École polytechnique fédérale de Lausanne

Semester project spring 2024

Mechanical Engineering

Federated Learning for Data Streams

Author:
Tristan Carruzzo

Supervisor:
Mahrokh Ghoddousiboroujeni
Professor:
Giancarlo Ferrari Trecate

June 7, 2024

Contents

List of Figures

List of Tables

1 Introduction

Nowadays, with the increasing amount of data generated by IoT devices, finding meaningful ways to use this data has become crucial. As more devices come online and generate continuous streams of information, it is essential to develop machine learning algorithms that can process and analyze this data efficiently while preserving privacy and security. One promising approach is federated learning, which leverages decentralized data to train machine learning models collaboratively, and online learning, which allows models to update dynamically as new data arrives. These methodologies are particularly relevant in scenarios where data privacy is a significant concern, and data is generated continuously, such as in smart homes or industrial IoT applications.

Federated learning is a machine learning approach where multiple clients (e.g., devices or organizations) collaboratively train a model while keeping their data localized. Instead of sending their raw data to a central server, each client trains a model on its local dataset and only shares model updates (e.g., gradients or weights) with the central server. The central server then aggregates these updates to form a global model, which is sent back to the clients. This process helps preserve data privacy since the raw data never leaves the client's environment.

Online learning is a paradigm where the model is continuously updated as new data becomes available. Unlike traditional batch learning, which trains the model on a fixed dataset, online learning processes data in real-time or in small batches, allowing the model to adapt to new information dynamically. This is particularly useful for data streams, where data is generated continuously, and the model needs to remain up-to-date to make accurate predictions.

The goal of the project is to evaluate the performance of federated learning for data streams. This could prove very useful in a context where multiple clients are continuously generating data, which cannot be shared due to privacy concerns. In this project, the data represents the power generated by solar panels. Each client, in this case a household with solar panels, generates data hourly. The goal is to obtain a model capable of predicting the solar power generated for the next month and to understand how to make use of the data streams to improve the model.

Different training scenarios will be tested to evaluate the model's performance under various conditions, such as training on historical data, using fresh data, or combining both. The results will provide insights into the effectiveness of federated learning for data streams and the benefits of continuous model updates in dynamic environments.

2 Litterature Review

Federated learning is a decentralized machine learning approach that enables multiple clients to collaboratively train a model without sharing their raw data. This privacy-preserving technique has gained significant attention in recent years due to its potential to address data privacy concerns while leveraging the collective intelligence of distributed datasets. Various FL algorithms have been proposed to optimize the model training process across multiple clients, such as Federated Averaging (FedAvg), Federated Proximal (FedProx), and Federated Optimization (FedOpt). These algorithms aim to balance the trade-offs between model performance, communication efficiency, and privacy preservation.

While FL has shown promise in various applications, its effectiveness in scenarios with continuously generated data streams remains an open research question. Traditional FL frameworks are designed for static datasets, where clients contribute their data at the beginning

of the training process. However, in dynamic environments with data streams, such as IoT devices or online platforms, new data is continuously generated, requiring models to adapt in real-time. This poses challenges for existing FL algorithms, which may not be optimized for handling streaming data.
The paper titled Federated Learning for Data Streams [1] by Othmane Marfoq and colleagues introduces a novel approach to training machine learning models on continuously generated data streams from multiple clients. This method addresses the limitations of traditional federated learning frameworks, which typically rely on static datasets collected before training begins. These static approaches can be inefficient as they ignore new data generated during the training process and require a long preparatory phase for data collection. Furthermore, they may be impractical in scenarios where devices have limited storage capacity.
The main contribution of Marfoq et al.'s paper is the introduction of a bias-optimization trade-off, which is controlled by balancing the importance of older samples against newer ones. At each training step, clients receive batches of data comprising both historical and fresh samples. The model is then trained on this mixed data, with a weight parameter that balances the losses from the two types of data. The objective is to minimize the weighted sum of these losses, optimizing the model by considering both past and present data.
In more detail, the algorithm proposed in the paper allows each client to update its local model using a weighted empirical risk minimization. This process incorporates the different times that data samples spend in memory and their varied usage during training. By adjusting the weights of these samples, the algorithm manages the bias-optimization trade-off: increasing the weight of newer samples can speed up the training but might introduce bias, while giving more weight to older samples can reduce bias but slow down the training process.
The theoretical analysis provided in the paper offers insights into how to configure this federated learning algorithm effectively. The authors demonstrate through simulations across various machine learning tasks that their approach outperforms traditional federated learning methods adapted to data streams, particularly those extending the FedAvg algorithm. This highlights the significance of their configuration rules in achieving near-optimal performance in learning from data streams.

3 Methodology

3.1 Dataset

The dataset used for the different scenarios is a simulated dataset created using the Python library *pvlib* [2], which specializes in simulating solar power generation. This library leverages raw weather data to model the power output of solar panels. This weather data is sourced from the Photovoltaic Geographical Information System (PVGIS), a comprehensive database providing solar radiation and temperature data. For this project, the dataset represents solar power generated in Lausanne, Switzerland, spanning hourly data from 2005 to 2020.
To simulate real-world conditions, the dataset includes data for multiple households, each equipped with its own inverter and solar module. Variations are introduced by altering the orientation and tilt angle of the solar panels, as well as by adding slight variations in the weather data, reflecting the natural diversity in installation and local weather conditions.
In line with common practices in handling time series data for solar energy, nighttime data is excluded since solar panels do not generate power during the night. For the purpose of

this dataset, nighttime is defined as the period from 6 pm to 8 am. This leaves 10 usable data samples per day, focusing on the periods when solar power generation occurs. This preprocessing step ensures that the model is trained and evaluated on relevant data, enhancing its predictive performance and reliability.
The dataset is composed of 10 features. They are a mix of wheather information and temporal information. The wheather information includes temperature, wind speed, and sun height. The temporal information are the hour of the day and the day of the year. There is also the possibility to add autoregressor features, which are the power generated by the solar panel at the previous hours. In this project, only 1 autoregressor feature is used. Lastly, the target variable is the power generated by the solar panel during the next hour.

Data Visualization

Figure 1 shows the distribution of the power generated by the solar panels for one household over the years and over the months with standard deviation. These plots show the variability in the data, indicating the need for a robust model capable of capturing these patterns. We can also observe the seasonal trends in power generation, with higher values in the summer months and lower values in the winter months. Winter months also show a lower variability compared to spring and summer months. Over the years, there does not seem to be a clear trend in the power generation and the standard deviation is relatively stable.

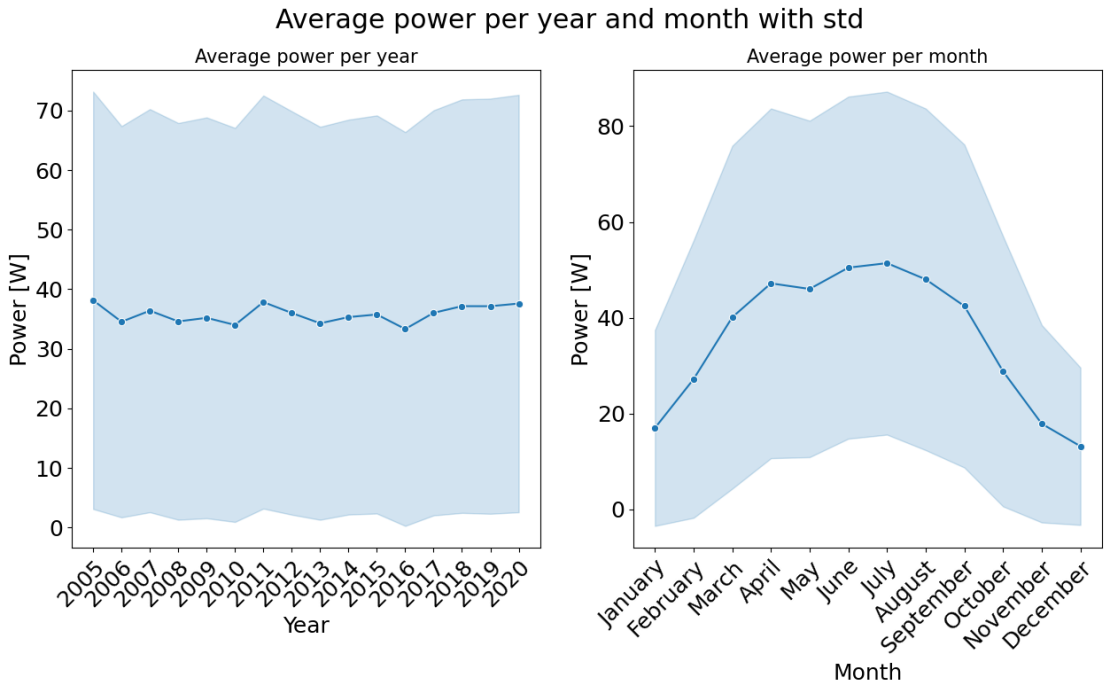

Figure 1: Power generation for one household.

Figure 2 shows the distribution of the temperature and wind speed for the same household over the months. We observe clear seasonal trends in temperature, with higher values in the summer months and lower values in the winter months. The wind speed, on the other hand, is more stable over the months, with similar variability throughout the year.

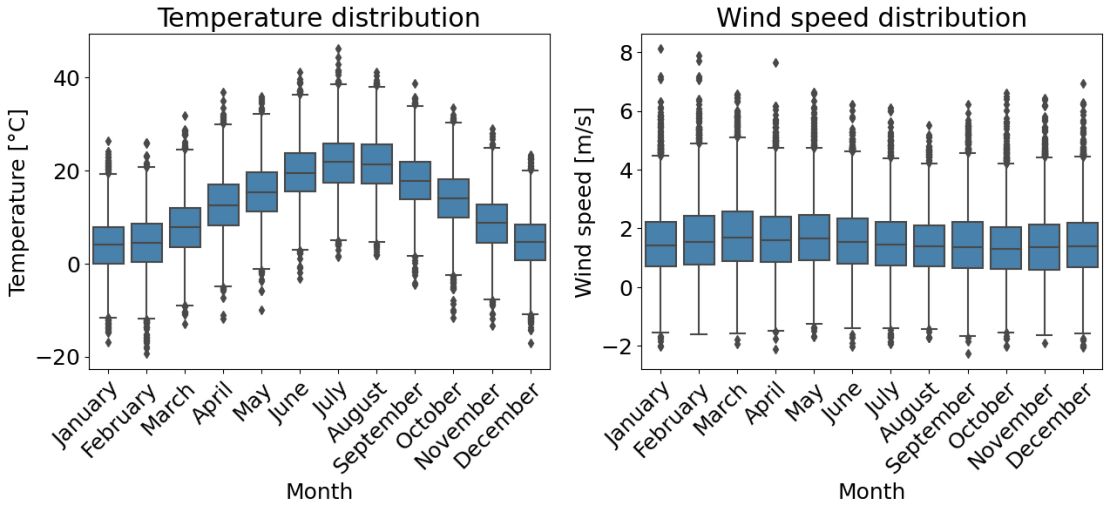

Figure 2: Temperature and wind speed for one household.

For the federated learning part of the project, the data is generated for multiple households. Each of these household can have a different PV setup, which will result in different power generation values. Table 1 shows the difference in the setup of the solar panels for 5 households. Tilt represents the angle of the solar panel with respect to the horizontal plane, azimuth is the angle of the solar panel with respect to the north, and altitude is the height of the solar panel above the ground in meters.
In order to have a meaningful real-world dataset, the variation in the weather features was setup as to have relatively different values for each household. The power is not only influenced by the setup of the solar panels, but also by the weather conditions, in particular the cloud cover. This parameter adds a random factor to the irradiation values of some clients, which will greatly impact the power generated by the solar panels.

House	Inverter	Module	Tilt	Azimuth	Altitude
0	ABB MICRO 25I 208V	Advent Solar AS160 2006	43.81	179.57	482.02
1	ABB MICRO 25I 240V	Advent Solar Ventura 210 2008	44.07	187.57	454.10
2	ABB MICRO 3I 208V	Advent Solar Ventura 215 2009	43.33	183.35	581.88
3	ABB MICRO 3I 240V	Aleo S03 160 2007E	54.53	178.61	542.03
4	ABB MICRO 3HVI 208V	Aleo S03 165 2007E	46.80	180.79	539.56

Table 1: Setup of the solar panels for 5 households.

For the Federated Learning experiements, the number of clients is set to 40. Figure 3 shows the distribution of the PV panels setup for the 40 clients used in the FL experiements. These variation in setup will influence the power generated by the PV panels.

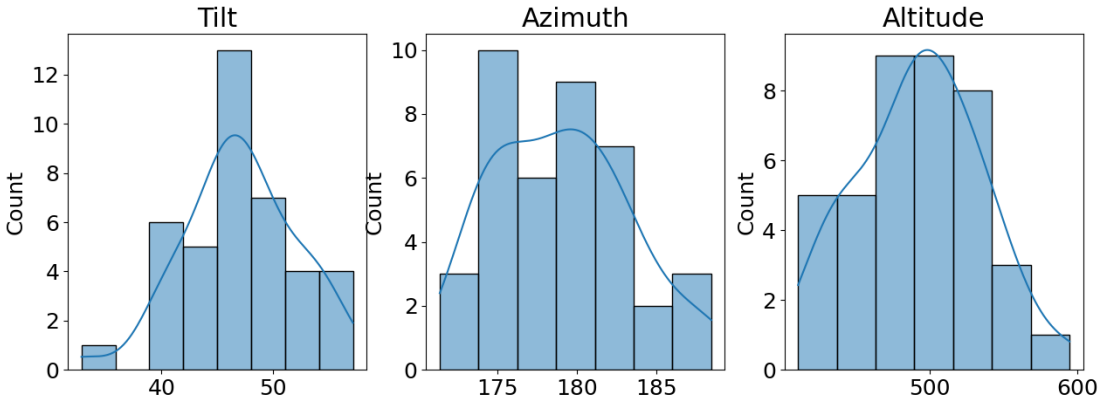

Figure 3: PV panels setup for 40 clients.

Figure 4 shows the power generated for those 40 households in Lausanne for a week of July 2018. Since the data will be normalized before training the models, the normalized power values is also shown. Here, we observe that the power values are relatively different for each household, which means the individuals models will have to learn different patterns. We comment more about this in Section 4.3.

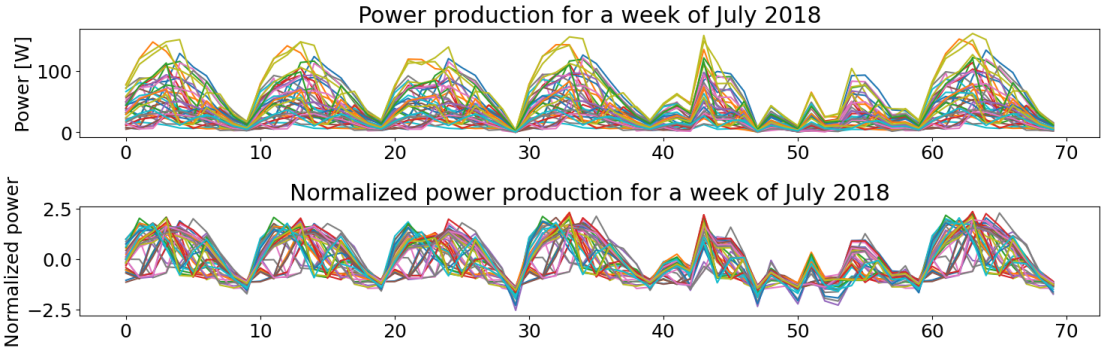

Figure 4: Power generation for 40 households.

As for the weather features, Figure 5 shows the temperature and wind speed for the same week of July 2018 for the same 40 households. We can observe that these features are very different for each household, but still follow an overall similar trend.

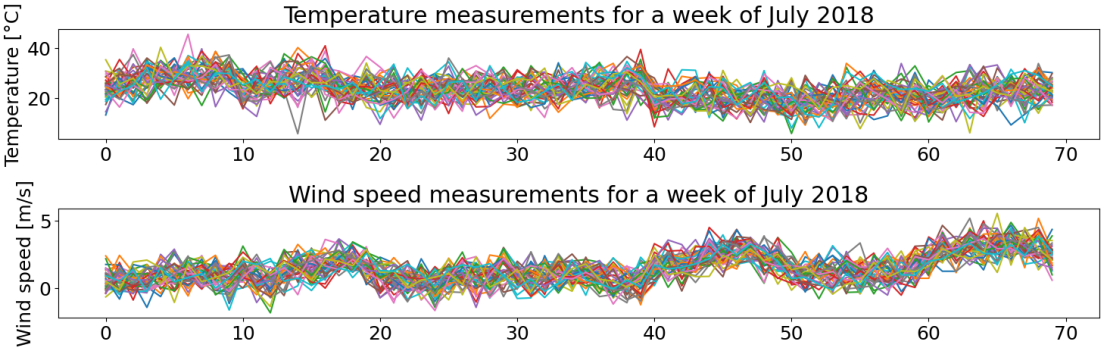

Figure 5: Temperature and wind speed for 40 households.

3.2 Model Architecture

The model used for the online learning scenarios is a simple feedforward neural network, implemented using the Python library *PyTorch*. Feedforward neural networks are a type of artificial neural network where connections between the nodes do not form cycles. They are straightforward and effective for various regression and classification tasks.
In this project, the neural network has two hidden layers, each with 32 neurons. The input layer consists of 10 neurons, which correspond to the 10 features of the dataset. The output layer contains a single neuron, which represents the predicted power generated by the solar panel.
Each of the two hidden layers are followed by a ReLU (Rectified Linear Unit) activation function, which introduces non-linearity to the model, enabling it to learn more complex patterns in the data. The ReLU function is defined as ReLU (x) = max (0, x), which helps in addressing the vanishing gradient problem and accelerates the convergence of the model. The output layer uses a linear activation function, appropriate for regression tasks as it outputs a continuous value representing the predicted power generation.
Initially, the model was designed with only one hidden layer. In the second step, the model complexity was increased by adding an additional hidden layer to improve performance. This adjustment allows the model to capture more intricate relationships within the data.
The loss function used for training the model is the Mean Squared Error (MSE) loss, commonly used for regression tasks, which calculates the average squared difference between the predicted and actual values. It is defined as: MSE = $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$, where y_i is the actual value and $\hat{y}_i$ is the predicted value.
Two optimizers were tested for this project. The first optimizer was Stochastic Gradient Descent (SGD), with a momentum term set to 0.9. SGD is an iterative method for optimizing the loss function, and the momentum term helps accelerate gradients vectors in the right directions, thus leading to faster converging. The momentum term helps in smoothing the optimization path and avoiding local minima. Later on, the optimizer was changed to Adam, an adaptive learning rate optimization algorithm that combines the advantages of two other extensions of SGD: AdaGrad and RMSProp. Adam is known for its efficiency in training deep neural networks and is widely used in various machine learning tasks.
To fine-tune the model's performance, a grid search is employed to determine the optimal hyperparameters. Specifically, the learning rate and the weight decay are tuned. The learning rate controls the step size during the optimization process, and weight decay is a regularization technique that helps prevent overfitting by penalizing large weights. A weight decay of 0 is equivalent to no regularization, potientially leading to overfitting, whereas a weight decay of 1 overregularizes the model, resulting in all weights being close to 0.

3.3 Scenarios

Different scenarios are considered in this project, each employing a distinct training strategy. The primary aim is to compare the performance of online methods, which continually update the model with new data, with offline methods, where the model is trained once and then used for predictions without further updates.
Initially, three preliminary scenarios are devised to gain a comprehensive understanding of the dataset and the model's behavior. In these preliminary scenarios, the model is trained on data from a specific period and then used to predict the same period in the following year. While this approach is less applicable in real-world scenarios where the objective would typically be to predict short-term future power generation, it helps in building foundational

insights about the dataset and the model's capabilities. To simplify the initial comparisons, these scenarios are first tested on data from a single household. Subsequent scenarios then incorporate data from multiple households to evaluate the performance of federated learning and compare it against the single-household results.
The definition of each scenario is as follows:

- **Scenario 0 - Three months of data**
  In this scenario, the model is trained using three consecutive months of data and is then tasked with predicting the power generation for the same three months in the next year. This provides a broad understanding of the model's ability to handle medium-term predictions based on historical data.

- **Scenario 0 - One month of data**
  For this scenario, the model is trained with data from one month and then used to predict the same month in the following year. This approach narrows the focus to a shorter training period, examining how the model adapts with less data.

- **Scenario 0 - One week of data**
  In this scenario, the model is trained on one week of data and predicts the same week of the following year. An additional complexity is introduced by shifting the training window by one day each time, adding one new day of data and removing the oldest day from the training set. This simulates a more realistic situation where data streams are continuously updated, and the model needs to be frequently retrained.

- **Scenario 1**
  In this offline scenario, the model is trained on data from January and then used to predict power generation for the remainder of the year (February to December). This scenario aligns better with real-world applications where a model is trained on recent data and then applied to predict future outcomes without further updates throughout the prediction period.

- **Scenario 2**
  This online scenario involves retraining the model each month with all data available up to that point, assuming data collection starts on the first on January. The model is used to predict power generation for the following month. This scenario demonstrates how the model can improve over time by updating it monthly as more data becomes available, providing insights into the benefits of continuous model updates. However, it also highlights the computational challenges of training the model frequently with a growing dataset.

- **Scenario 3**
  Similar to Scenario 2, this online scenario trains the model each month but only uses data from the previous month for training. This keeps the training dataset size relatively constant and allows for an evaluation of the model's performance with a limited, but recent, dataset.

- **Scenario 4**
  This scenario extends Scenario 3 by using data from the previous two months for training. The goal is to assess whether increasing the training data size, albeit less recent, can improve the model's performance.

- **Scenario 5**
  In this offline scenario, the model is trained on historical data from previous years and used to predict power generation for the entire next year, with separate models for each month. This scenario benefits from a large training dataset and aims to leverage historical patterns to improve prediction accuracy.

- **Scenario 6**
  This final scenario combines elements from previous scenarios by using both historical and fresh data. The model receives 300 samples of historical data, randomly selected from the prediction window of the previous years, and data from the 30 days preceding the prediction window. During training, losses are calculated separately for the historical and fresh data, and a weight is applied to balance the two. The objective is to minimize the weighted sum of these losses, integrating the advantages of both historical trends and fresh observations.

Scenario	Training data	Testing data
1	January	February to December
2	January to M	M+1
3	M	M+1
4	M-1 & M	M+1
5	M - previous years	M - current year
6	M - previous years & M-1	M - current year

Table 2: Training and testing data for each scenario.

Federated Learning

Once all these scenarios have been tested on a single household, federated learning is introduced to improve the results of the models. The training scenarios are the same as in the single household experiments, but data is generated for multiple households. As such, after each training epoch, the models of each client are averaged to obtain a global model, which is then sent back to each client. The final global model is used to make predictions for each client. As presented in [1], only one global model is trained for all clients without any personalization.

4 Results

4.1 Preliminary scenarios - Single household

4.1.1 Scenario 0 - Three months of data

This first scenario was initialy tested to gain an understanding of the dataset. The results are shown for the optimal learning rate and weight decay found through the grid search. Those values are respectively: learning rate=0.01; weight decay=0.005.
As an example, Figure 6 shows the results for the months of July to September, with 2018 as training year and 2019 as test year. Only the first 7 days are shown of the prediction window are shown. The loss value for this prediction window is 0.157.

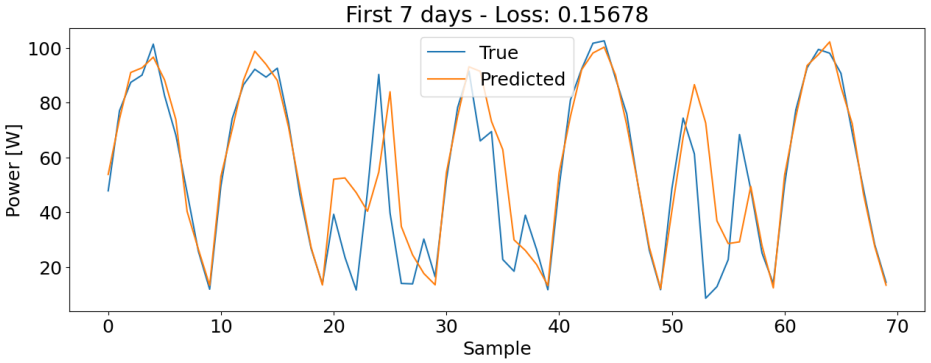

Figure 6: Scenario 0 - First week of July.

This first finding shows that the model is capable of making very good predictions using three months of data. The model makes especially good predictions for the days with almost no variation in the power generation and still follows relatively well the days with more variation.

4.1.2 Scenario 0 - One month of data

This second scenario was tested on each month of the year with 2018 as training year and 2019 as test year. Here, the optimal hyperparameters found through a grid search are: learning rate=0.01; weight decay=0.05. Figure 7 shows the results for the month of June. The loss value for this prediction window is 0.130.

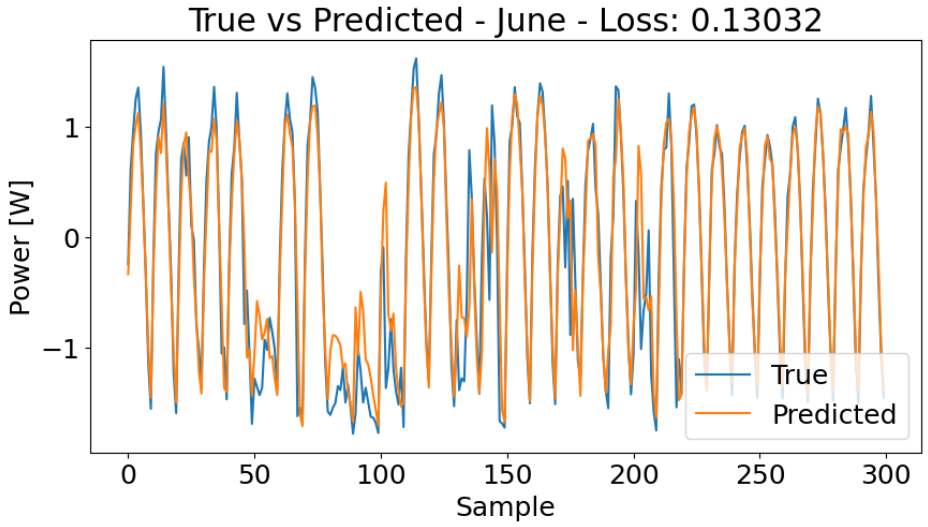

Figure 7: Scenario 0 - June.

In these results, we can see that the model is again able to make good predictions for days with little variation in the power generation, but struggles with days where the power generation is more variable. Still, the loss value is very low, which indicates that the model can really benefit from historical data.

4.1.3 Scenario 0 - One week of data

For this scenario, since the prediction windows are overlapping, the days will be predicted multiple times. An idea to obtain a final prediction for a given day was to average the

predictions of the different models. Figure 8 shows the results for the first week of June. This model resulted in worse results, as only 1 week of data is not sufficient for the model to learn the patterns in the data. The loss value when predicting three months is 0.265.

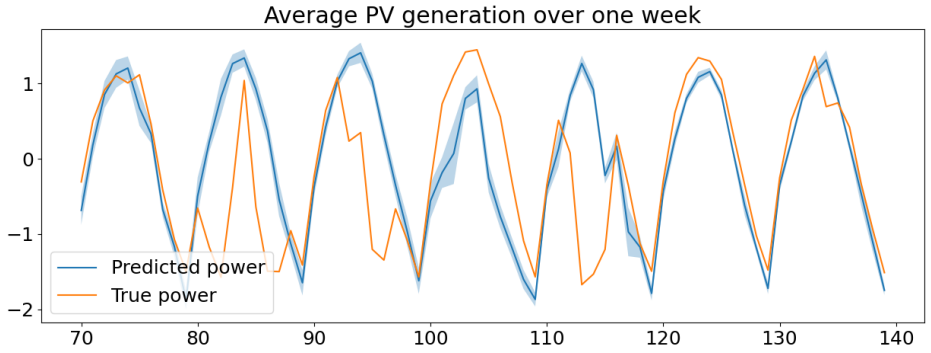

Figure 8: Scenario 0 - First week of June.

4.2 Main scenarios - Single household

For each of the following scenarios, the experiments were conducted for 5 years, from 2014 to 2018. We reset the data collection at the beginning of each year. By doing so, the training dataset size is the same for each year, allowing us to test different scenarios in face of different weather conditions in various years. The results are then averaged over the 5 years to obtain a general idea of the model's performance. Here, 5 individual experiements are done, this means that a new model is defined in January of each year. This also means, depending on the scenario, that the first month(s) of the year are never predicted, since we start the training process in this month. For each year, the model is kept inbetween the months, meaning that the models do not start the training process from scratch each month but simply adapt to the new data.
The data is generated for a single household.

4.2.1 Scenario 1

This scenario uses the data from January to train the model. There are 310 training samples. The model is then used to predict the rest of the year.
Through a grid search, the best results for this scenario were found to be with a learning rate of 0.001 and a weight decay of 0.05. Over the 5 years, the average loss is 1.370 ± 0.457. Figure 9 shows the average loss for each month over the 5 years.

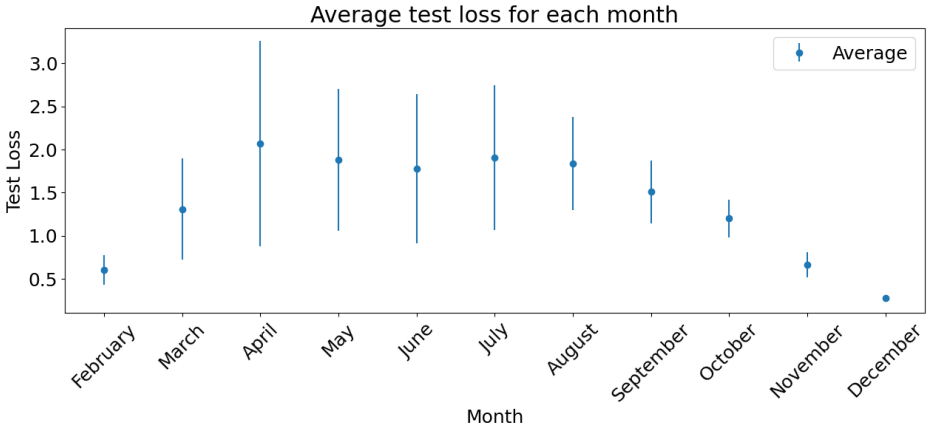

Figure 9: Average test loss for each month for Scenario 1.

The loss value is higher for the spring and summer months, since the data is more variable in spring and has higher values in summer. The average loss for those months also has more variation compared to the winter months, where the loss value is more stable. This shows that this first scenario is not suited for a year-long prediction, and the model should benefit from more frequent updates to adapt to the changing wheather conditions.
Figure 10 shows the predictions and the true values for the months of February and August 2016. Here, we observe that the model struggles to predict the higher values of the power generation in August, but is able to make relatively good predictions for the month of February.

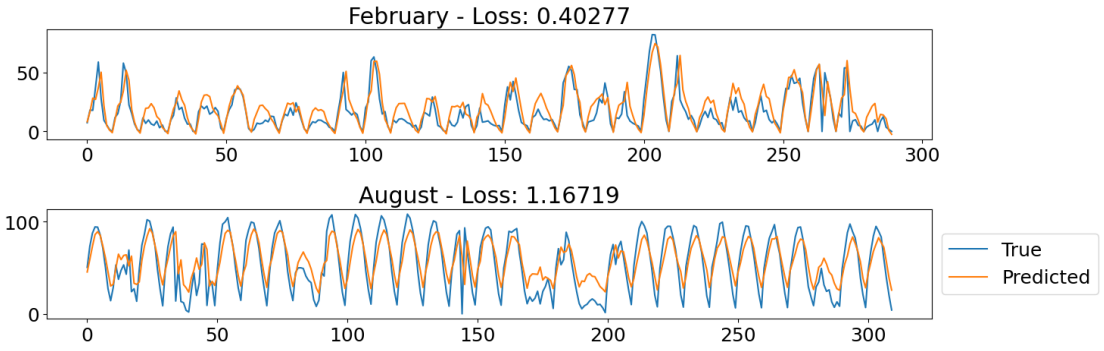

Figure 10: February and August 2016 - Predictions and True values.

4.2.2 Scenario 2

This training scenario is different since the training dataset changes each month. Here, we use all the samples of the year up to the testing month to train the model. The first model used to predict the month of February is composed of 310 samples, and the last model used to predict the month of December is composed of 3340 samples. It is intersting to compare the evolution of the loss value for each month to see if the model is improving as the training dataset grows.
For this scenario, the best parameters found through a grid search are: learning rate=0.01; weight decay=0.05, resulting in a loss of 0.195 ± 0.012 over the 5 years. Figure 11 shows the loss value for each month for each of the 5 testing years.

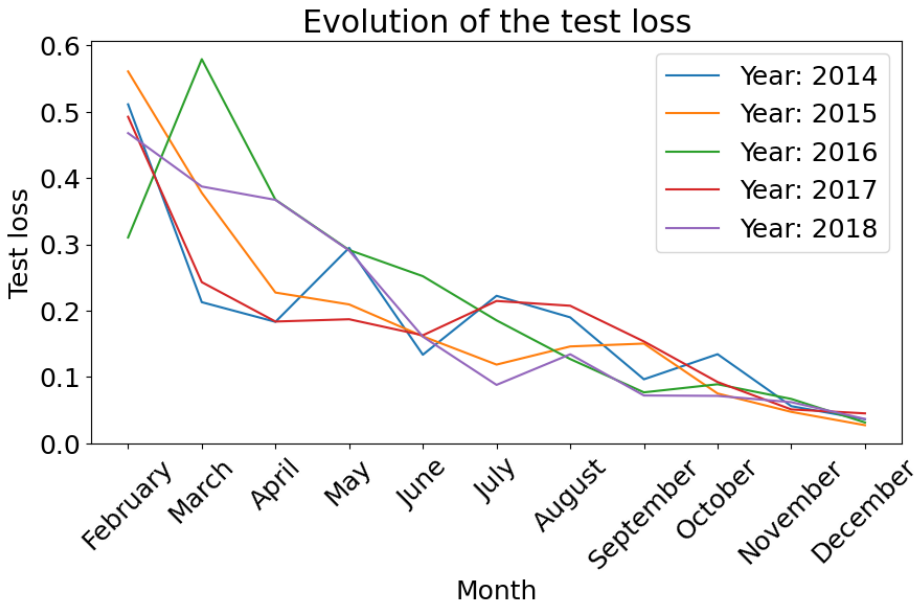

Figure 11: Average test loss for each month for Scenario 2.

As can be expected, the loss value greatly decreases as the training dataset grows. This shows that the model is improving as more data is added. However, the computational cost of training the model with a growing dataset should also be taken into account. Performant models trained with less data are prefered for real-world scenarios.

4.2.3 Scenario 3

Here, the training dataset is composed of all samples from the previous months, so the models are trained on datasets ranging between 280 and 310 samples.
For this scenario, the best hyperparameters are: learning rate=0.001; weight decay=0.05. The loss value for the 5 years is 0.248 ± 0.016.
Since the training dataset is smaller than for Scenario 2, the higher loss value is expected. When computing the average loss for each month, we can see which month are typically easier to predict. Unlike Scenario 1, the summer months are not less predictable, as June and July have a relatively low loss value. This shows that recent data from last month is more informative than historical data for summer months.

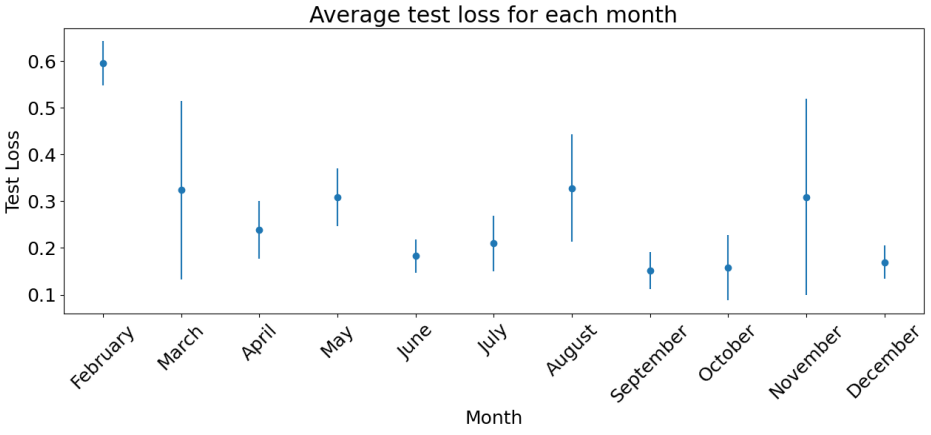

Figure 12: Average test loss for each month for Scenario 3.

From Figure 12, we can interpret that the months of January and February are the least accurately predicted months, whereas the months of September and October are the most accurately predicted ones. This is relatively surprising since the month of October has a lot of variation in the power generation while the winter months of February and March have less variation.

4.2.4 Scenario 4

When increasing the training dataset to the two months before the prediction window, the loss value might decrease since the model has more training samples to learn from. The smallest training dataset is composed of 590 samples and the largest has 620 samples.
The best hyperparameters found through a grid search are: learning rate=0.01; weight decay=0.05. The loss value for the 5 years is 0.187 ± 0.024.

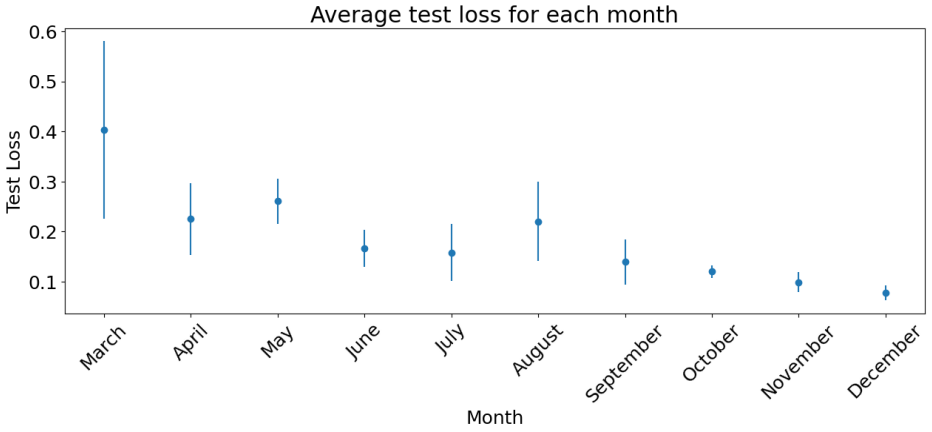

Figure 13: Average test loss for each month for Scenario 4.

Figure 13 shows that, apart for the month of November, there is no notable difference between the loss values for Scenario 3 and Scenario 4, indicating the two months of data do not bring much improvement in this scenario.
Since this model uses two months of training data, February is not predicted in Scenario 4. Because it was the month with the highest loss value in Scenario 3, its absence in Scenario 4 could explain the smallest loss value.

4.2.5 Scenario 5

This scenario has a much bigger training dataset compared with the previous scenarios. Here, we start collecting data in 2005 and keep all samples. For example, when predicting 2014, the model is trained on data from 2005 to 2013, and when predicting 2018, the model is trained on data from 2005 to 2017. As such, the training datasets are ranging between 2540 and 4030 samples.
The best hyperparameters found through a grid search are: learning rate=0.01; weight decay=0.0005. The loss value for the 5 years is 0.144 ± 0.008.

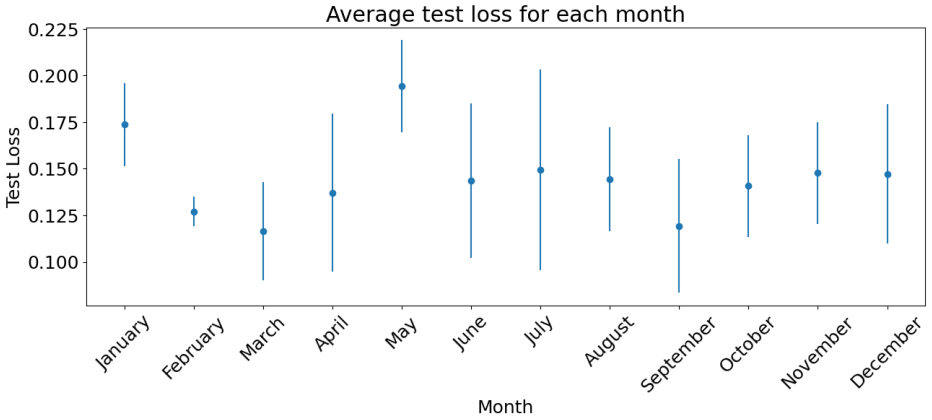

Figure 14: Average test loss for each month for Scenario 5.

Figure 14 shows the average loss for each month over the 5 years. In this scenario, the loss value are typically lower than for the previous scenarios. It is a result of the bigger training dataset and the fact that the model is trained on similar month as the prediciton window, so the samples are more similar. This shows the importance of having historical data to train the model as well as fresh data to adapt to the changing conditions.

4.2.6 Scenario 6

This final scenario combines ideas from the previous scenarios by using historical and fresh data to train the model. An additional hyperparameters is added and is used to balance the two losses. This scenario has a fixed training dataset size, composed of 300 samples of historical data and 300 samples of fresh data.
The best hyperparameters found through a grid search are: learning rate=0.06; weight decay=0.05; memory weight=0.75. The loss value for the 5 years is 0.158 ± 0.011. The results are similar to Scenario 5, even though the model is trained on a much smaller dataset. These results highlight the benefit of training a model on both historical and fresh data. Figure 15 shows the average loss for each month over the 5 years. We can see that only the month of May has a loss higher than 0.2, and that all other months have a loss value lower than 0.2. This model profoundly benefits from the mix of historical and fresh data to make predictions.

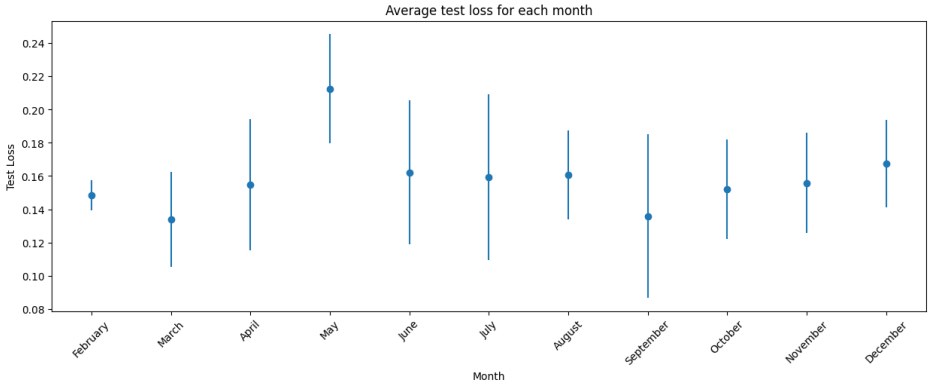

Figure 15: Average test loss for each month for Scenario 6.

Figure 16 shows the predictions for the month of February for the year 2018, as well as the prediction error.

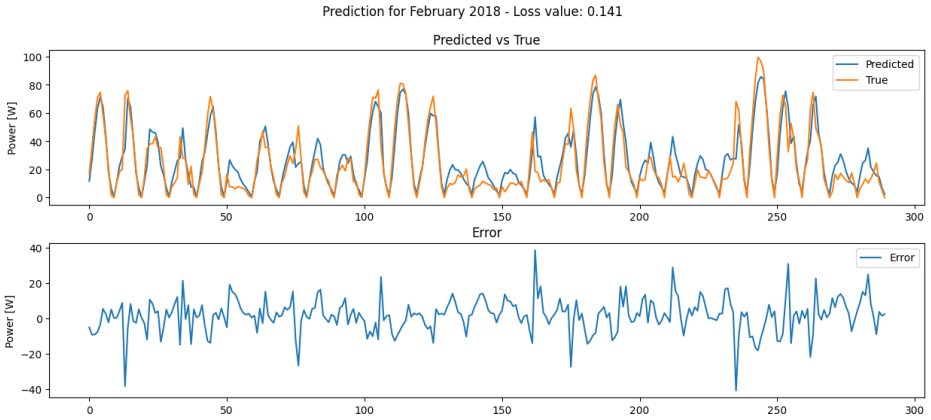

Figure 16: February 2018 predictions and error for Scenario 6.

Here, the predictions are closeley following the true values. This shows that the model is able to make good predictions even though the training dataset is relatively small. Mixing historical and fresh data is a good strategy to improve the model's performance.

4.3 Main scenarios - Multiple households

With a good understanding of online learning capabilities, the next step is to evaluate the performance of federated learning. In particular, Federated Averaging (FedAvg) is employed to aggregate the models of multiple clients and improve the global model.
Based on the results observed in the first part, only scenarios 3 and 6 will be conducted for multiple households. The training scenarios were also slightly changed to make them more realistic, as will be discussed in 4.3.1 and 4.3.2. To understand the impact of FL, those two scenarios are tested for 40 clients with the weight averaging, and for the same 40 clients where each client individually trains its model. This will allow to measure the impact that FL has on the scenarios.
In order to reduce the computational cost, each client will only train its model using a batch of its data. The batch size is set to 10% of the client's data and each client only takes one Gradient Descent step before sending the updates to the server.

4.3.1 Scenario 3

The scenario is slightly adjusted in comparison with the experiment presented in 4.2.3. Instead of running the experiment 5 times for a year, the experiement is performed consecutively for 5 years. This means that a model is trained with the data of December to predict January of the following years.

Individual Training for 40 clients
Each one of the 40 clients has an individual model that they use to predict their own power production.
The results of the grid search give an optimal learning rate of 0.001 and an optimal weight decay of 0.05. Over the 5 years, the average loss is 0.294 ± 0.020. This value is higher than what was obtained before. This can be explained by the higher number of clients, which means that some clients may have a more difficult data and a higher loss value. We are also now predicting January, which might have been particularly hard to predict, hence the higher loss value.

Figure 17 shows the average test loss for all clients for each month, as well as the average latest training loss (last epoch). Since the training loss is much lower than the test loss for almost all months, the models are always overfitting. We can also deduct from the Figure that the winter months are always harder to predict, as the loss value is higher.

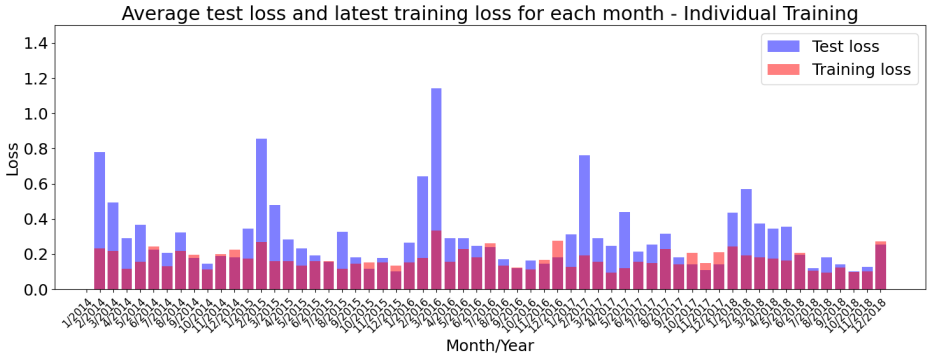

Figure 17: Average testing and training loss for each month for Scenario 3 - Individual Training.

Figure 18 shows the evolution of the training losses for all clients for each month. A linear regression line was also added to each curve to show the evolution of the training loss. For almost each month, the loss values are always decreasing with the epochs, meaning that the models are able to learn from the training data of each client, but that the model are also overfitting since the testing loss is higher.

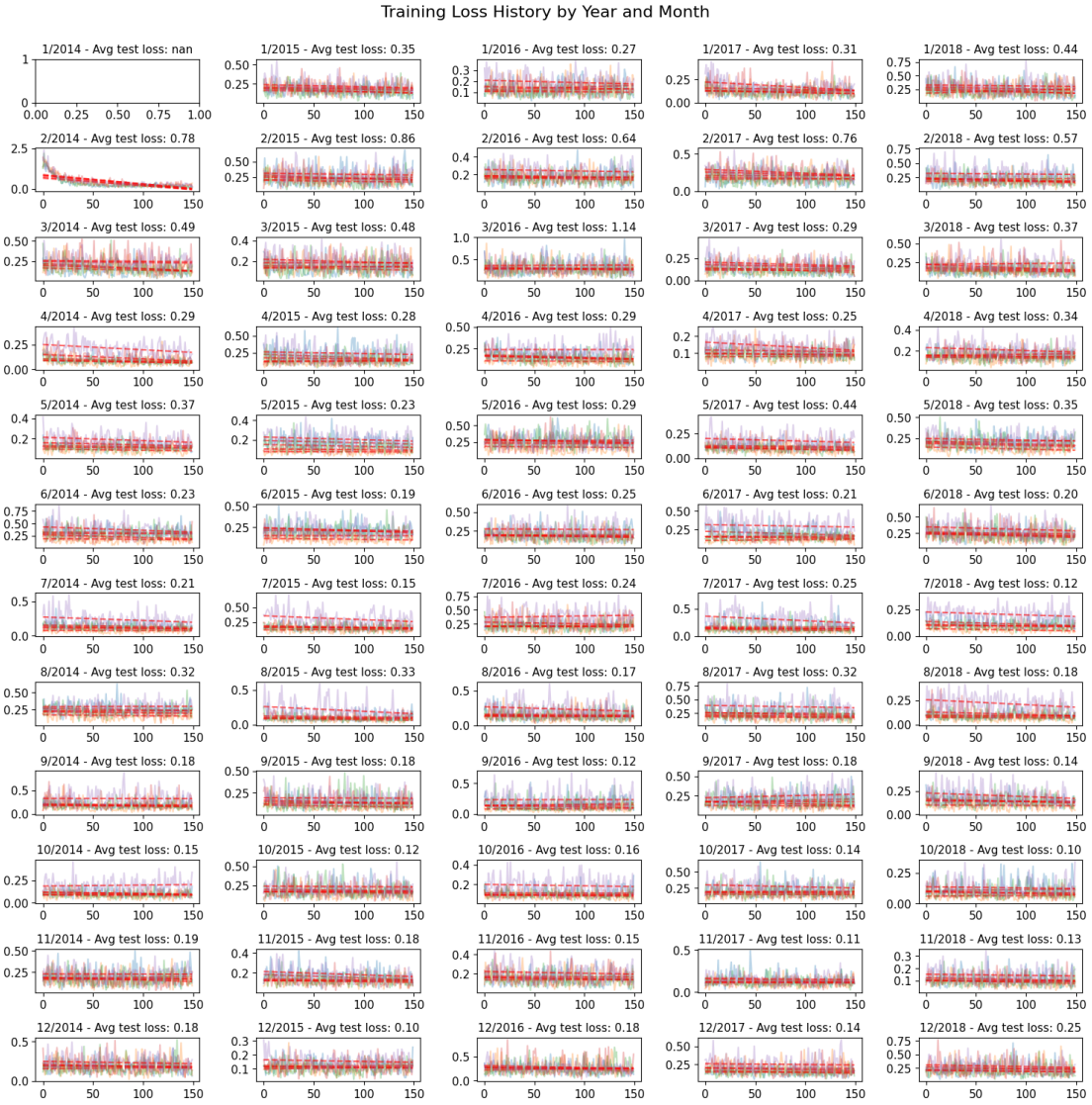

Figure 18: Training loss evolution for each month - Individual Training.

Federated Averaging for 40 clients

The same 40 clients perform the same scenario with the additional step of averaging their weights at each training epoch. This result in only one global model used to make preditions for each household.

The results of the grid search give an optimal learning rate of 0.001 and an optimal weight decay of 0.05. Over the 5 years, the average loss is 0.417 ± 0.018. This value is higher than what was obtained before, meaning that averaging the weights of the models does not improve the results.

However, when comparing the training loss with the testing loss, Figure 19 shows that the models are not overfitting as much as before, as the testing loss is closer to the training loss and often lower. The training loss is also more stable over the months. Similarly to the individual training, the winter months are harder to predict, as the loss values are also higher.

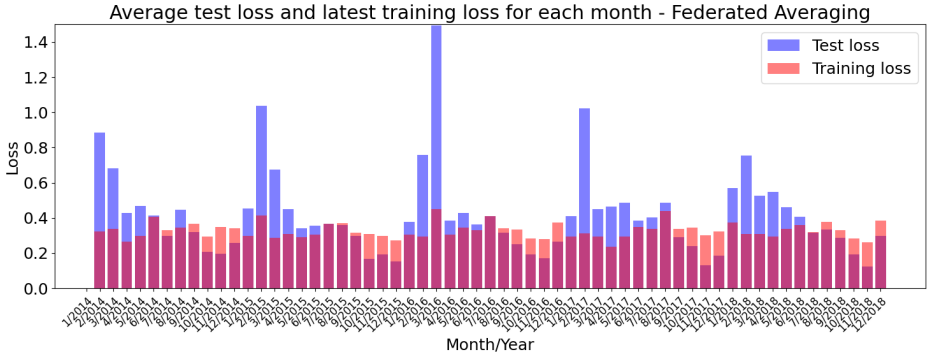

Figure 19: Average testing and training loss for each month for Scenario 3 - Federated Averaging.

When plotting the evolution of the training loss for each month, Figure 20, we can see that the loss values are more stable over the epochs, and typically decrease less than for the individual training, indicating that the models adapt to the new data without overfitting as much. The observed relatively high training loss and its slow decrease might also indicate that a single global model is not capable in capturing different data patterns of clients. We tried to reduce clients heterogeneity by generating data in a scenario with lower variations. Still, FL was not found useful as clients data was extremly similar, hence the data of other clients did not provide further insights. In summary, we osberved that it is difficult to design a scenario where a global federated model outperforms individual models, even though earlier projects have the effectivements of personalization techniques in FL.

Figure 20: Training loss evolution for each month - Federated Averaging.

4.3.2 Scenario 6

This scenario was also slighty adjusted to be tested for 5 years. For computational reasons, the training dataset is composed of 100 samples of historical data and the last 15 days as fresh data. For both the historical and fresh data, the batch size is set to 5% of the data.

Individual Training for 40 clients

The results of the grid search give an optimal learning rate of 0.001, an optimal weight decay of 0.05, and an optimal memory weight of 0.75. Over the 5 years, the average loss is 0.236 ± 0.019, which is also higher than the single household results.

Figure 21 shows the comparison between the average test loss for each month and the average latest training loss for each month. The training loss is lower than the testing loss for almost all months. Scenario 6 with individual training already shows less overffiiting than Scenario 3. The higher loss values for the winter months are also not present in this scenario, reasserting the importance of having fresh and historical data to train the model.

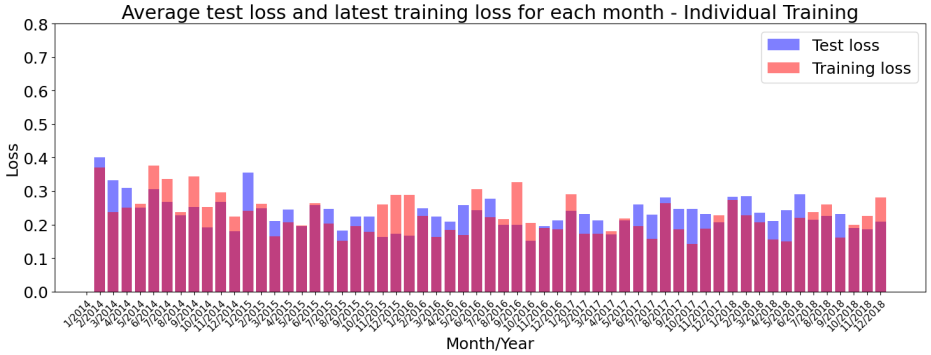

Figure 21: Average testing and training loss for each month for Scenario 6 - Individual Training.

Federated Averaging for 40 clients

When averaging the weights of the models, the results are also not improved. The average loss over the 5 years is 0.344 ± 0.022, with the optimal parameters being: learning rate=0.001; weight decay=0.05; memory weight=0.75.

Figure 22 shows the comparison between the test loss and the train loss. The relationship between training and testing loss is similar to the individual training, but the testing loss is higher. For this particular scenario, federated averaging does not bring much improvement to the results.

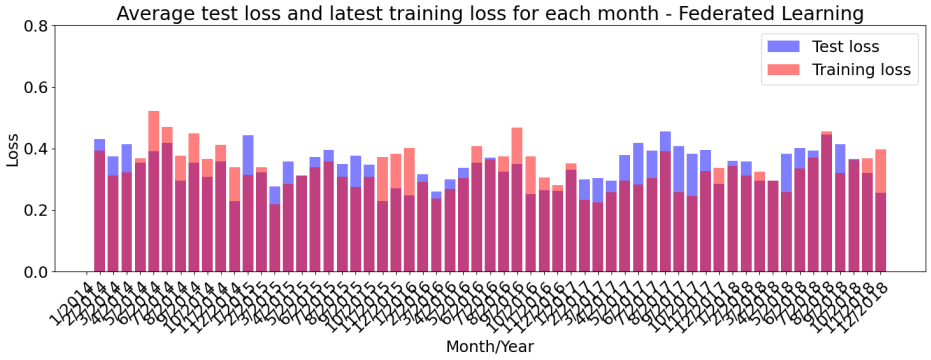

Figure 22: Average testing and training loss for each month for Scenario 6 - Federated Averaging.

5 Discussion

5.1 Single household - Results Comparison

Table 3 shows the average loss value for each scenario when tested on a single household. The results show that online methods are substantially useful for data streams, achieving highly accurate results, comparable to Scenario 5, where the model was trained on a substantially larger dataset. In particular, Scenario 6, which combines historical and fresh data, demonstrates the potential of leveraging both types of data in an online learning fashion to improve model performance, as presented in [1].

Scenario	Training dataset size	Average loss value (5 years)
1	310	1.370 ± 0.457
2	310 - 3340	0.195 ± 0.012
3	280 - 310	0.248 ± 0.016
4	590 - 620	0.187 ± 0.024
5	2540 - 4030	0.144 ± 0.008
6	600	0.158 ± 0.011

Table 3: Average loss value for each scenario with a single household.

5.2 Federated Learning Improvements

The two scenarios implemented with Federated Learning have been compared between single household training and multiple household training. The results show that the loss value is higher when training the models with multiple households, but that the models are generaly less overfitting. This is a good sign that the models are able to generalize better. However, the loss value is higher, which means that the models are not able to learn as well as when trained on a single household.
To improve the loss value in our Federated Averaging pipeline, we should consider implementing personalization techniques. Personalization can help tailor the global model to better fit the unique data characteristics of each individual participant. By doing so, we can address the diverse data distributions among participants and reduce the overall test loss. This approach allows the model to leverage the benefits of collaborative learning while maintaining the flexibility to adapt to specific local patterns. Implementing personalization in the FL pipeline can enhance model performance and ensure that the improvements are consistent across different participants, ultimately leading to better generalization and lower overfitting.
Lastly, the setup could be made more realistic by adding different client importance based on the number of samples they have. We could imagine a real-world scenario where some clients have more data than others or more memory capabilities which would alter their training process.

6 Conclusion

In conclusion, this study shows that Federated Learning for data streams can be very useful for training models with data from different households. Our results show that online learning reduces the amount of data needed to achieve good results and that FL can help reduce overfitting, which means the model can work better on new data. However, we also see that the predictions are less accurate when using data from multiple households. This suggests that the models need more improvement to learn as well as models trained on data from a single household.
This work also suggested possible improvements to the FL pipeline, such as implementing personalization techniques to tailor the global model to individual participants and considering different client importance based on the number of samples they have. These improvements could help enhance model performance and ensure better generalization across different participants.

References

[1] Othmane Marfoq, Giovanni Neglia, Laetitia Kameni, and Richard Vidal. Federated learning for data streams, 2023.

[2] William F. Holmgren, Clifford W. Hansen, and Mark A. Mikofski. pvlib python: a python package for modeling solar energy systems. *Journal of Open Source Software*, 3(29):884, 2018.

[3] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning, 2021.