Aggregation algorithms use in Federated Learning for smart buildings

This paper comes from a student project lead by BRAHMIA Mohamed-El-Amine

Hoang Donovan
CESI École d'Ingénieurs
donovan.hoang@viacesi.fr

Amary Clément
CESI École d'Ingénieurs
clement.amary@viacesi.fr

Antoni Luc
CESI École d'Ingénieurs
luc.antoni@viacesi.fr

Blum Adeline
CESI École d'Ingénieurs
adeline.blum@viacesi.fr

***Abstract*—Smart Buildings are more and more common in new or renovated cities, and their data are often open-source. Artificial Intelligence and Internet of Things are often used in this domain, in order to optimize the building's environment and metrics and lower its energy consumption. Federated Learning is more and more used in this context, but needs to be tested further to determine its efficiency and which algorithm suits the best. In this paper, we will test several Federated Learning algorithms on an open-source dataset coming from a Smart Building, in order to compare their results.**

***Index Terms*—Artificial intelligence; Federated learning; Aggregation; Energy consumption; Smart buildings**

I. Introduction

Nowadays, energy consumption and saving are the most important problems in order to preserve our comfort of life without degrading the Earth. In order to answer this problem, Artificial Intelligence in pair with Internet of Things seems a perfect solution because it is an autonomous solution that learns from our behavior easily. Which is perfect in the context of building smart cities that will adapt to inhabitants and the environment based on collected data. Also, privacy is very important regarding our way of life and how we live in our residence. In this context, Federated Learning meets all our expectations. Indeed, models are trained locally and then transfer to a central server without sending raw data and are aggregated to create a global model that will update local models at the end of the learning round. In this paper we will try to develop an efficient aggregation approach in the case of Smart building. The aim of the use of FL is to reduce electricity consumption by influencing building parameters while ensuring user's comfort.

II. State of the Art

Federated Learning is completely new approach of learning and is still in development. We started by reviewing the most recent and relevant work for our study and we will briefly recap them in this section.
They are several types and algorithms, which implements Federated Learning (FL in the rest of the document) in different ways, and for different purposes. We started by learning about **Federated Personalization (FedPer)**. With this method, the edge nodes will send only the base layers (base layers) of their models to the central server, but will keep the other layers as "personalized layers". The advantages of this approach are that personalized layers will be confidential and not transmitted to a central server. According to [1], FedPer may have some advantages with local model that generalize well. However, it is mostly not the case with our local models (because we encounter several climatic conditions, depending on the localization of the Smart Building), so this algorithm is not relevant to our objective. Another type of FL is **Federated Match Averaging (FedMA)**: [1] claims that FedMA has better results than FedPer, based on a Human Activity Recognition training. FedMA construct the global model layer by layer, by matching the different local model layers with similarity. FedMA is also a subcategory of a type of FL, which is **Federated Averaging (FedAvg)**. This model is way more spread and known, which is an important criteria for our researches, since we are looking to implement the algorithm in the most efficient way. And, according to [1], FedAvg gives better results in terms of accuracy, compared to FedPer and FedMA. Federated Averaging (FedAvg) will act as an average function to aggregate the local models and create a global model, and mainly used in the field of HAR, but is also mainly used for Internet of Things. As said before, FedAvg has several subcategories, and one of them is **FedProx**. This algorithm is based on FedAvg, but it introduces some improvements in the tolerance of partial work and proximal term. These results are presented in a more realistic approach of devices' work capacity based on their system resources. [2] concluded that the FedProx algorithm can improve the convergence of federated learning for heterogeneous networks with higher performance as FedAvg algorithm.
Another approach would be to implement **Federated Distance (FedDist)**, because the study [3] proves that FedDist gives the advantage of evolving the global model compared to FedAvg, and will produce better results in this case. The FedDist approach will calculate the Euclidean distance between each neuron of the aggregated model and the local models, then find the divergent neurons and add them to the global models to create a new aggregated model. We have enough data at our disposal to build a Neuronal Network.

$$O(N * log(N)) \quad (1)$$

which is a great upgrade compared to usual algorithms that have a quadratic complexity O(N^2). It also solves the problem of the user or sensor dropout. TurboAggregate experimentally achieves a total running time that grows almost linear in the number of users, and provides up to 40× speedup. It can be used with both centralised and decentralised architectures.
Algorithm performance can vary depending on the local models, used devices, and collected data in the federated system. We used all the results we found in the studies quoted before to make a comparison of the different approaches in the Table I.

Algorithm	Advantages
Turbo Aggregating	O(N*log(N)) complexity Manage device dropout
FedProx	Efficient on heterogeneous environments Takes into account the capacities of the device
FedAvg	Good results on the HAR field
FerPer	Personalized layers will stay confidential FedPer is efficient with model that generalize
FedMA	Not as efficient as FedAvg and FedPer
FedDist	Better results as FedAvg Evolving global model

TABLE I
TABLE OF COMPARISON OF THE DIFFERENT OPERATIONS AND ADVANTAGES OF THE DIFFERENT FL APPROACH

Comparing the different algorithms and the results that are shown in the paper, we would choose three algorithms to test and compare in the context of energy prediction. The first one would be the **FedAvg** algorithm. Indeed, it is the first and basic algorithm and results have shown good performances and beat most of the algorithms. The second one is **FedDist** because it is a very new algorithm, and it is the only one that beat FedAvg. Finally, the last one is **Turbo Aggregating** because it reduces the complexity to O(N*log(N)) which is a huge gap compared to O(N^2).

III. EXPERIMENTS

In this section we will describe how we implemented these three federated learning algorithms, and the results we obtained. The final purpose is to compare the results and establish if, in the case of smart buildings, one algorithm shows better results in terms of efficiency. The main purpose is to predict the energy consumption of the building. We will use multiple metrics to compare them; scores, loss, MAE, MSE, RMSE, and total duration. The following explanations will concern the results of our work and you can find the source code this implementation in our GitHub repository (branch main) [4]

Moreover, we can make some hypothesis of results regarding of the previous work in this fields :

- FL may occurs decrease of the users comfort if local models which are extremely different being aggregated (activity hours, temperatures preferences, etc...), for example in the case of different building. But can be advantageous in the case of one unique building (same building materials, etc...)
- FL may give a lot of advantages in the field of IoT data sources, because of the divergence of values depending of the different IoT devices location
- If the dataset doesn't allow us to create local clients which has clear differences of values, the FL will not shows good results

A. Data wrangling

To test our algorithms, we retrieved data from the open-source datasets CU-BEMS [5], smart building electricity consumption and indoor environmental sensor datasets. This seven-floors building located in Bangkok, Thailand, was ideal because the dataset is split between 7 floors, each floor being used as a local model in our case. We have at our disposition several metrics, i.e. individual air conditioning consumption (in kW), lighting load (in kW), indoor temperature. Each entry of this dataset contains the all this sensors and consumptions values minutes by minutes for 2 years. We may change or transform the data entry by hours, but we choose to avoid this process because of the loss of information (means of temperatures, etc...)
Because of the differences between the types of data as well as the number of columns between some floors, we had to only use data from floor 5 to 7.We kept 5 floors (3-7) for our experiments, because they were the most complete. In order to make the dataset usable to train models, we processed the dataset as follows :

- Filling missing values with average
- Sum separates energy consumption into one unique value
- Convert non-numerical values (DateTime to unix epoch)
- Scale values

The following Figure 1 describes the different approaches of research we have made :

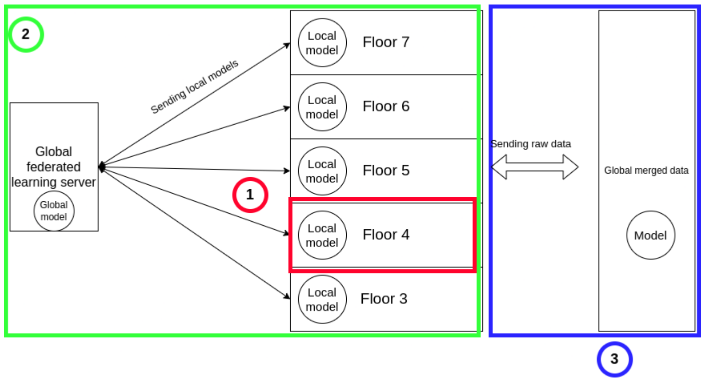

Fig. 1. FL architecture for CU-BEMS building [5]

- 1 - Local learning : Train models floor by floor only with their local data
- 2 - Federated learning : Train local model and aggregate them into a global one and return it to local client (floor)

- 3 - Centralized learning : Merging all floor's data and train one model

Each solution has its advantages and inconveniences. Local models are highly adapted to their own floors, and doesn't demand too many resources. However, the model is not adaptive. If the floor is encountering a new situation, it will take some time (days, sometimes weeks) for it to be trained with the actual situation. It is an issue centralized learning is correcting ; although the main problem is the privacy of the data, because all data from each floors are sent to the server. Federated learning ensures the data privacy, and the adaptiveness, but it requires a significant computing power, and it takes some time to train the global model using each local models' parameters.

B. Training local models

Before trying to aggregate local model, we have simulated a FL architecture by training different models floor by floor, one floor representing a local client. Each client will next send their model to the global server, which will process an aggregated model and send it back to each client as shown in the following.

The aims of each model is to predict the energy consumption regarding of other parameters such as the percentage of humidity, quantity of light and temperature.

We have tested the following learning models, to determine the one which the most suitable with the smart building data :

- Decision Tree
- Gradient Boosting
- K-Neighbors
- Linear regression
- MLP
- Random Forest
- SGD

C. Local models comparison

After training different models for each floor, you can find the following results of the floor 3 in the Figure 2:

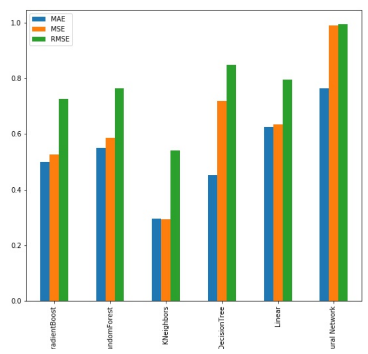

Fig. 2. MAE, MSE, RMSE Comparison for the floor 3 training

In this situation, we can see that KNeighbors is the best algorithm for training local models when we only want to train our AI with unique floor data without aggregate models for improvements, because we want to minimize MSE, MAE and RMSE as much as we can. Other floors are confirming the choice of KNeighbors. All the local training run results are stored in CSV files for analyse and compare with the aggregated models run.

D. Train on centralized models

We also aimed to prove the advantage of the use of FL against Centralized Learning. A global dataset has been made by merging all the 5 floors data into one dataset. We then ran each learning model previously tested on local models, in order to compare the results. The results of centralized learning approach in the following Figure 3

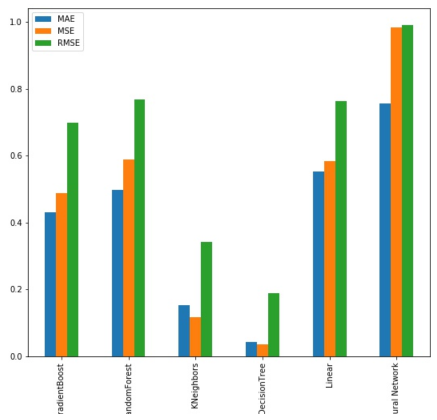

Fig. 3. MAE, MSE, RMSE Comparison for the centralized learning

According to our results, the best algorithm for centralized model is the **DecisionTree**: it has the best scores in terms of MAE, MSE and RMSE. The second best algorithm is KNeighbors, which was also very efficient for the local models. All other algorithm have much higher errors.

E. FL-algorithms implementation

Now that the local models are trained, we can focus on implementing Federated Learning Algorithms. It is important to precise that Federated Learning is based on Deep Learning. Thus, our local models are trained using MLP for the federating part. Because FL needs many rounds to get a global model which generalize well, we had to split each local training dataset (floor) by the number of wanted rounds so we simulate the new fresh new data for FL round. This feature makes learning easier for local training due to the reduction of data quantity (less time of learning processes) but may appears loss of generalization if uses too badly.
We used the floors 3-6 to train our local models (each floor), so a total of 4 local clients. The last floor (floor 7) is used after the aggregation, to test the global aggregated model on a new dataset and ensure that it is not over-fitted.

We started by implementing **FedAvg**, using the logic presented in the Algorithm 1. Here, we are trying to improve the Machine Learning Model by passing the best parameters retrieved from the local models to train the Federated Model.

Algorithm 1 FederatedAveraging. We have K clients indexed by k, L the local model size, E the number of local epochs, μ the learning rate, C a constant.

```
**ServerSide**:
initialize $w_0$
**for** each round $t$ **do**
    $m \leftarrow max(C * K, 1)$ + indentation
    $S_t \leftarrow$ set of $m$ clients + indentation
    **for** each client $k \in S_t$ **do**
        $w_{t+1}^k \leftarrow$ ClientUpdate($k, wt$)
    **end for**
    $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
**end for**

**ClientUpdate**($k, w$):
$L \leftarrow$ (split $P_k$ into batches of size $L$)
**for** each local epoch $i$ from 1 to $E$ **do**
    **for** batch $b \in L$ **do**
        $w \leftarrow w - \mu \nabla l(w; b)$
    **end for**
    **return** $w$ to server
**end for**
```

We also implemented the **FedDist** algorithm. To calculate the Euclidean distance between each neuron, we used the logic presented in the Algorithm 2.

Algorithm 2 FederatedDistance. We have m our matrix and M the global matrix.

```
$b \leftarrow M$
$distances, newMatrix = []$
$i = 0$
**for** $eleminM$ **do**
    $p_b \leftarrow b[i]$
    $distances$ += euclidian distance between $elem$ and $p_b$
    $elem = elem * distances[i]$
    $newMatrix$ += $elem$
**end for**
**return** $NewMatrix$
```

We can now compare the result of our two Federated-Learning algorithms. You can see on the Figure 4 that the two algorithms have close results, and overlaps each other. The better MAE score was obtained by **FedAvg** while training on the Floor 5. FedDist also obtained a higher maximum error than FedAvg (up to 3,9, vs. 3,2 for FedAvg), according to our results.
We can say that we obtained better results overall using FedAvg, but the differences are small.

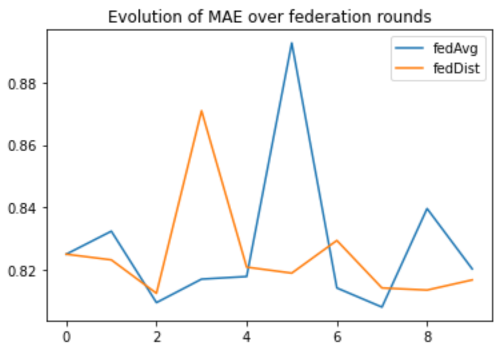

Fig. 4. MAE Comparison between FedDist and FedAvg

F. *FL-algorithms vs local models*

The next step was to re-run the global models for each floor (local dataset) and compare the new results with the local models run. The following Figures 5, 6 and 7 show you the values of MAE, RMSE and Max error for each approach of learning :

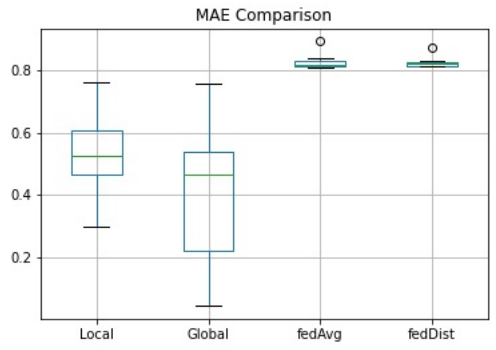

Fig. 5. MAE Comparison between approachs

Regarding MAE, we can see that the Federated Learning approach is not better than keeping local models as they were trained.

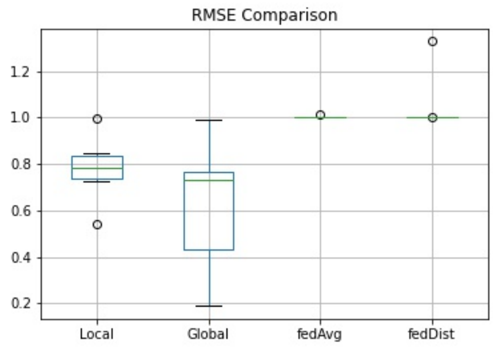

Fig. 6. RMSE Comparison between approachs

The same conclusion can be made for RMSE, except that it reach the maximum boundaries of the classic local model training.

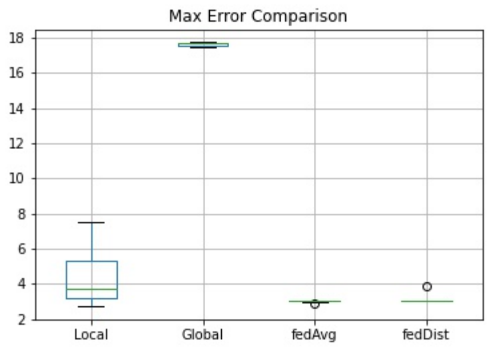

Fig. 7. Max error Comparison between approachs

As we can understand with this results, FedAvg and FedDist seems to give advantages regarding to Max error and theses two are equivalent even if we can notice minimal better result for FedAvg. Thus, Federated Learning seems not the best for our use case. Nevertheless, it can be due to our dataset which was not very heterogenous. Or simply because these two Federation algorithm were not the best strategy in our case.

IV. Conclusion

To conclude, theses researches were made in order to find an optimal way to use federated learning with the aims of reducing energy consumption in Smart Building. Many studies claimed that FedAvg is the best suitable algorithm for now. We focused our research on finding the best aggregation algorithm in our case even if we studied the efficiency of different learning model for local learning. We based our study on a public dataset [5] containing sensor's data (humidity, lighting, temperature) and energy consumption for a building, and assume that each floor was a FL client. The result on local training showed that the KNeighbor model was the best regarding different metric (MAE, RMSE, Max error).

But most important is the results of the global aggregated model with different approaches: FedAvg and FedDist. It showed that in our case, the two approaches were almost equivalent with a little advantage to FedAvg.

However, due to the dataset, the results may be not as accurate as we need (similar values between each floor so there is no real need to aggregate theses local models). So this study could be replicated with a more complex dataset (motion sensors, advanced IoT devices, ...) and maybe the new implementation of FL algorithm.

To go further, we could have implemented the **Turbo Aggregating** [6], which is also a relevant algorithm in our study case, because it reduces the complexity . It could be interesting to compare our results to the Turbo Aggregating (in the same conditions, using the same database), and establish conclusions about the more efficient and accurate algorithm between these three.

Acknowledgment

We would like to thank BRAHMIA Mohamed-El-Amine, who led the project and our work, answering our questions and gave us important resources to guide our work.
We also give a thanks to searchers from two papers [1] [3], who showed us their implementation of FL algorithm, which helped us to understand the process of model aggregation.

References

[1] S. Ek, F. Portet, P. Lalanda, and G. Vega, "Evaluation of Federated Learning Aggregation Algorithms Application to Human Activity Recognition," pp. 638–643, Sep. 2020. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02941944

[2] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated Optimization in Heterogeneous Networks," *arXiv:1812.06127 [cs, stat]*, Apr. 2020, arXiv: 1812.06127. [Online]. Available: http://arxiv.org/abs/1812.06127

[3] F. Portet, P. Lalanda, and G. Vega, "A Federated Learning Aggregation Algorithm for Pervasive Computing: Evaluation and Comparison," *2021 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 1–10, Mar. 2021, arXiv: 2110.10223. [Online]. Available: http://arxiv.org/abs/2110.10223

[4] D. HOANG, C. AMARY, L. ANTONI, and A. BLUM, "Federated-learning-for-smart-building source code," May 2022. [Online]. Available: https://github.com/Donovan1905/federated-learning-for-smart-building

[5] M. Pipattanasomporn, G. Chitalia, J. Songsiri, C. Aswakul, W. Pora, S. Suwankawin, K. Audomvongseree, and N. Hoonchareon, "Cu-bems, smart building electricity consumption and indoor environmental sensor datasets," *Nature News*, Jul 2020. [Online]. Available: https://www.nature.com/articles/s41597-020-00582-3

[6] J. So, B. Güler, and A. S. Avestimehr, "Turbo-Aggregate: Breaking the Quadratic Aggregation Barrier in Secure Federated Learning," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 1, pp. 479–489, Mar. 2021, conference Name: IEEE Journal on Selected Areas in Information Theory.