FedEx wrapped in SHA. Successive halving is a great choice of a wrapper because it performs well on deep neural networks. Therefore, it is an algorithm recommended for state-of-the-art models trained with federated learning. SHA can be viewed as a baseline tuning method for all complex models, regardless of setting being centralised or federated. Wrapping FedEx with SHA is beneficial for many reasons. Firstly, the wrapper SHA can tune the settings $b \in B$ of for the aggregation step Agg_b, which FedEx cannot achieve alone. Additionally, FedEx itself has a few hyperparameters, such as the baseline λ_t, that need tuning. So, wrapping FedEx with SHA means that the hyperparameters inside the FedEx algorithm are also optimised with SHA. By running multiple seeds and potentially using early stopping, we can adopt more aggressive step-sizes in FedEx, with the wrapper discarding any configurations that lead to poor results. Secondly, this approach allows for a direct comparison between FedEx and a regular hyperparameter optimisation scheme applied to the original federated learning algorithm, such as FedAVG. By using the same scheme (in our case, SHA) to wrap both FedEx and tune FedAVG, we can evaluate their performance on equal footing. Lastly, employing the wrapper enables us to determine the configurations $c_1, \dots, c_k$ given to FedEx through a local perturbation scheme while still exploring the entire hyperparameter space. This ensures a comprehensive search and tuning process, maximising the effectiveness of FedEx.

Let's take a look at how local perturbation method works. Instead of picking configurations randomly from a large set (which can be unstable), the local perturbation method starts with one random configuration $c_1 \epsilon C$ and then uniformly samples similar ones $c_1, \dots, c_k \in C$ from its neighbourhood. This helps us to find good hyperparameters more reliably by keeping the configurations not too different from each other. For example, for continuous hyperparameters like step-size or dropout that are drawn from an interval $[a, b] \subset \mathbb{R}$ the local neighborhood is $[c \pm (b - a)\varepsilon]$ for some $\varepsilon \geq 0$. In other words, the neighborhood is a scaled ε-ball. For discrete hyperparameters like batch-size or epochs drawn from a set $\{a, \dots, b\} \subset \mathbb{Z}$, the local neighborhood is $\{c - \lfloor (b - a)\varepsilon \rfloor, \dots, c + \lceil (b - a)\varepsilon \rceil\}$. In the experiments conducted on datasets Shakespeare, Femnist and Cifar that will be shown in Figure ε is set to 0.1.

The Algorithm 4 shows FedEx wrapped in SHA:

Algorithm 4 [13]: FedEx wrapped in SHA

```
Input: distribution $D$ over hyperparameters $A$, elimination rate $\eta \in N$, elimination rounds
$\tau_0 = 0, \tau_1, \ldots, \tau_R$, sample set of $\eta^R$ hyperparameters $H \sim D^{[\eta^R]}$
initialize a model $w_a \in \mathbb{R}^d$ for each $a \in H$
for elimination round $r \in [R]$ do
    for setting $a = (b, c) \in H$ do
        $s_a, w_a, \theta_a \leftarrow$ FedEx$(w_a, b, c, \theta_a, \tau_{r+1} - \tau_r)$
    $H \leftarrow \left\{a \in H: s_a \leq \frac{1}{\eta} - quantile(\{s_a: a \in H\})\right\}$
Output: remaining $a \in H$ and associated $w_a$
```

```
FedEx$(w, b, \{c_1, \ldots, c_k\}, \theta, \tau \geq 1)$:
Initialise $\theta_1 \leftarrow \theta$, initialise shared weights $w_1 \leftarrow w$
for communication round $t = 1, \ldots, \tau$ do
    for client $i = 1, \ldots, B$ do
        send $w_t$,$\theta_t$ to client
        sample $c_{ti} \sim \mathcal{D}_{\theta_t}$
        $w_{ti} \leftarrow Loc_{c_{ti}}(T_{ti}, w_t)$
        Send $w_{ti}$, $c_{ti}$, $L_{V_{ti}}(w_{ti})$ to server
    $w_{t+1} \leftarrow Agg_b(w, \{w_{ti}\}_{i=1}^{B})$
    Set step size $\eta_t$ and baseline $\lambda_t$
    $\nabla_j \leftarrow \frac{\sum_{i=1}^{B} |V_{ti}| \left(L_{V_{ti}}(w_{ti}) - \lambda_t\right) 1_{c_{ti}=c_j}}{\theta_{t[j]} \sum_{i=1}^{B} |V_{ti}|}, \forall j$
    $\theta_{t+1} \leftarrow \theta_t \odot \exp(-\eta \nabla)$
    $\theta_{t+1} \leftarrow \frac{\theta_{t+1}}{\|\theta_{t+1}\|_1}$
    $s_a \leftarrow \sum_{i=1}^{B} |V_{ti}| L_{V_{ti}}(w_i) / \sum_{i=1}^{B} |V_{ti}|$
Return $s$, model $w$, hyperparameter distribution $\theta$
```

Algorithm 4, because of the chalenges it handles, is considered a suitable algorithm for tuning the hyperparameters of deep neural network models trained in federated setting. Here are the results of training the FedAVG on datasets Shakespeare, next-character prediction dataset, Femnist, image classification dataset, and Cifar, image classification dataset from „Federated hyperparameter tuning: Challenges, baselines, and connections to weight-sharing":

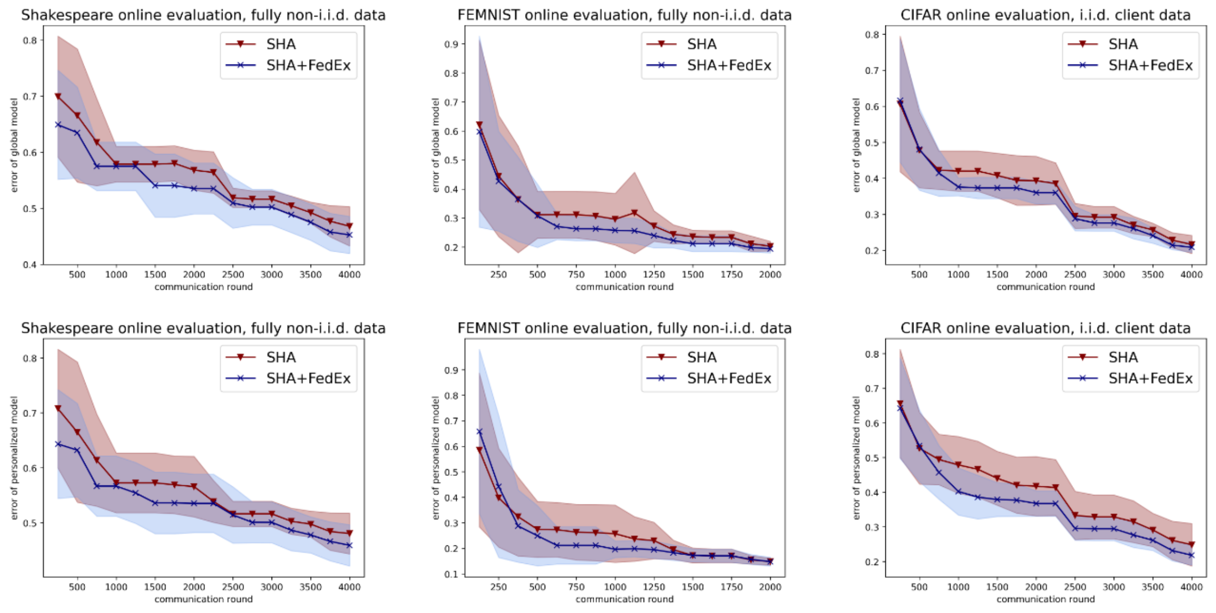


Figure 25: Evaluation of SHA and FedEx Wrapped in SHA on Shakespeare (left graphs), FEMNIST (middle graphs) and CIFAR (right graphs) Datasets of Global (top graphs) and Local (bottom graphs) Models [13]

Figure 25 represents model performance, namely the decline of the error that the model is making after the shown amount of communication rounds. Performance of the global model is represented in the three graphs on top and the personalised is displayed on the bottom three graphs. Left two graphs show error evaluation on Shakespeare, center two graphs are for Feminst evaluation and the right two graphs are for Cifar evaluation. The red line represents FedAVG preformance with hyperparemeters tuned with SHA and the blue line shows performance of FedAVG with hyperparemeters optimised with FedEx wrapped in SHA.

On every graph in Figure we observe that the blue line is below the red line, which means that the error is lower and it decreases faster for a model trained with combination of SHA and FedEx then model trained with SHA only. Therefore, we conclude that federated learning combined with FedEx wrapped in SHA shows signifficant improvement in performance compared to the performance of the same algorithm combined with just SHA.

Algorithm 4 is an excellent hyperparameter optimisation strategy for federated learning that can be used on most federated learning algorithms. On those algorithms that are not suited for FedEx, successive halving algorithm can be applied and it yields satisfactory results.

To summarise, our strategy for greener federated learning is to use synchronous federated learning with lower concurrency and to tune other hyperparameters of the model with FedEx wrapped with SHA, as it has been proven to be a great algorithm for federated setting. By doing so, we can optimise both our FL models' performance and energy efficiency, making it competitive with similar state-of-the-art models while using less electric energy and therefore emitting less carbon gases into the atmosphere.

4.3 Additional Strategies

We have seen how the wise choice of federated learning algorithm and its hyperparameters can benefit the environment by lowering the electricity usage and CO_2e emissions. These strategies are studied in multiple research papers [1,2,3] and represent a good way to tackle the challenges of green deep and federated learning.

However, there exist some energy-efficient strategies suitable only for federated setting whose impact on lowering carbon emissions could be huge in the future when they are properly evaluated. These strategies are model compression and quantisation.

In chapter 2 – Green Federated Learning, we saw that the internet connection of devices and their upload and download speed play a significant part in calculating the energy consumption of the overall training process. So, the downloading of the current global model and uploading the updated local model use a certain amount of electrical power that isn't negligible. For instance, it has been observed [2] that in some cases this communication between central server and devices could contribute up to 60% of total greenhouse gases emissions. This communication can be improved by compressing the model before it is sent via internet.

Compression and quantisation have been used in federated learning training to improve communication efficiency. Compression can be applied only on global model sent by central server and received by devices or only on local model sent by devices to the server or it can be applied to both. The model can be modified in many ways before being sent through the internet so it can get to the destination faster and so that the arrival of the model is almost guaranteed even with a really bad internet connection. It can be quantised, linearly transformed, rotated, subsampled, etc. It is important that it can be decompressed in the destination. These transformations are computations done by central server and end-devices for every round. We can assume that some of these computations use less energy and emit less CO_2e than the non-compressed model uses for traveling from one device to another via internet. It has been proposed in „Green Federated Learning“[2] that the compression strategy for improving communication efficiency can be energy efficient. However, the experiments that would show how much less energy is used in such communication than in standard communication and to which amount of CO_2e gases emitted it correlates to are yet to be conducted.

Good candidates for such experiments would be LFL (lossy federated learning) [15], QSGD (quantised SGD), sparse SGD, DoubleSqueeze [11], Federated Dropout [18] and similar compression solutions.

Reducing the carbon emissions of server-to-client and client-to-server communication would be beneficial in the future and this could benefit the federated setting the most because this communication represents the biggest difference, electrical energy consumption wise, between federated and centralised setting. Lowering the communication cost could make federated learning models’ energy consumption and CO_2e emissions more comparable to similar state-of-the-art models trained in the centralised setting.

5. Conclusion

Carbon emissions of federated learning represent a real threat for global warming. Federated learning can emit up to two orders of magnitude more carbon than centralised learning. It is important to evaluate CO_2e emissions for training large scale state-of-the-art models in order to prevent climate change.

We have seen that emissions of CO_2e of a particular FL learning model can be calculated by multiplying total electrical energy consumption of the training process of each client with a factor PUE that differs according to the clients' geolocation. Clients' hardware's efficiency is important for running FL tasks as well as their internet connection. However, these are the factors that mostly depend on the users of the model and not on the ones creating it. Similarly, distribution among devices (IID and non-IID) and the nature of the learning task (language learning, image classification, etc.) have an influence on emissions, but they usually cannot be altered. There are other factors that we have more control of and that should be optimised with respect to carbon emissions. We can choose synchronous over asynchronous federated learning since synchronous emits significantly less CO_2e. Also, we can choose hyperparameters in such a way to reduce the carbon footprint.

We concluded that concurrency (maximal number of clients that can train simultaneously) and time or rounds needed for the model to reach target accuracy are factors that impact the carbon footprint the most. With set concurrency and with an evaluation of training time we can predict the CO_2e emissions of the model's training since the emissions are linearly correlated with the product of concurrency and training time. Low concurrency corresponds to low carbon emissions, so this is the hyperparameter that should be kept at lower values in order to prevent the global warming. However, benefits of higher concurrency are faster convergence an increased model performance, although not scaled linearly. It is important to find the balance and optimise concurrency and other parameters of the model especially those with greater impact on training time.

Hyperparameters (batch sizes, learning rates, local epochs, concurrency etc.) of the FL model should be optimised with hyperparameter tuning methods. This is a proven way of reducing carbon footprint. We propose a combination of successive halving algorithm (SHA) and FedEx for hyperparameter optimisation. SHA is an algorithm that performs well on optimising deep learning models and FedEx is an algorithm made specifically for federated learning for tuning hyperparameters on the local models (models on users' devices). This is the method we advocate for because it tackles some of the greater challenges of hyperparameter tuning in the federated setting like federated validation data, limited computational capabilities of clients’ devices and evaluation of personalisation. This is a method made for federated setting and it performs in this setting better than other tuning methods.

At last, we conclude that the strategy for greener federated learning is to optimise hyperparameters of FL model, especially concurrency, with FedEx wrapped with SHA and to choose synchronous over asynchronous training.

This strategy can be improved in the future. The future research could mainly focus on lowering the electrical energy used to transmit the model via internet. We already know that model compression and quantisation are being used to lower the communication cost between clients and server, but there are still no experiments on whether compression and quantisation lower the carbon emissions. This is something that can be looked into and added to the strategy.

Bibliography

[1] Qiu X, Parcollet T, Fernandez-Marques J, Gusmao PP, Gao Y, Beutel DJ, Topal T, Mathur A, Lane ND. A first look into the carbon footprint of federated learning. Journal of Machine Learning Research. 2023;24(129):1-23.

[2] Yousefpour A, Guo S, Shenoy A, Ghosh S, Stock P, Maeng K, Krüger SW, Rabbat M, Wu CJ, Mironov I. Green federated learning. arXiv preprint arXiv:2303.14604. 2023 Mar 26.

[3] Xu J, Zhou W, Fu Z, Zhou H, Li L. A survey on green deep learning. arXiv preprint arXiv:2111.05193. 2021 Nov 8.

[4] Schwartz R, Dodge J, Smith NA, Etzioni O. Green ai. Communications of the ACM. 2020 Nov 17;63(12):54-63.

[5] Anderson K. The inconvenient truth of carbon offsets. Nature 2012; 484, 7.

[6] McMahan B, Moore E, Ramage D, Hampson S, y Arcas BA. Communication-efficient learning of deep networks from decentralized data. InArtificial intelligence and statistics 2017 Apr 10 (pp. 1273-1282). PMLR.

[7] Xu C, Qu Y, Xiang Y, Gao L. Asynchronous federated learning on heterogeneous devices: A survey. Computer Science Review. 2023 Nov 1;50:100595.Architectural Patterns for the Design of Federated Learning Systems

[8] Mammen PM. Federated learning: Opportunities and challenges. arXiv preprint arXiv:2101.05428. 2021 Jan 14.

[9] Wolff Anthony LF, Kanding B, Selvan R. Carbontracker: tracking and predicting the carbon footprint of training deep learning models. arXiv e-prints. 2020 Jul:arXiv-2007.

[10] Tang H, Yu C, Lian X, Zhang T, Liu J. Doublesqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression. InInternational Conference on Machine Learning 2019 May 24 (pp. 6155-6165). PMLR.

[11] Chen R, Li L, Xue K, Zhang C, Pan M, Fang Y. Energy efficient federated learning over heterogeneous mobile devices via joint design of weight quantization and wireless transmission. IEEE Transactions on Mobile Computing. 2022 Oct 11.

[12] Nguyen J, Malik K, Zhan H, Yousefpour A, Rabbat M, Malek M, Huba D. Federated learning with buffered asynchronous aggregation. InInternational Conference on Artificial Intelligence and Statistics 2022 May 3 (pp. 3581-3607). PMLR.

[13] Khodak M, Tu R, Li T, Li L, Balcan MF, Smith V, Talwalkar A. Federated hyperparameter tuning: Challenges, baselines, and connections to weight-sharing. Advances in Neural Information Processing Systems. 2021 Dec 6;34:19184-97.

[14] Konecný J, McMahan HB, Yu FX, Richtárik P, Suresh AT, Bacon D. Federated learning: Strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492. 2016 Dec 9;8.

[15] Amiri MM, Gunduz D, Kulkarni SR, Poor HV. Federated learning with quantized global model updates. arXiv preprint arXiv:2006.10672. 2020 Jun 18.

[16] Konečný J, McMahan HB, Ramage D, Richtárik P. Federated optimization: Distributed machine learning for on-device intelligence. arXiv preprint arXiv:1610.02527. 2016 Oct 8.

[17] Federated Learning, Min Du, Postdoc, UC Berkeley (slide-show)

[18] Caldas S, Konečny J, McMahan HB, Talwalkar A. Expanding the reach of federated learning by reducing client resource requirements. arXiv preprint arXiv:1812.07210. 2018 Dec 18.

[19] Bonawitz K, Eichner H, Grieskamp W, Huba D, Ingerman A, Ivanov V, Kiddon C, Konečný J, Mazzocchi S, McMahan B, Van Overveldt T. Towards federated learning at scale: System design. Proceedings of machine learning and systems. 2019 Apr 15;1:374-88.

[20] Henderson P, Hu J, Romoff J, Brunskill E, Jurafsky D, Pineau J. Towards the systematic reporting of the energy and carbon footprints of machine learning. Journal of Machine Learning Research. 2020;21(248):1-43.

[21] Li X, Huang K, Yang W, Wang S, Zhang Z. On the convergence of fedavg on non-iid data. arXiv preprint arXiv:1907.02189. 2019 Jul 4.

[22] Kundu A, Yu P, Wynter L, Lim SH. Robustness and personalization in federated learning: A unified approach via regularization. In2022 IEEE International Conference on Edge Computing and Communications (EDGE) 2022 Jul 10 (pp. 1-11). IEEE.

[23] McDonald R, Hall K, Mann G. Distributed training strategies for the structured perceptron. InHuman language technologies: The 2010 annual conference of the North American chapter of the association for computational linguistics 2010 Jun (pp. 456-464).

[24] Hu E, Tang Y, Kyrillidis A, Jermaine C. Supplemental Material for Federated Learning Over Images: Vertical Decompositions and Pre-Trained Backbones Are Difficult to Beat.

[25] Dimitriadis D, Ken'ichi Kumatani RG, Gmyr R, Gaur Y, Eskimez SE. A Federated Approach in Training Acoustic Models. InInterspeech 2020 Oct (pp. 981-985).

[26] Li L, Talwalkar A. Random search and reproducibility for neural architecture search. InUncertainty in artificial intelligence 2020 Aug 6 (pp. 367-377). PMLR.

[27] Brander M, Davis G. Greenhouse gases, CO2, CO2e, and carbon: What do all these terms mean. Econometrica, White Papers. 2012 Aug.

[28] Yu T, Zhu H. Hyper-parameter optimization: A review of algorithms and applications. arXiv preprint arXiv:2003.05689. 2020 Mar 12.

[29] Bergstra J, Bengio Y. Random search for hyper-parameter optimization. Journal of machine learning research. 2012 Feb 1;13(2).

[30] Allan RP, Arias PA, Berger S, Canadell JG, Cassou C, Chen D, Cherchi A, Connors SL, Coppola E, Cruz FA, Diongue-Niang A. Intergovernmental Panel on Climate Change (IPCC). Summary for Policymakers. InClimate change 2021: The physical science basis. Contribution of working group I to the sixth assessment report of the intergovernmental panel on climate change 2023 (pp. 3-32). Cambridge University Press.

[31] Ranasinghe R. Climate change 2021: Summary for all.

[32] Ebrahimi K, Jones GF, Fleischer AS. A review of data center cooling technology, operating conditions and the corresponding low-grade waste heat recovery opportunities. Renewable and sustainable energy reviews. 2014 Mar 1;31:622-38.

[33] Kontar R, Shi N, Yue X, Chung S, Byon E, Chowdhury M, Jin J, Kontar W, Masoud N, Nouiehed M, Okwudire CE. The internet of federated things (IoFT). IEEE Access. 2021 Nov 10;9:156071-113.

Biography

Nevena Đilas was born on 23rd of May 1997 in Vršac. In her hometown, she graduated primary school “Mladost” and then grammar school “Borislav Petrov Braca”, both with GPA 5.00. She competed in mathematics every year since third grade of primary school and following her passion for mathematics she acquired Bachelor’s degree in Theoretical Mathematics at Faculty of Sciences in Novi Sad in 2020 with GPA 9.11. Then, she enrolled in Data Science Master Studies at the same faculty.

UNIVERZITET U NOVOM SADU

PRIRODNO-MATEMATIČKI FAKULTET

KLJUČNA DOKUMENTACIJSKA INFORMACIJA

Redni broj: RBR	
Identifikacioni broj: IBR	
Tip dokumentacije: BF	Monografska dokumentacija
Tip zapisa: TZ	Tekstualni štampani materijal
Vrsta rada: VR	Master rad
Autor: AU	Nevena Đilas
Mentor: MN	dr Dušan Jakovetić
Naslov rada: NR	Strategija treniranja zelenog federativnog učenja
Jezik publikacije: JP	Engleski
Jezik izvoda: JI	Engleski
Zemlja publikovanja: ZP	Republika Srbija
Uže geografsko područje: UGP	Vojvodina
Godina: GO	2024
Izdavač: IZ	Autorski reprint

Mesto i adresa: MA	Novi Sad, Prirodno-matematički fakultet, Departman za matematiku i informatiku, Trg Dositeja Obradovića 4
Fizički opis rada: FO	5 poglavlja/58 strana/33 lit. citata/ 23 slike/ 2 tabele
Naučna oblast: NOS	Nauka o podacima
Naučna disciplina: ND	Primenjena matematika
Ključne reči: KR	Mašinsko učenje, duboko učenje, federativno učenje, veštačka inteligencija, optimizacija hiperparametara
Univerzalna decimalna klasifikacija: UDK	
Čuva se: ČU	Biblioteka Departmana za matematiku i informatiku Prirodnomatematičkog fakulteta, u Novom Sadu
Važna napomena: VN	
Izvod: IZ	Prvo poglavlje predstavlja uvod u temu. U drugom poglavlju se definiše šta je federativno učenje i koje vrste federativnog učenja postoje. Treće poglavlje se bavi uticajem treniranja neuronskih mreža na životnu sredinu i važnošću zelenog federativnog učenja. Četvrto poglavlje proučava faktore treniranja modela federativnim učenjem koji utiču na zagađenje životne sredine i predlaže metode i tehnike da se štetan uticaj minimalizuje. Peto poglavlje je zaključak.
Datum prihvatanja teme od strane NN veća: DP	5.7.2024.
Datum odbrane: DO	11.9.2024.
Članovi komisije: KO	Predsednik: dr Nikola Obrenović, naučni saradnik Instituta BioSens u Novom Sadu Mentor: dr Dušan Jakovetić, vanredni profesor Prirodno-matematičkog fakulteta u Novom Sadu Član: dr Oskar Marko, naučni saradnik Instituta BioSens u Novom Sadu

UNIVERSITY OF NOVI SAD

FACULTY OF SCIENCES

KEY WORDS DOCUMENTATION

Accession number: ANO	
Identification number: INO	
Document type: DT	Monograph type
Type of record: TR	Printed text
Contents code: CC	Master’s thesis
Author: AU	Nevena Đilas
Mentor: MN	Dr Dušan Jakovetić
Title: TI	Green Federated Learning Training Strategy
Language of text: LT	English
Language of abstract: LA	English
Country of publication: CP	Republic of Serbia
Locality of publication: LP	Vojvodina
Publication year: PY	2024

Publisher: PU	Author’s reprint
Publ. place: PP	Novi Sad, Department of Mathematics and Informatics, Faculty of Science and Mathematics, University of Novi Sad, Trg Dositeja Obradovića 4
Physical description: PD	6 chapters/58 pages /33 references/ 23 figures/2 tables
Scientific field: SF	Data Science
Scientific discipline: SD	Applied mathematics
Key words: KW	Machine learning, deep learning, federated learning, artificial intelligence, hyperparameter optimisation
Universal decimal classification: UDC	
Holding data: HD	The Library of the Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad
Note: N	
Abstract: AB	First chapter is an introduction to the topic. Second chapter defines what federated learning is and what types of federated learning exist. Third chapter deals with the impact of training neural networks on the environment and the importance of green federated learning. The fourth chapter studies model training with federated learning factors that affect environment pollution, proposing methods and techniques to minimise the harmful impact.The fifth chapter is the conclusion.
Accepted by the Scientific Board on: ASB	5.7.2024.
Defended: DE	11.9.2024.
Thesis defend board: DB	Chairperson: Dr. Nikola Obrenović, Research Associate, BioSense Institute, Novi Sad Mentor: : Dr Dušan Jakovetić, associate professor, Faculty of Sciences, University of Novi Sad

	Member: Dr. Oskar Marko, Research Associate, BioSense Institute, Novi Sad