When Federated Learning Meets Neural Architecture Search: A Comparison

Caíque S. Lima[1], **Rafael C. Ito**[1]

[1]Faculdade de Engenharia Elétrica e de Computação
Universidade Estadual de Campinas (Unicamp)
Av. Albert Einstein, 400 – Cidade Universitária Zeferino Vaz
Distrito Barão Geraldo – Campinas-SP – Brasil

{[redacted]}`@dac.unicamp.br`

Abstract. *The traditional way of training a machine learning model considers that both data and model are available at the same place (called centralized model). However, there are plenty of cases in which this statement is not true, specially if we consider the edge devices. Moreover, there are situations in which besides having the data divided across several devices, sharing them is not an option, due to privacy concerns. In this context, a recent research area called Federated Learning (FL) aims to target this problem by training a single model in a distributed fashion. At the same time, another research area called Neural Architecture Search (NAS) tries to solve a different problem of finding the neural network for a given problem in an automated fashion, instead of hand-crafting its design. In this work, we attempt to provide a full comparison of both these areas and the relationships between them. We compare the scenarios of centralized and federated learning, each of them making use or not of a NAS framework. The source code used to train the models presented in this project is available at* `https://github.com/ito-rafael/MO809A-FederatedLearning/tree/main/project/code`.

Resumo. *A forma tradicional de se treinar um modelo de aprendizado de máquina considera que ambos dados e modelo estão disponíveis no mesmo local (chamado de modelo centralizado). No entanto, existem muitos casos em que essa afirmação não é verdadeira, especialmente se considerarmos os dispositivos de borda. Além disso, há situações em que além de ter os dados divididos em vários dispositivos, compartilhá-los não é uma opção, devido a questões de privacidade. Nesse contexto, uma área de pesquisa recente chamada Aprendizado Federado (FL) visa abordar esse problema treinando um único modelo de maneira distribuída. Ao mesmo tempo, outra área de pesquisa chamada Neural Architecture Search (NAS) tenta resolver um problema diferente de encontrar a rede neural para um determinado problema de maneira automatizada, ao invés de criar seu design manualmente. Neste trabalho, tentamos fornecer uma comparação completa de ambas as áreas e as relações entre elas. Comparamos os cenários de aprendizado centralizado e federado, cada um deles fazendo uso ou não de um framework de NAS. O código-fonte utilizado para treinar os modelos apresentados neste projeto está disponível em* `https://github.com/ito-rafael/MO809A-FederatedLearning/tree/main/project/code`.

1. Introduction

The 21st century has undoubtedly become the data age. Powered by the expansion of the Internet and the emergence of concepts such as the so-called Fourth Revolution (e.g. Industry 4.0, Health 4.0), disruptive innovations have increasingly gained space on the global stage. These new digital tools integrate various innovations such as Internet of Things (IoT), Internet of Bodies (IoB), Big Data and Machine Learning (ML) that have resulted in advances in digital health, smart manufacturing and self-driving cars.

In this context, the essential part is the data. The information that flows between the countless entities that make up these intelligent systems has a significant value, to the point that data can already be considered the new oil of this century [Bhageshpur 2019]. On a philosophical realm, data can also be considered an emerging ideology or even a new form of religion, some authors describe the great data revolution as *dataism*, as discussed by [Harari 2017].

In a scenario where data has a high commercial value while being highly sensitive, new concerns about privacy and data governance have arisen. Many efforts have been made towards the complexity of collecting, curating and maintaining a high-quality dataset, while meeting regulatory, ethical and legal challenges. These data-driven efforts have been identified as one of the great challenges in healthcare area, for example [Rieke et al. 2020].

In the context of healthcare, the data produced in this environment has been a bottleneck since health data is highly sensitive and its usage is tightly regulated [van Panhuis et al. 2014]. One of the biggest challenges regarding health data is the issue of patient privacy, as even the anonymization process of data may not be sufficient to preserve privacy [Rocher et al. 2019]. Another problem is information leakage that can cause great harm to both patients and healthcare providers.

Faced with these issues and also in other domains, a learning paradigm emerges seeking to address the problem of data governance and privacy by training algorithms collaboratively without exchanging the data itself. This new approach is known as Federated Learning (FL). FL allows training a global model without moving data beyond the firewalls of the local units in which they reside. Instead, the ML process occurs locally at each participating federated entity and only model characteristics (e.g., parameters, gradients) are transferred as depicted in Figure 1.

Mathematically, a general formulation of FL reads as follows: let ℓ denote a global loss function obtained via a weighted combination of K local losses $\{\ell_k\}_{k=1}^{K}$, computed from private data X_k, which is residing at the individual involved parties and never shared among them:

$$\underset{\phi}{min}\, \ell\left(X; \phi\right) \text{ with } \ell\left(X; \phi\right) = \sum_{k=1}^{K} \left[w_k\, \ell_k\left(X_k; \phi\right)\right], \tag{1}$$

where $w_k > 0$ denote the respective weight coefficients. In practice, each federation unit (training node) typically obtains and refines a global consensus model by performing a few rounds of optimization locally and before sharing updates, either directly or through a parameter server.

As a contribution to the study of FL, this paper compares the performance of four

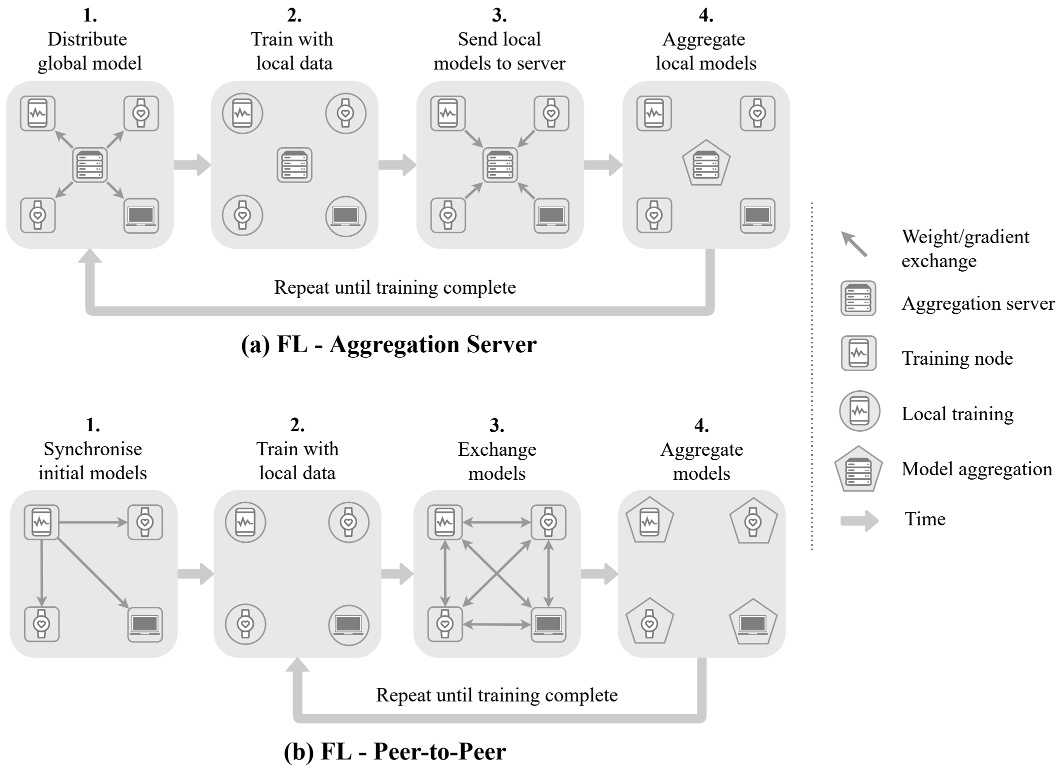
1. Distribute global model
2. Train with local data
3. Send local models to server
4. Aggregate local models
Repeat until training complete
(a) FL - Aggregation Server
1. Synchronise initial models
2. Train with local data
3. Exchange models
4. Aggregate models
Repeat until training complete
(b) FL - Peer-to-Peer
Weight/gradient exchange
Aggregation server
Training node
Local training
Model aggregation
Time

Figure 1. Example of federated learning (FL) workflows. (a) FL aggregation server — typical FL workflow in which a federation of local units (training nodes) receive the global model, locally trains the models using the local data, resubmit their partially trained models to a central server intermittently for aggregation and then continue training on the consensus model that the server returns. (b) FL peer-to-peer — alternative formulation of FL in which each local unit exchanges its partially trained models with some or all of its peers and each does its own aggregation, without the central server. Adapted from [Rieke et al. 2020].

ML approaches — centralized `DenseNet`, centralized `MiLeNAS`, Federated Averaging (`FedAvg`) and Federated Neural Architecture Search (`FedNAS`) — over a non-IID (non identical and independent distribution) image dataset based on CIFAR10 deployed in a distributed computing environment consisting of 10 clients and 1 server for the `FedAvg` approach, 5 clients and 1 server for the `FedNAS` approach, and a single node for the centralized approaches. The following section 2 summarizes previous works relevant to the FL context in order to establish the link between existing knowledge in this learning paradigm and new findings. The rest of this paper is organized as follows: section 3 presents the techniques and materials used in this study, section 4 describes how the experiments were set up and their respective performances, and section 5 concludes the findings achieved in this work.

2. Related Works

Neural Architecture Search (NAS) is one of the three subfields of the AutoML (automated machine learning) research area, alonside with Hyperparameter Optimization and Meta-learning [Hutter et al. 2019]. The main idea is to automatically search for the neural architecture (number of layers, operations, number of channels, etc.), instead of hand-crafting it. There are several strategies in NAS that tries to automate the design of the neural network for a given problem, such as reinforcement learning [Zoph and Le 2017], evolutionary algorithms [Real et al. 2017], gradient descent techniques [Liu et al. 2018], random search [Li and Talwalkar 2019], and many others.

Recently, a few works started combining both fields of NAS and Federated Learning. [Zhu et al. 2020] did a survey with a brief review in both research areas and the relationships between them. [Zhu and Jin 2020] performed NAS in a federated learning paradigm using evolutionary algorithms. [Yuan et al. 2022] presented a federated NAS approach considering the cross-device scenario, while [He et al. 2021] did the same, but for a cross-organization scenario (cross-silo). Our work is more close to the last one, but instead of proposing a different federated NAS method, we aim to make a comparison between centralized and federated scenarios, considering or not the automated search of the neural networks with NAS.

The Federated Averaging (`FedAvg`) — which will be properly discussed in detail in subsection 3.2.3 — is an effective yet simple algorithm that is most commonly used for federated aggregation. The `FedAvg` aggregation is the main method to improve model performance in FL domain, it consists of equal distribution of model parameters for every local model [Jiang et al. 2020].

3. Materials and Methods

This section gives a detailed account of the procedures as well as the materials and techniques used in this work. Subsection 3.1 presents the dataset used in the experiments and the applied approaches are described in subsection 3.2.

3.1. Dataset

Aiming to investigate the performance of each ML approaches assessed here, a collection of labeled images that are commonly used to train ML and computer vision models was used, the CIFAR-10. This dataset contains 60,000 color images pre-processed into 32×32 (2D) with the corresponding classification labels. The ten different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships and trucks. Figure 2 displays 10 random samples of each class present in the database.

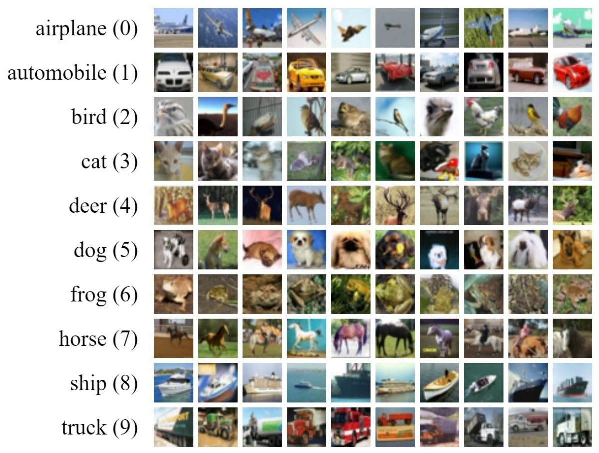

Figure 2. Random images samples of CIFAR-10 dataset.

The CIFAR-10 dataset is subdivided by default into two subsets: 50,000 training images and 10,000 test images. To evaluate the models, we generate non-IID dataset by splitting the 50,000 training images into K clients in an unbalanced manner: sampling $p_c \sim Dir_J(0.5)$ and allocating a $p_{c,k}$ proportion of the training samples of class c to local client k. The remaining 10,000 test images are used for a global test after the aggregation of each round.

3.2. Approaches

Four approaches were compared during the experiments, two in a centralized way — `DenseNet` and `MiLeNAS` — and the remaining two in a federated context — `FedAvg (DenseNet)` and `FedNAS (MiLeNAS)`. Table 1 below summarizes which techniques were applied (or not) Neural Architecture Search and Federated Learing.

	NAS	FL
`DenseNet`	✗	✗
`MiLeNAS`	✓	✗
`FedAvg (DenseNet)`	✗	✓
`FedNAS (MiLeNAS)`	✓	✓

Table 1. Comparison between the evaluated approaches.

3.2.1. Centralized `DenseNet`

The first approach considered is the centralized paradigm without using NAS to search for the architecture. The fixed architecture used here is the `DenseNet` [Huang et al. 2016]. This choice was made since this neural network presents decent results in computer vision tasks reported in the literature, and also to be able to compare the results with the one made by [He et al. 2021], since they used this same architecture.

The `DenseNet` architecture is an extension of the ResNet [He et al. 2015], a neural network that won the `ImageNet` [Deng et al. 2009] competition in 2015, that connects each layer to every other layer of the network. Figure 3 illustrates the architecture of the `DenseNet`.

3.2.2. Centralized `MiLeNAS`

The centralized approach but using NAS to find the final architecture used the `MiLeNAS` [He et al. 2020b] framework. This framework is similar to the `DARTS` framework [Liu et al. 2018]. The difference is that while `DARTS` use the training set to train the network weights and the validation set to train the architecture parameters separately, the `MiLeNAS` still uses the training set for the network weigths, but also incorpored the training set to train the architecture parameters. That is, they used both the training and validation set to find the architecture parameters. This leads to improvements in both search time and final accuracy, as reported by the authors. Figure 4 shows a flowchart demonstrating the `MiLeNAS` framework. As usual in the NAS works, w represents the neural network weights and α represents the architecture parameters.

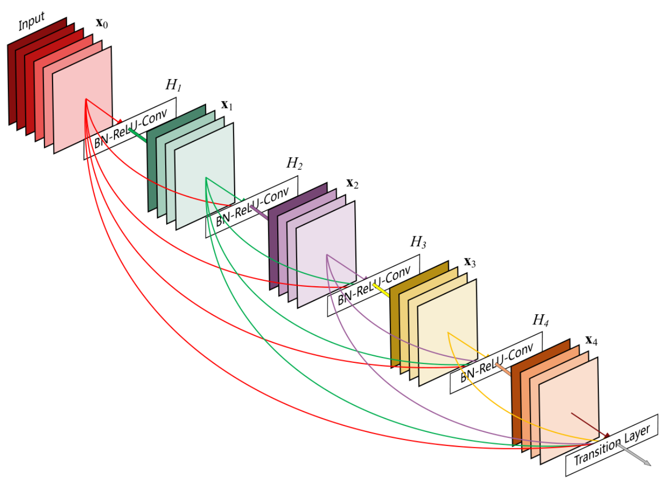

Figure 3. DenseNet architecture: each layer is connected to every other layer of the network. Figure taken from [Huang et al. 2016].

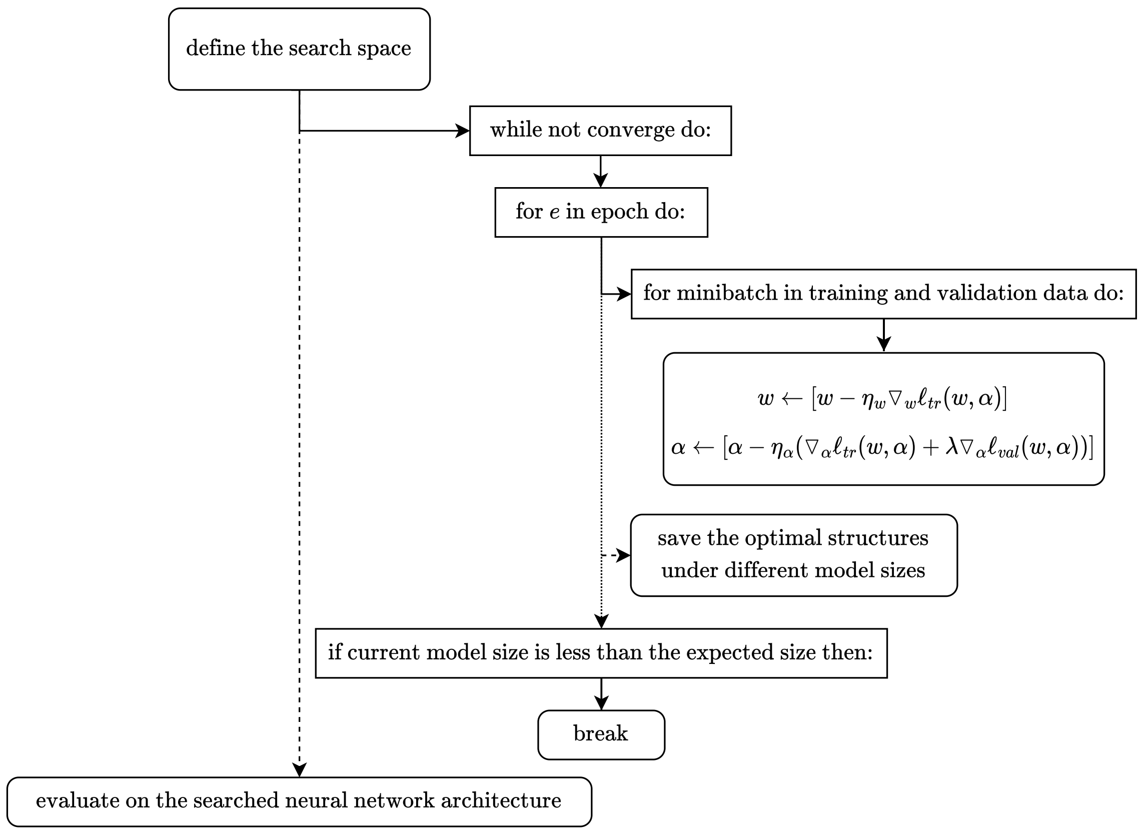
define the search space
while not converge do:
for e in epoch do:
for minibatch in training and validation data do:
$w \leftarrow [w - \eta_w \nabla_w \ell_{tr}(w, \alpha)]$
$\alpha \leftarrow [\alpha - \eta_\alpha (\nabla_\alpha \ell_{tr}(w, \alpha) + \lambda \nabla_\alpha \ell_{val}(w, \alpha))]$
save the optimal structures under different model sizes
if current model size is less than the expected size then:
break
evaluate on the searched neural network architecture

Figure 4. `MiLeNAS` algorithm proposed by [He et al. 2020b].

3.2.3. Federated Averaging (`FedAvg`)

Federated Averaging, popularly known as `FedAvg`, is a communication efficient algorithm proposed by [Brendan McMahan et al. 2017] for the distributed training with a huge number of clients (training nodes), such as smartphones, smartwatches, autonomous vehicles and hospitals. In `FedAvg`, client data is kept locally for privacy protection, and a central parameter server is used for communication between the training nodes. This central server distributes parameters to each client and collects updated parameters from clients. The clients achieve communication efficiency by making multiple local updates of a shared global model before sending the result to the central server, which averages the locally updated models to aggregates the next global model [Collins et al. 2022]. The `FedAvg` algorithm workflow is depicted in Figure 5 below.

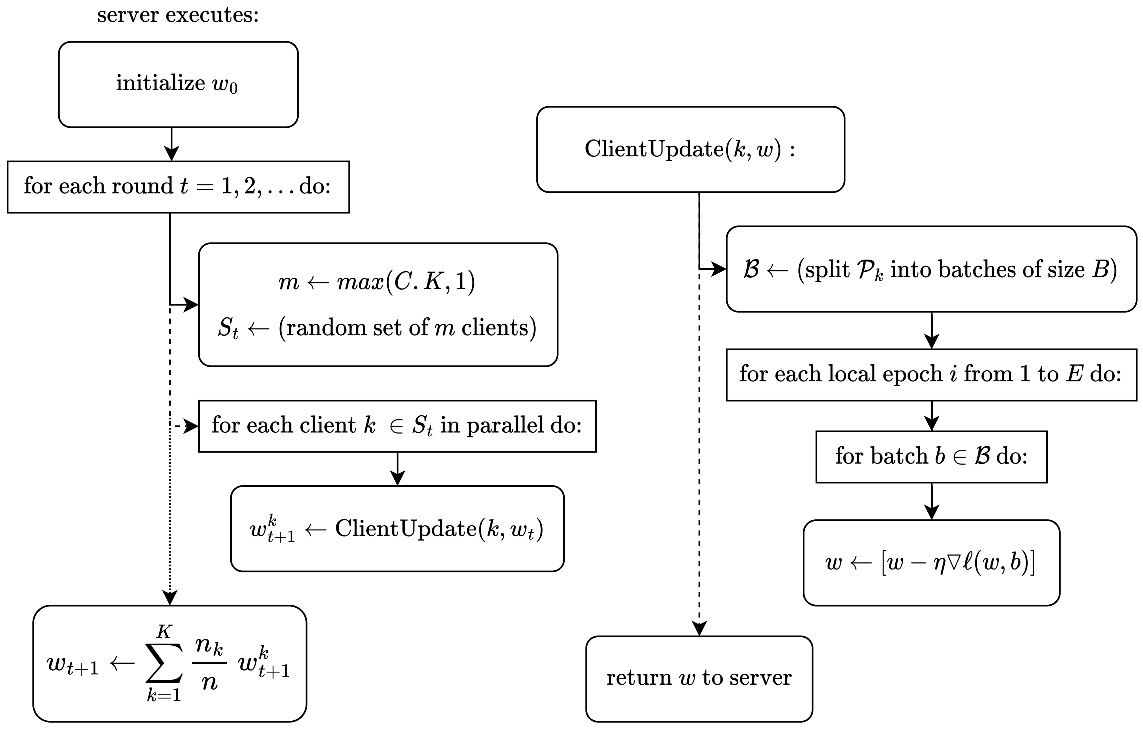
server executes:
initialize w_0
for each round $t = 1, 2, \ldots$ do:
$m \leftarrow max(C.K, 1)$
$S_t \leftarrow$ (random set of m clients)
for each client $k \in S_t$ in parallel do:
$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
ClientUpdate(k, 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:
$w \leftarrow [w - \eta \nabla \ell(w, b)]$
return w to server

Figure 5. `FedAvg` algorithm proposed by [Brendan McMahan et al. 2017]. The K clients are indexed by k, B is the local minibatch size, E is the number os local epochs and η is the learning rate.

3.2.4. Federated Neural Architecture Search (`FedNAS`)

The Federated Neural Architecture Search (`FedNAS`) is a distributed neural architecture search algorithm proposed by [He et al. 2020a] for automating the model design process in the FL context. According to the authors, `FedNAS` allows for a collaborative search for a better architecture to achieve better performance. In addition to automating and improving FL model design, `FedNAS` also provides a new paradigm for custom FL via customizing not only the model weights (w) but also the neural architecture (α) of each user.

In the FL setting, there are K local nodes in the network. Each node has a private dataset $\mathcal{D}_k := \left\{ \left(x_i^k, y_i \right) \right\}_{i=1}^{N_k}$ wich is non-IID. When collaboratively training a model with K nodes, the objective function is defined as follows:

$$\min_{w} f(w, \underbrace{\alpha}_{fixed}) \overset{def}{=} \min_{w} \sum_{k=1}^{K} \frac{N_k}{N} \cdot \frac{1}{N_k} \sum_{i \in \mathcal{D}_k} \ell(x_i, y_i; w, \underbrace{\alpha}_{fixed}) \tag{2}$$

where w denotes the network weight, α determines the neural architecture and ℓ is the loss function of the model. To minimize the objective function above, previous works choose a fixed model architecture α then design variant optimization techniques to train the model w. On the other hand, `FedNAS` proposes to optimize the FL problem from a completely different angle, optimizing w and α simultaneously. Thus, the objective function in this scenario is reformulated as:

$$\min_{w,\alpha} f(w,\alpha) \overset{def}{=} \min_{w,\alpha} \sum_{k=1}^{K} \frac{N_k}{N} \cdot \frac{1}{N_k} \sum_{i \in \mathcal{D}_k} \ell(x_i, y_i; w, \alpha) \tag{3}$$

The process performed by the `FedNAS` is summarized in the flowchart shown in the Figure 6.

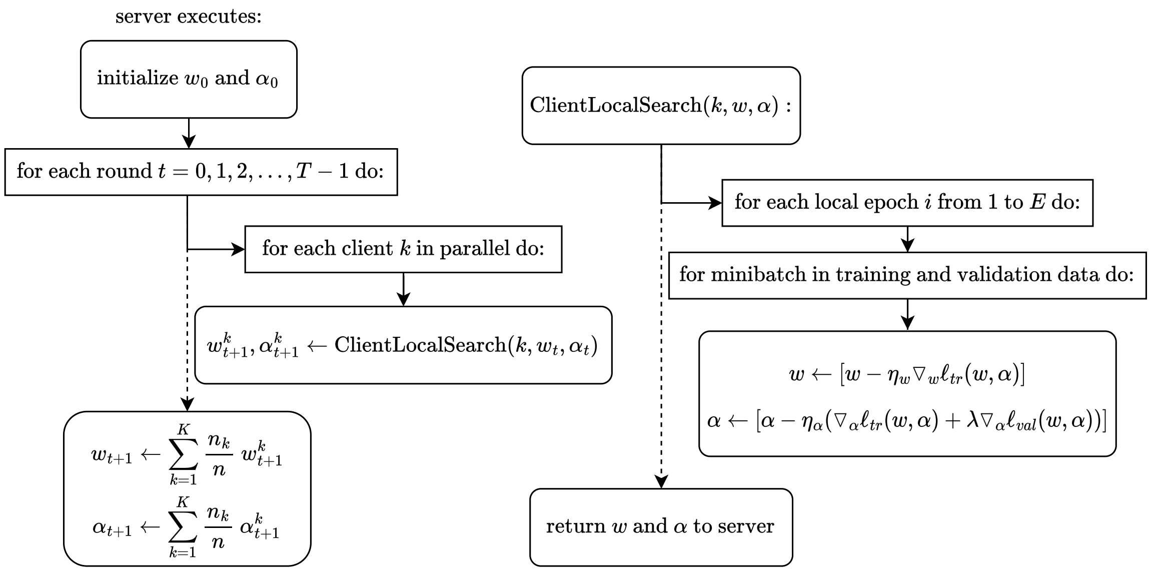
server executes:
initialize w_0 and α_0
for each round $t = 0, 1, 2, \ldots, T-1$ do:
for each client k in parallel do:
$w^k_{t+1}, \alpha^k_{t+1} \leftarrow$ ClientLocalSearch(k, w_t, α_t)
$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w^k_{t+1}$
$\alpha_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} \alpha^k_{t+1}$
ClientLocalSearch(k, w, α) :
for each local epoch i from 1 to E do:
for minibatch in training and validation data do:
$w \leftarrow [w - \eta_w \nabla_w \ell_{tr}(w, \alpha)]$
$\alpha \leftarrow [\alpha - \eta_\alpha (\nabla_\alpha \ell_{tr}(w, \alpha) + \lambda \nabla_\alpha \ell_{val}(w, \alpha))]$
return w and α to server

Figure 6. `FedNAS` algorithm proposed by [He et al. 2020a]. The K clients are selected and indexed by k, E is the number of local epochs, T is the number of rounds.

4. Experiments and Results

In this section, details about the experiments done for the four approaches discussed in section 3.2 are given. All experiments were done in Python using the PyTorch framework. For the federated approaches, we used the Flower framework for the implementation of the server and the clients, and a non-IID data distribution was adopted. The training parameters were the same of [He et al. 2020b], since the base of the code used was the same.

The system was architectured to be run in a Docker container. Inside the container, the code for all four approaches can be found. For the centralized versions, the training procedure is the conventional, with a single Python code running the training. For the federated versions though, the code is split into the server and the clients. While the server runs only once and contains information about the aggregation method and the number of epochs for each round, for example, the clients are launched by a shell script, that defines the total number of clients being considered in the experiment and the portion of the dataset allocated for each client. Figure 7 depicts this architecture.

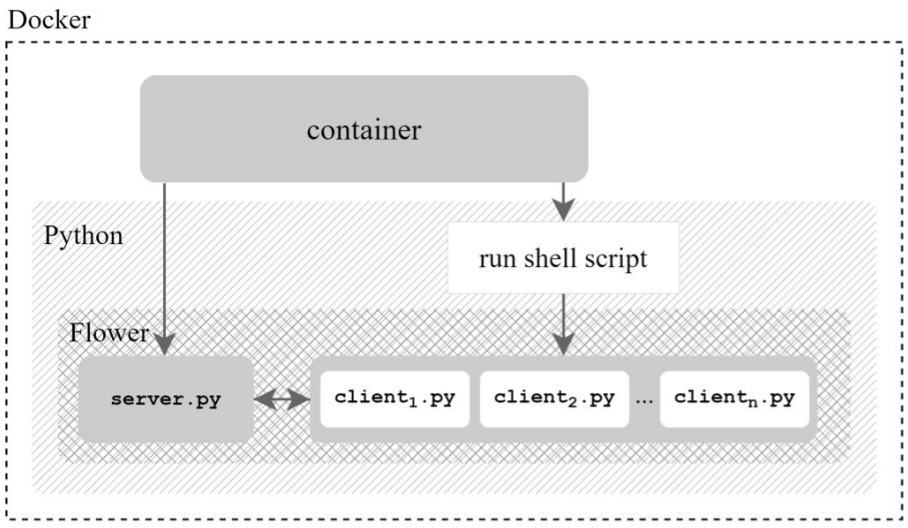

Figure 7. System design.

4.1. Centralized `DenseNet`

This first experiment used the fixed `DenseNet`-201 architecture and performed the conventional training. In total, three runs were done for this scenario. The first run resulted in a final accuracy of 83.44% top-1 accuracy and took 4h02min. The second run achieved 84.53% accuracy taking 4h13min to be trained. The last run got 83.53% and 3h48min. All runs were done in a single RTX 2080 Ti GPU with 11 GB of memory.

4.2. Centralized `MiLeNAS`

The second approach also used a single RTX 2080 Ti GPU with 11 GB of memory. In these experiments, instead of fixing the neural network a priori, the `MiLeNAS` framework was used to search the final architecture. Three runs were also done: the first one achieved 89.10% top-1 accuracy under the test set and took 10h31min, the second run resulted in 88.60% accuracy with a total time of 11h51min, and the last one got 88.80% accuracy taking 12h13min to train.

4.3. Federated Averaging (`FedAvg`)

In this federated scenario the data needed to be divided into the number clients. Here we considered a total of 10 clients with data divided in a non-IID fashion. Table 2 shows the data distribution, where k is the ID of the client and C_n represents the data of class n in the CIFAR-10 dataset. Note that the total amount of data is 50,000, equally divided in all classes (5,000 samples per class).

Again, we did three runs for this setup. The GPU used in these experiments was a single Quadro RTX 8000 with 48 GB of memory. The number of clients was 10 and all clients participated of all rounds. Each client trained considering only its own data for 10

	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	Total
k=0	2538	2206	409	3154	40	178	387	128	2363	0	11403
k=1	43	215	202	1423	1777	2276	1	498	446	878	7759
k=2	23	1517	1244	192	2349	208	259	2175	682	114	8763
k=3	1053	1004	2473	0	108	512	897	1057	1368	2583	11055
k=4	1343	58	672	231	726	1826	3456	1142	141	1425	11020
total	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000	50000

Table 2. Data distribution for the 10 clients in the federated scenario without NAS.

epochs per round. The total training lasted 100 rounds and the aggregation method used by the server was the `FedAvg`.

The first run resulted in a final accuracy of 57.54% top-1 accuracy under the test set and took 6h34min to accomplish the training. The second run achieved 56.83% accuracy taking 7h17min to finish. The last run got 56.23% accuracy in 6h50min.

4.4. Federated Neural Architecture Search (`FedNAS`)

In this last federated scenario, each client ran in its own GPU, since they would not fit in a single GPU, due to memory limitation. The data was then divided into five different clients. Again we considered a non-IID data distribution that can be seen in Table 3.

	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	Total
k=0	953	142	141	75	695	819	0	2482	0	0	5307
k=1	16	43	902	1650	86	182	0	693	110	6	3688
k=2	9	8	290	769	841	283	119	1044	1014	58	4435
k=3	395	1200	48	68	896	681	90	17	1351	301	5047
k=4	504	2917	570	721	121	356	0	0	0	0	5189
k=5	1262	71	325	119	1560	14	1	85	366	0	3803
k=6	9	273	1657	40	1	130	1911	21	1160	0	5202
k=7	722	3	281	738	22	974	624	1	82	878	4325
k=8	1127	153	680	500	698	1139	92	23	1	3665	8078
k=9	3	190	106	320	80	422	2163	634	916	92	4926
Total	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000	50000

Table 3. Data distribution for the 5 clients in the federated scenario using NAS.

With this configuration only a single run was done, due to the relative high demanding for computational resources. Each client was trained in a RTX 2080 Ti with 11 GB of memory with its own data, and trained for one epoch in each round. The total training lasted the same 100 rounds as the previous setup, but the aggregation method used by the server was the `FedNAS`. That is, it used the `FedAvg` for the network weights and a similar aggregation for the architecture parameters. The final top-1 accuracy was of 83.77% and the total training took 19h27min.

Figure 8 shows the results for each of the previous scenarios described. The left part of the figure shows the results of the centralized approaches, with the red curve being the `DenseNet` and the green curve being the architecture found by the `MiLeNAS` framework. The right side of the figure illustrates the results obtained for the federated approaches, with the blue curve being the federated `DenseNet` and the pink curve being the federated `MiLeNAS` approach. Note that the y-axis of both plots represents the top-1 percentage accuracy under the teste set of CIFAR-10, while the x-axis represents the number of epochs for the centralized approaches and the number of rounds for the federated approaches.

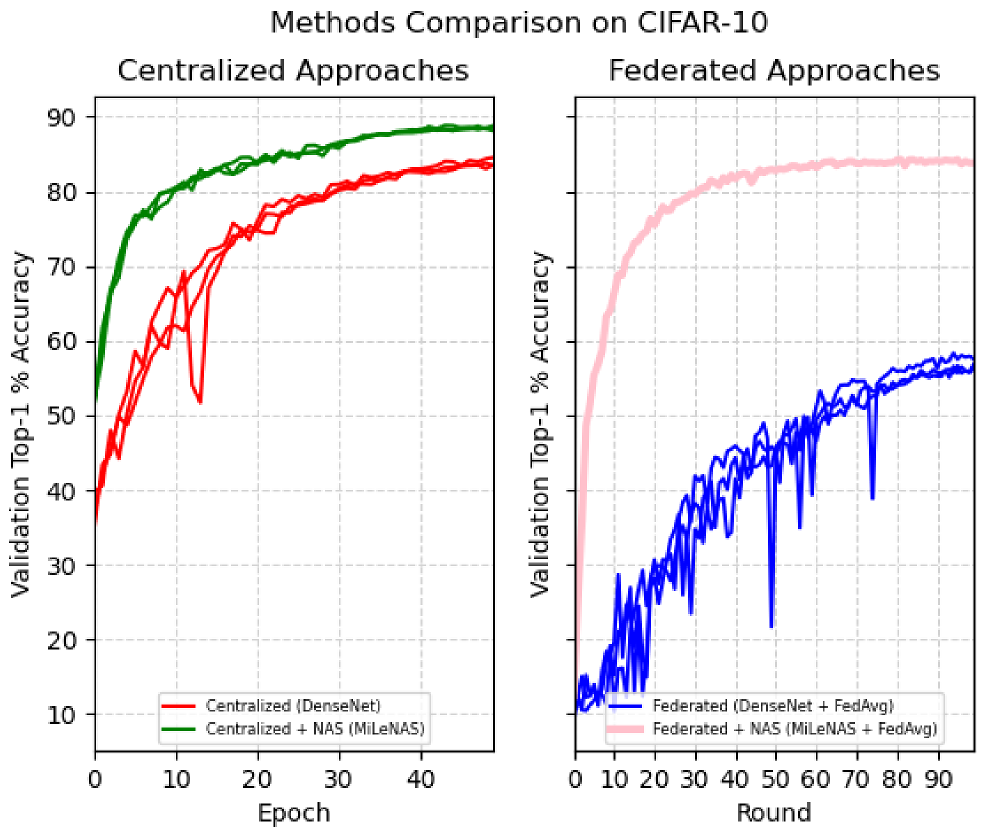

Figure 8. Comparison between all four approaches considered in the experiments.

5. Conclusion

Comparing centralized and federated learning may not be fair, since in federated approaches the data are divided across the clients and there are potential obstacles to the training, such as non-IID data and the longer time for convergence. Saying that, there are still some insights that we can take from the results obtained of the experiments. As we can see in Figure 8, searching for the final architecture using a NAS framework instead of using a fixed one can help finding for better architectures. In both centralized and federated approaches, the architecture found by the `MiLeNAS` framework achieved a final better accuracy and also reached that in less time than the approaches with a fair fixed architecture (`DenseNet`). Furthermore, the `FedNAS` aggregation method for both network weights and architecture parameters helps to make the training more stable if we compare both the federated approaches. Finally, if the dataset is available all in one place, we may expect a centralized approach to perform better. However, if the data is divided across many clients and data privacy is a concern, it is simply not possible to share information between the nodes. Then, the federated learning paradigm arrives to solve this problem of training a single model in this challenging scheme. Additionally, associating NAS techniques to both centralized and federated scenarios can help to improve the efficiency of the machine learning pipeline by finding better architectures in less time.

References

Bhageshpur, K. (2019). Data is the new oil – and that's a good thing. Available at: `https://www.forbes.com/sites/forbestechcouncil/2019/11/15/data-is-the-new-oil-and-thats-a-good-thing/?sh=6c1df8d97304`. Access on: 07 Nov. 2022.

Brendan McMahan, H., Moore, E., Ramage, D., Hampson, S., and Agüera y Arcas, B. (2017). Communication-Efficient learning of deep networks from decentralized data. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1273–1282.

Collins, L., Hassani, H., Mokhtari, A., and Shakkottai, S. (2022). FedAvg with fine tuning: Local updates lead to representation learning.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Harari, Y. N. (2017). *Homo Deus: A Brief History of Tomorrow*. Harper, 1st edition.

He, C., Annavaram, M., and Avestimehr, S. (2020a). FedNAS: Federated Deep Learning via Neural Architecture Search. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Workshop on Beyond Neural Architecture Search*.

He, C., Annavaram, M., and Avestimehr, S. (2021). Towards Non-I.I.D. and Invisible Data with FedNAS: Federated Deep Learning via Neural Architecture Search.

He, C., Ye, H., Shen, L., and Zhang, T. (2020b). Milenas: Efficient neural architecture search via mixed-level reformulation. *CoRR*, abs/2003.12238.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

Huang, G., Liu, Z., and Weinberger, K. Q. (2016). Densely connected convolutional networks. *CoRR*, abs/1608.06993.

Hutter, F., Kotthoff, L., and Vanschoren, J., editors (2019). *Automated Machine Learning: Methods, Systems, Challenges*. The Springer Series on Challenges in Machine Learning. Springer International Publishing.

Jiang, J. C., Kantarci, B., Oktug, S., and Soyata, T. (2020). Federated learning in smart city sensing: Challenges and opportunities. *Sensors*, 20(21).

Li, L. and Talwalkar, A. (2019). Random Search and Reproducibility for Neural Architecture Search.

Liu, H., Simonyan, K., and Yang, Y. (2018). DARTS: differentiable architecture search. *CoRR*, abs/1806.09055.

Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y. L., Tan, J., Le, Q. V., and Kurakin, A. (2017). Large-Scale Evolution of Image Classifiers. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2902–2911. PMLR.

Rieke, N., Hancox, J., Li, W., Milletarì, F., Roth, H. R., Albarqouni, S., Bakas, S., Galtier, M. N., Landman, B. A., Maier-Hein, K., Ourselin, S., Sheller, M., Summers, R. M., Trask, A., Xu, D., Baust, M., and Cardoso, M. J. (2020). The future of digital health with federated learning. *NPJ Digital Medicine*, 3:119.

Rocher, L., Hendrickx, J. M., and de Montjoye, Y.-A. (2019). Estimating the success of re-identifications in incomplete datasets using generative models. *Nature Communications*.

van Panhuis, W. G., Paul, P., Emerson, C., Grefenstette, J., Wilder, R., Herbst, A. J., Heymann, D., and Burke, D. S. (2014). A systematic review of barriers to data sharing in public health. *BMC Public Health*.

Yuan, J., Xu, M., Zhao, Y., Bian, K., Huang, G., Liu, X., and Wang, S. (2022). Federated Neural Architecture Search.

Zhu, H. and Jin, Y. (2020). Real-time federated evolutionary neural architecture search. *CoRR*, abs/2003.02793.

Zhu, H., Zhang, H., and Jin, Y. (2020). From federated learning to federated neural architecture search: A survey. *CoRR*, abs/2009.05868.

Zoph, B. and Le, Q. (2017). Neural Architecture Search with Reinforcement Learning.