# Data Disparity and Temporal Unavailability Aware Asynchronous Federated Learning for Predictive Maintenance on Transportation Fleets

**Leonie von Wahl[1], Niklas Heidenreich[1], Prasenjit Mitra[2], Michael Nolting[1], Nicolas Tempelmeier[1]**

[1]Volkswagen Group, Hannover, Germany
[2]Penn State University, State College, USA
leonie.von.wahl@volkswagen.de

## Abstract

*Predictive maintenance* has emerged as a critical application in modern transportation, leveraging sensor data to forecast potential damages proactively using machine learning. However, privacy concerns limit data sharing, making *Federated learning* an appealing approach to preserve data privacy. Nevertheless, challenges arise due to disparities in data distribution and temporal unavailability caused by individual usage patterns in transportation. In this paper, we present a novel *asynchronous federated learning* approach to address system heterogeneity and facilitate machine learning for predictive maintenance on transportation fleets. The approach introduces a novel data disparity aware aggregation scheme and a federated early stopping method for training. To validate the effectiveness of our approach, we evaluate it on two independent real-world datasets from the transportation domain: 1) oil dilution prediction of car combustion engines and 2) remaining lifetime prediction of plane turbofan engines. Our experiments show that we reliably outperform five state-of-the-art baselines, including federated and classical machine learning models. Moreover, we show that our approach generalises to various prediction model architectures.

## Introduction

Federated Learning (FL) has emerged as a collaborative learning paradigm that prevents privacy issues by not sharing training data but only model parameters, e.g., the learned neural network weights. A global server coordinates the training of multiple clients, which only have access to their local data (McMahan et al. 2017). In canonical federated learning, client models are collected synchronously to update the global model.

However, this approach does not apply to many heterogeneous real-world problems, in which clients may send updates at irregular intervals or drop out during training (Li et al. 2020). Asynchronous federated learning (AFL) schemes that address the heterogeneity of the systems have emerged as an alternative to classic FL (Chen, Mao, and Ma 2019). In AFL, the global server aggregates parameter updates of a single client as soon as that device has completed its local training round. However, existing AFL approaches suffer from the following problems: (i) While existing systems aim to select the most promising nodes for training (Chen et al. 2021; Wu et al. 2020), they neglect the data disparity, i.e., differences in the amount of available training data among the clients. The data disparity can introduce a bias towards single contributors to the global model. The few existing approaches addressing data disparity lack flexibility and favour contributors with large amounts of data (Hao, Zhao, and Zhang 2020). (ii) Existing approaches can not handle *dropouts*, i.e., clients that lose the connection to the server during training, and typically exclude these clients from being aggregated into the global model (Zhang, Bosch, and Olsson 2021). (iii) Due to the different data volumes and convergence speeds, it is unclear when to stop the federated training since existing schemes, e.g., early stopping, can not be directly applied (Li et al. 2021).

These challenges are particularly relevant for *predictive maintenance* in transportation fleets, where the clients are individual vehicles. Due to individual travel patterns, the vehicles produce data at different sizes and may drop out due to connectivity problems (Yang et al. 2022). In this paper, we propose a novel data disparity aware asynchronous federated model which enables predictive maintenance for engines in transportation fleets. We address the *data disparity* and *dropouts* by proposing a Disparity Aware AFL (DAAFL) aggregation scheme which considers the training data sizes and update frequencies of the individual clients throughout the aggregation weights. Moreover, we address the *training length determination* by introducing a novel AFL *early stopping* method, which computes a federated validation loss. To the best of our knowledge, this is the first study exploring an asynchronous federated stopping criterion. We perform an extensive evaluation on two real-world datasets of transportation fleets, i.e. the prediction of oil dilution of car engines and the prediction of the remaining lifetime of aeroplane turbofan engines. We simulate individual dropout patterns for each client to show the performance under temporal unavailability. Our experiments show that our proposed DAAFL approach can reliably improve over existing synchronous, semi-asynchronous and asynchronous federated learning schemes in both tasks. Furthermore, we demonstrate that our DAAFL generalises to the established neural model architectures for time series prediction.

Our contributions can be summarised as follows:

- We formulate a dynamic parameter aggregation rule and a federated early stopping scheme which considers data

disparity and temporal availability of clients and identifies an appropriate number of training iterations in a highly heterogeneous federation.

- We implement a data disparity and temporal unavailability aware asynchronous federated learning approach for time series data for predictive maintenance on transportation fleets.
- We evaluate our DAAFL approach on two real-world datasets of oil dilution and the remaining lifetime of aeroplane engines.

Our approach is extendable to other federated learning problems based on time series.

## Problem Statement

FL aims to learn a global model by training on distributed nodes without sharing the training data (McMahan et al. 2017). We want to minimise the global objective function $f$ defined by

$$f(\theta_G) \coloneqq \frac{1}{n}\sum_{i=1}^{n} f_i(\theta_G)$$

where $f_i$ are local objective functions of the individual nodes, $\theta_G \in \mathbb{R}^d$ are the global model parameters, and $n$ denotes the number of participating models. In the machine learning context, the individual training losses typically serve as local objective functions $f_i$. In synchronous FL, the updated local model parameters yield the new global model parameters by the aggregation scheme

$$\theta_G^{v+1} = \frac{1}{n}\sum_{i=1}^{n} \theta_i^v. \tag{1}$$

Here, $v$ is the current aggregation round, and $\theta_i$ denotes the local model parameter. This scenario is synchronous, i.e., the aggregation server has to wait for all participating devices.

In asynchronous FL, the global server aggregates local model updates whenever a local model has completed its training rounds. The asynchronous aggregation scheme is

$$\theta_G^{v+1} = (1-\alpha)\theta_G^v + \alpha\theta_i^{v+1} \tag{2}$$

where we exclusively aggregate the model parameters of the latest trained local model (Xu et al. 2023). The choice of the aggregation weight $\alpha$ depends on several influencing factors (Hao, Zhao, and Zhang 2020). First, the global data volume is not equally distributed among the clients. Devices with small data volumes train much faster than those with large ones. Therefore, the large data volume devices will be stragglers. In a synchronous FL scheme, stragglers will scarcely influence the global model, although containing the majority of the training material. Second, classic synchronous learning schemes are not designed to handle sudden drop-outs of clients. The aggregation parameter choice can compensate for the heterogeneous availability of clients.

Another issue of FL is the choice of the total number of local epochs. Classical early stopping methods measuring the validation error are no option when no global dataset is available. A trivial averaging of the local model's validation error neglects the different convergence speeds of the models due to temporal unavailability or data disparity.

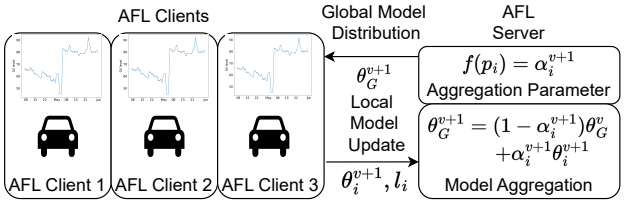



Figure 1: Schematic overview of the DAAFL method for oil dilution prediction.

In this paper, we aim to enable the training of an effective asynchronous federated model which takes different data volumes and drop-outs into account. In particular, we aim to find the optimal aggregation parameter rule $\alpha_i^{v+1}$ for device $i$ such that the global model's loss is minimised:

$$\min_{\theta_G} f(\theta_G) \quad \text{with } \theta_G^{v+1} = (1-\alpha_i^{v+1})\theta_G^v + \alpha_i^{v+1}\theta_i^{v+1}. \tag{3}$$

Moreover, we aim to develop an early stopping policy adapted to asynchronous FL.

## Data Disparity Aware AFL

This section presents our DAAFL approach to data disparity aware asynchronous federated learning. The aim is to mitigate single clients providing frequent updates to the model and dominating the shared federated model while boosting slow models with frequent dropouts. The core concept behind DAAFL involves assigning weights to client updates based on their data volumes and past involvement in the training process. An overview of the DAAFL methodology is depicted in Figure 1: The clients submit their local models $\theta_i^{v+1}$, dataset size and validation loss to the global server. The server uses the DAAFL aggregation scheme to compute the ensuing global model $\theta_G^{v+1}$.

We introduce the model aggregation function and the early stopping procedure in Section *DAAFL Model Aggregation and Early Stopping*. Comprehensive algorithms detailing the DAAFL approach are provided in Section *DAAFL Algorithm*.

### DAAFL Model Aggregation and Early Stopping

We aim to develop an aggregation scheme and an early stopping mechanism that prevents frequent updates from clients with smaller data volumes from biasing the model.

**Aggregation Parameter Rule** We seek to establish an aggregation parameter rule for Eq. 3, customised for each client, which considers data disparity and prior aggregation frequency. In FedAvg, the original FL scheme (McMahan et al. 2017), the relative data volume is quantified as $d_i = p_i/p_G$ in the aggregation in Eq. 1 where denotes the data points on client $i$ and $p_G$ represents the total data points. However, relying solely on data proportion as an aggregation weight proves inadequate for asynchronous FL. The frequent updates resulting from smaller data volumes and

the temporal availability will bias the model towards consistently updating clients. Hence, incorporating update frequency into the aggregation function is pivotal for effective asynchronous training.

Our approach works as follows: First, we assess the cumulative contribution of each client to the global model until the present point. Then, we compute the equitable share of contribution that each client ideally should possess. Next, we determine how far the actual contribution is from the ideal one. Lastly, we derive an aggregation parameter that rectifies this disparity by under- or over-weighting clients during the current aggregation cycle. Formalising the approach: In asynchronous FL, one client updates per aggregation round. In aggregation round $v+1$, let a client $i$ send its parameters to the server for computing the global model's $(v+1)$-th version. The actual contribution of the client, including the current aggregation round, can be expressed as the sum of its aggregation parameters from round 1 to $v+1$:

$$\alpha_{sum_i}^{v+1} \coloneqq \sum_{k=1}^{v+1} \alpha_i^k b_i^k. \tag{4}$$

Here, $b_i^k \in \{0,1\}$ indicates for aggregation round $k$ whether client $i$ sent its parameters to the server, i.e., $b_i^k = 1$, or whether some other client updated, i.e., $b_i^k = 0$. In an equitable scenario, all clients update uniformly, approximately every $n$ round. After $v+1$ aggregation rounds, each client should have contributed $(v+1)/n$ times. Furthermore, the server would aggregate the clients according to their relative data volume $d_i$. Hence, the equally balanced aggregation parameter sum $\hat{\alpha}_{sum_i}^{v+1}$ would be

$$\hat{\alpha}_{sum_i}^{v+1} \coloneqq \frac{d_i}{n}(v+1). \tag{5}$$

We want to determine how to choose the aggregation parameter $\alpha_i^{v+1}$ to close the gap between actual and ideal contribution. Therefore, we compare the real aggregation parameter sum to the ideal one by equating Equations 4 and 5:

$$\hat{\alpha}_{sum_i}^{v+1} = \alpha_{sum_i}^{v+1}.$$

Since client $i$ updates in aggregation round $v+1$, we have $b_i^{v+1} = 1$. Therefore, by splitting the sum in Eq. 4 we get to

$$\alpha_i^{v+1} = \frac{d_i}{n}(v+1) - \alpha_{sum_i}^{v}. \tag{6}$$

The resulting aggregation parameter $\alpha_i^{v+1}$ can not become negative, which can be verified by an induction proof starting from $\alpha_i^1 \in \{0, \frac{d_i}{n}\}$. However, we may receive $\alpha_i^{v+1} > 1$ if the client has a very low update frequency such that $v$ is high whereas $\alpha_{sum_i}^{v}$ is very low. To receive no negative coefficients in the aggregation formula Eq. 3, we set

$$\alpha_i^{v+1} = \min(1, \frac{d_i}{n}(v+1) - \alpha_{sum_i}^{v}).$$

Our new aggregation formula ensures that clients who have not been sufficiently included in the past have a more significant impact in the present.

Algorithm 1: Client Update

```
function CLIENT_UPDATE(θ_G)
    θ_i ← θ_G
    while True do
        for e ← 1, E do
            G_t ← stochastic gradient for f_i at θ_i
            l_i ← validation loss of θ_i
            θ_i ← θ_i − ηG_e
        end for
        return θ_i, p_i, l_i
    end while
end function
```

**Early Stopping Mechanism** We want to develop an asynchronous federated early stopping method. Regular early stopping methods for evaluating the model on a global validation set do not work if all data are on the clients. The apparent idea is to average the local validation losses. However, we have to deal with the problem of different convergence speeds: Models based on small data volumes and less temporal dropouts will sooner overfit than slow training clients. Hence, we determine the local validation loss by each client's contribution so far. First, we initialise $l_G^0$ by some value. Let $l_G^v$ be the current federated validation loss in aggregation round $v$. When a client has finished its local training round, it computes its local validation loss $l_i^{v+1}$. Then, we compute the new federated validation loss by using the aggregation parameter established in Eq. 6:

$$l_G^{v+1} = (1 - \alpha_i^{v+1}) l_G^v + \alpha_i^{v+1} l_i^{v+1}.$$

Thus, we have a composition of the validation loss representing the composition of the federated model.

## DAAFL Algorithm

This section describes the algorithms to run the DAAFL approach. Algorithm 2 describes the server operations while Algorithm 1 presents the client-side update.

In the beginning, we initialise the global and local models' parameters with the same weights. We set the global model's version number to $v = 0$ and the aggregation parameter sum for each client to $\alpha_{sum_i} = 0$. Then, we initialise the federated validation loss $l_G$, the former validation loss $l_{G,old}$ and the non-improvement counter $l_{count}$. After that, the clients start parallel training on their respective data set in Algorithm 1. We limit the number of local epochs per round by $E$. The section *Machine Learning Model Implementation* describes a single node's underlying machine learning architecture.

We then send the local weights and the local validation loss to the server, see Algorithm 2, and compute the local aggregation parameter $\alpha_i$ as explained in Section *DAAFL Model Aggregation and Early Stopping*. The server aggregates the new weights and sends them back to the client from which it received the update. Next, we increase the global version number by one and update $\alpha_{sum_i}$. Finally, we calculate and compare the new federated validation loss to the previous loss. If the loss has not improved by at least $\epsilon$ for

Algorithm 2: Server Function

**function** SERVER_FUNCTION()
  Initialise $\theta_G, p_G, l_G, l_{G,old}, l_{count}, M, \epsilon$
  $v \leftarrow 0$
  $\alpha_{sum_i} \leftarrow 0$ for $i \in \{1, ..., n\}$
  Initialise all clients
  **while** $l_{count} < M$ **do**
    **if** any client $i$ sends its update to the server **then**
      $\theta_i, p_i, l_i \leftarrow Client\_Update(\theta_G)$
      $d_i = \frac{p_i}{p_G}$
      $\alpha_i \leftarrow \min(1, d_i/n \cdot (v+1) - \alpha_{sum_i})$
      $\theta_G \leftarrow (1-\alpha_i)\theta_G + \alpha_i\theta_i$
      $v \leftarrow v + 1$
      $\alpha_{sum_i} \leftarrow \alpha_{sum_i} + \alpha_i$
      $l_G \leftarrow (1-\alpha_i)l_G + \alpha_i l_i$
      **if** $l_{G,old} - l_G < \epsilon$ **then**
        $l_{count} \leftarrow l_{count} + 1$
      **else**
        $l_{count} \leftarrow 0$
        $l_{G,old} \leftarrow l_G$
      **end if**
    **end if**
  **end while**
**end function**

some $\epsilon > 0$, we increase $l_{count}$ by 1. Then, if $l_{count}$ does not surpass some $M > 0$, we continue the training.

## Evaluation Setup

This section describes the baseline approaches, the dataset preparation, the machine learning model used by the DAAFL, and the considered evaluation metrics. We design the experimental setup to be as realistic as possible.

### Baselines

To evaluate our solution to the problem stated in Section *Problem Statement*, we compare it to several state-of-the-art baselines. The architecture of the regression model itself is identical in all approaches.

**Classic ML**. The collective model is the classic machine learning approach, which does not preserve privacy. We train one global model with all devices' data at its disposal.

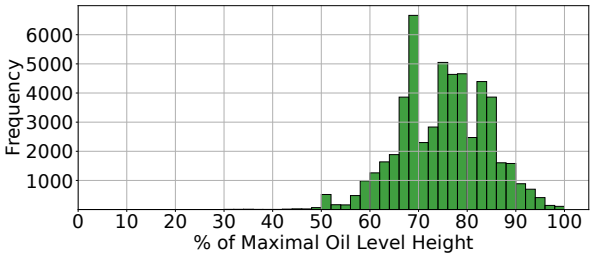
**FedAvg**. The first FL scheme is proposed in (McMahan et al. 2017). This approach is synchronous, i.e., we select some nodes in each round to participate in the training and wait until all of them have completed the training process.

**St-AFL**. In this approach in (Zhang, Bosch, and Olsson 2021), the authors develop an asynchronous FL scheme for steering angle prediction. They select the client by comparing the local model's version numbers to the global version number. They only admit these clients whose updates are neither too recent nor too stale for aggregation.

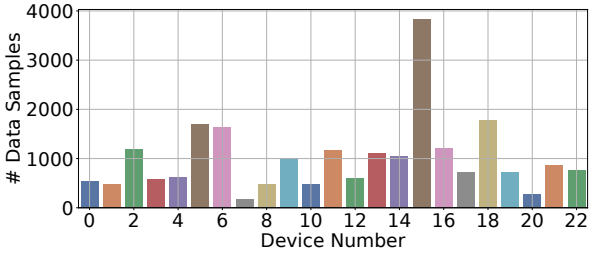
**Pr-SAFL**. This approach proposed by (Hao, Zhao, and Zhang 2020) is semi-asynchronous. The authors use a priority function for the node selection, considering computing power, delay and data volumes. The selected nodes then train synchronously while the others train asynchronously.

| Dataset | #Clients | #Hours / #Cycles | #Upper / #Lower Cases |
|---|---|---|---|
| Oil Dilution | 23 | 53.647 | 1401 / 887 |
| Jet Engine | 10 | 33.727 | - |

Table 1: Statistics of the datasets.




(a) The distribution of oil levels in the oil dilution dataset.




(b) The different training data volumes of the 23 vehicles.

Figure 2: Data distribution on the vehicles' datasets.

**FedSA**. This semi-asynchronous approach (Chen, Mao, and Ma 2021) divides the training into a synchronous initial stage and an asynchronous convergence stage. The client's staleness steers the number of local epochs.

**LR-AFL**. This asynchronous scheme described in (Chen et al. 2020) uses a surrogate objective function to ensure the local models do not diverge too much. Moreover, the authors adopt an adaptive learning rate which privileges devices with large data volumes.

### Dataset Preparation

**Oil Dilution Dataset** *Oil dilution* is a potential engine damage that analysis of vehicle sensor data can avoid. Oil dilution denotes the unwanted entrance of fuel into the engine oil. In such cases, the oil level sensor readings increase with fluid levels. The dilution can lead to the tearing of the oil film such that the oil can not lubricate the engine appropriately. If discovered in time, an oil exchange can avoid engine damage. Thus, our first application is to predict the oil level of the next 24 hours, knowing the last 48 hours. Then, we can forecast critically high or low oil levels.

Our data set for oil dilution detection was collected by logistics vehicles of the Volkswagen Group. All 23 cars have combustion engines, and a CAN logger records the oil level height with a sampling rate of 0.1 seconds. For the data preprocessing, we identify the oil refills and split the oil level

| Data Set | Input [h]/[cycles] | Output [h]/- | Optimiser | Batch Size | Learning Rate |
|---|---|---|---|---|---|
| OD | 48 | 24 | Adam | 32 | $10^{-3} \cdot 0.9^E$ |
| JE | 80 | RUL | Adam | 32 | $10^{-3} \cdot 0.9^E$ |

Table 2: Model parameters.

time series accordingly since the oil refills are unpredictable for the model. In the resulting time series, we have 53.467 hours of driving; see Table 1. The histogram in figure 2a displays the distribution of oil levels. We spot the most frequent oil levels from 60 to 80 mm. We set an upper and a lower threshold for the oil level to detect abnormal oil behaviour. The upper threshold is crossed 1401 times in the data, while the oil level only falls 887 times below the lower threshold. As we see Figure 2b, we have a heavy data disparity among the 23 vehicles: While vehicles 3, 10 and 15 have more than 3000 training samples, the majority of the samples have 500 to 1500 training samples. This high disparity can occur if the vehicles' recording or usage patterns differ. We use part of the data as a test set such that it contains threshold cases. The remaining data are randomly split into a validation set (20%) and a training set (80%) for each client.

**Jet Engine Dataset** Our second application is to predict aircraft engines' remaining useful life (RUL). We use a run-to-failure data set simulated and presented by NASA in (Saxena et al. 2008). A thermodynamical simulation model creates the simulation and yields four datasets which differ in combinations of operational settings and fault modes. All in all, the dataset covers 33.727 life cycles. We have 26 sensor measurements for each life cycle, three operational settings and the engine number. In the first step, we calculate the RUL for each time cycle. Then, we split the dataset into ten unbalanced clients. Thus, we create a federated dataset with heavy data disparity. The task is to predict the RUL from a data sample with a window size of 80 life cycles containing the corresponding sensor measurements and operational settings. The client-wise split into test, validation and training sets takes place randomly.

## Machine Learning Model Implementation

**Oil Dilution (OD)** In the first application, we use a stack of two encoder-decoder GRU networks to perform sequence-to-sequence learning, where each GRU network has 25 units. As a training loss metric, we choose the mean squared error to train the model for detecting outliers, i.e. the crossing of critical threshold values. We schedule the learning rate $\eta$ as $10^{-3} \cdot 0.9^E$ where $E$ is the current epoch's number. Moreover, we choose a batch size of 32 and Adam as the optimiser for the regression model.

**Jet Engine (JE)** For the regression problem of predicting the remaining useful life of aeroplane turbofan engines, we use a stacked network of three GRU with 200, 100 and 50 units. Loss, batch size, learning rate and optimiser are analogous to the oil dilution case; see Table 2.

## Dropout Simulation

Temporal unavailability is typical in automotive use cases due to connectivity issues and different vehicle usage patterns. Due to the lack of data sets containing temporal unavailability, we simulate it here by assigning an individual dropout behaviour to each client. The clients are unavailable for 15 to 75 seconds, while the dropouts can occur every few minutes. In particular, the length of dropout is constant per client but differs among the clients. Furthermore, each client's time pattern when the dropout happens is individual. That reflects that unavailability is not random but follows specific patterns.

## Evaluation Metrics

To understand the goodness of the oil level or RUL prediction, we propose the following quality indicators:
**Mean Absolute Percentage Error** (*MAPE*). The mean absolute percentage error of the regression prediction.

**Root Mean Square Error** (*RSME*). The square root of the mean squared error of the regression prediction.

For the oil dilution case, we need specific evaluation metrics. To classify whether there is a lower or upper threshold case in the predicted oil level time series, we compare the oil levels to the upper and lower threshold value, respectively.
**Lower Precision** (*L-P*). The lower precision is the fraction of true positive lower threshold cases among the samples classified as positive.
**Lower Recall** (*L-R*). The lower recall gives the fraction of correctly classified lower threshold cases among all lower threshold cases.
**Lower F1 Score** (*L-F1*). The lower F1 score combines the lower precision and lower recall using their harmonic mean. Hence, the F1 score is the most critical metric for recognising threshold cases.
**Upper F1 Score** (*U-F1*), **Upper Recall** (*U-R*), **Upper Precision** (*U-P*): We define the metrics analogously to L-P, L-R, and L-F1 for the upper threshold.

# Evaluation

The evaluation aims to assess the ability of the proposed DAAFL approach in asynchronous federated learning settings with heavy data disparity and frequent dropouts. To this end, we first compare the prediction performance with state-of-the-art baselines on two distinct data sets. Following this, we explore the versatility of DAAFL's model architecture by integrating it with established model architectures.

## Performance with Data Disparity and Dropouts

Our evaluation centres on two applications: Firstly, we predict the forthcoming 24-hour oil level using the preceding 48 hours, aiming to detect oil dilution. We classify critical oil dilution cases by predicting whether the oil level surpasses an upper threshold or descends below a lower one. The second application involves forecasting jet engines' remaining useful life (RUL), leveraging information from the last 80 life cycles. We juxtapose DAAFL against the baselines detailed in Section *Baselines* according to the metrics explained in Section *Evaluation Metrics*. In addition,

| Algorithm | FL Type | MAPE [%] | L-F1 [%] / L-P [%] / L-R [%] | U-F1 [%] / U-P [%] / U-R [%] |
|---|---|---|---|---|
| Classic ML | NF | 1.41±0.16 | 91.51±00.26 / 097.63±00.80 / 86.13±00.80 | 77.23±4.46 / 99.51±0.66 / 63.34±05.79 |
| FedAvg | S | 1.62±0.04 | 81.07±02.53 / **100.00**±00.00 / 68.24±03.72 | 75.28±1.66 / 98.61±1.12 / 60.95±02.58 |
| FedSA | SA | 1.79±0.09 | 89.03±00.69 / 099.83±00.24 / 80.36±01.25 | 77.11±2.88 / **99.74**±0.57 / 62.96±03.96 |
| Pr-SAFL | SA | 2.65±0.07 | 16.05±00.08 / **100.00**±00.00 / 08.72±00.05 | 74.02±0.80 / 98.76±0.48 / 59.20±01.21 |
| St-AFL | A | 3.34±1.75 | 74.68±31.22 / 091.99±25.07 / 67.60±29.71 | 75.83±9.33 / 88.92±9.19 / 69.12±15.22 |
| LR-AFL | A | 1.59±0.04 | 89.01±00.40 / 099.77±00.14 / 80.36±00.72 | 77.46±1.16 / 98.60±1.16 / 63.83±02.04 |
| DAAFL | A | **1.51**±0.07 | **89.18**±00.18 / 098.35±00.89 / **81.59**±00.61 | **80.87**±0.59 / 85.23±3.28 / **77.06**±01.77 |

Table 3: Prediction results on the oil dilution test set. Each experiment was run 30 times. The F1 score, precision and recall are grouped to recognise lower and upper threshold cases. The best federate results are marked in bold.

we provide the prediction performance of the *Classic ML* model, i.e., the same GRU-based model in a non-federated version. We run each experiment 30 times and provide the metric's mean and standard deviation for each metric. We stop the baseline runs after 1000 local training rounds, while the DAAFL is stopped by its early stopping mechanism.

**Oil Dilution Prediction** Table 3 presents the results for the oil dilution prediction performance of the DAAFL, the synchronous (S), semi-asynchronous (SA) and asynchronous (A) baselines and the non-federate Classic ML (NF). The best results of the FL approaches are marked in bold. First, the mean average percentage error indicates the oil level time series' prediction performance. The DAAFL is superior in MAPE to all baselines. Second, the three metrics, L-F1, L-P and L-R, refer to the lower threshold regarding F1 score, precision and recall. Our approach has the best values for L-F1 and L-R compared to the baselines. The FedAvg and the Pr-SAFL baseline have the best L-P at the cost of a low recall. Third, the metrics U-F1, U-P and U-R express the results for the upper threshold case. DAAFL excels in both U-F1 and U-R. The FedSA reaches the best U-P, albeit with a weak U-R. Notably, all models exhibit greater proficiency in detecting lower threshold cases than upper threshold cases. The standard deviation remains low for DAAFL and most baselines.

The threshold cases are challenging to detect since the oil level remains primarily constant in the long-term view. Consequently, all approaches struggle to identify substantial oil level fluctuations. We observe that baselines with higher precision than DAAFL only achieve higher precision by neglecting many threshold cases, ultimately leading to poor recall scores. In predictive maintenance, recall is a more critical metric than precision since a false alarm is often preferable to undetected damage. The observation that all models are better at detecting lower than upper threshold cases could stem from models learning the oil level's overarching tendency to decrease. The DAAFL achieves slightly better results than the Classic ML approach regarding the detection of upper threshold cases, indicating the usefulness of DAAFL's client weighting. This scheme can help to prioritise data shares with more helpful information for the learning process and can ultimately lead to a better overall performance. To dispel the notion that DAAFL's success hinges on favourable dropout patterns, we have performed the same experiment for DAAFL without the dropout simulation. The DAAFL achieves even better scores (0.05 pp. MAPE, 0.01 pp L-F1, 0.24 pp U-F1) when all clients are persistently available. We provide further details on this experiment in the technical appendix. Variations in standard deviation among approaches can be attributed to their specific aggregation methodologies. The aggregation that balances client contributions yields a low standard deviation, while reliance on random elements elevates it. In conclusion, the DAAFL performs better than the synchronous and semi-asynchronous approaches and thus combines prediction performance and the flexibility and velocity of AFL schemes.

| Algorithm | FL Type | MAPE [%] | RSME |
|---|---|---|---|
| Classic ML | NF | 09.475±0.537 | 0.090±0.004 |
| FedAvg | S | 11.202±9.282 | 0.102±0.062 |
| FedSA | SA | 10.397±3.858 | 0.095±0.032 |
| Pr-SAFL | SA | 11.844±1.271 | 0.103±0.007 |
| St-AFL | A | 14.074±2.514 | 0.129±0.015 |
| LR-AFL | A | 10.849±0.816 | 0.100±0.004 |
| DAAFL | A | **09.577**±0.580 | **0.088**±0.003 |

Table 4: Prediction results on the jet engine test set. Each experiment was run 30 times. The best federate results are marked in bold.

**Jet Engine Maintenance** Table 4 presents the outcomes for the remaining useful live (RUL) predictions of the jet engines. The DAAFL achieves the best results regarding MAPE and RSME among the federated models. Compared to the non-federated classic machine learning approach, the DAAFL model performs equally well regarding RMSE yet presents a higher MAPE. The relative performance patterns among the baselines align with those documented in Table 3: While the FedSA and LR-AFL emerge as robust baselines, the ST-AFL and PR-SAFL are weak in both experiments.

A possible reason the DAAFL is better than the classic ML regarding the RSME is once more the client weighting. The baselines can not adapt to the federated settings with dropouts and fail to achieve scores close to the non-federated Classic ML model.

## Generalisation to Other Architectures

This section examines the performance of the DAAFL approach regarding different prediction model architectures. In

| Architecture | MAPE [%] | L-F1 [%] / L-P [%] / L-R [%] | U-F1 [%] / U-P [%] / U-R [%] |
|---|---|---|---|
| GRU | **1.51**±0.07 | **89.18**±0.18 / 98.35±0.89 / 81.59±0.61 | **80.87**±0.59 / 85.23±3.28 / 77.06±1.77 |
| LSTM | 1.62±0.07 | 87.98±0.18 / **98.74**±0.89 / 79.41±0.61 | 77.65±0.59 / 91.02±3.28 / 67.84±1.77 |
| TCN | 1.95±0.24 | 83.86±1.41 / 79.21±3.01 / **89.19**±1.03 | 75.94±1.80 / 68.56±4.16 / **85.46**±2.47 |
| Feed Forward | 1.99±0.14 | 87.57±0.05 / 97.38±0.07 / 79.55±0.09 | 71.14±2.84 / **93.70**±2.33 / 57.55±4.34 |

Table 5: Oil dilution prediction performance of DAAFL with respect to the prediction model architecture. The best results are marked in bold.

this experiment, we replace the GRU network described in Section *Machine Learning Model Implementation* with an LSTM, a temporal convolution network (TCN) and a feed-forward (FF) model, respectively. Table 5 displays the results for the DAAFL when repeating the oil dilution experiments with the three new architectures. The GRU network achieves the optimal MAPE and F1 scores for upper and lower threshold cases, followed by the LSTM. While the LSTM has high precision, it comes at a relatively low recall. For the TCN, the high values for upper and lower recall coincide with low precision. The feed-forward model does not achieve high values for MAPE and the upper F1 score but demonstrates strong precision.

Comparing the results for the different architectures to the baseline performances in Table 3, we see that most architectures obtain better F1 scores than the baselines. An exception lies in the UF1 for the feed-forward model, reflecting its simpler architecture in contrast to the GRU employed by the baselines. The networks' notable recall signifies DAAFL's adeptness in identifying existing threshold cases across various architectures, outperforming baselines. In summary, our DAAFL approach has a data disparity aware weighting scheme and is dropout-robust, regardless of the model architecture. DAAFL can be combined with many established architectures, improving over state-of-the-art baselines.

## Related Work

Introduced in 2017 by Google, Federated Learning (FL) has evolved as a data privacy-focused distributed learning paradigm (McMahan et al. 2017). (Yang et al. 2019) and (Kairouz et al. 2021) give an excellent overview. FL has diverse applications in transportation (Li et al. 2022; Wang et al. 2021; Koetsier et al. 2021). In particular, FL benefits from the abundance of transportation sensor data, enabling, e.g., predictive maintenance systems (Manias and Shami 2021; Bemani and Björsell 2022).

*Asynchronous federated learning* (AFL) enables the training of a federated model without requiring all clients to be available at the same time. (Xu et al. 2023). Thus, AFL schemes do not suffer from low round efficiency (Zhang et al. 2021) as synchronous schemes do. The problems of straggler mitigation and dropped-out devices occur particularly frequently in transportation such that AFL is a vital approach for transportation applications (Xu et al. 2022; Lu et al. 2020). However, the data heterogeneity on the clients remains a problem (Vahidian et al. 2023). Various node selection policies have been proposed to overcome device heterogeneity in AFL. (Chen et al. 2021) considers local resources and communication for global aggregation. In contrast, (Wu et al. 2020) focuses on probability-based node selection, prioritising more reliable nodes. (Liu et al. 2021) employs reinforcement learning for node participation. Compared to node selection, aggregation weighting can steer the clients' impact more flexibly on the global model. (Zhang, Bosch, and Olsson 2021) aggregates a node only if it is neither stale nor too up-to-date by comparing the local version numbers versus the global model. (Zhou et al. 2021) considers the time efficiency of the clients when weighing the participating nodes. In contrast to these approaches, our aggregation weighting scheme considers data disparity.

*Semi-synchronous approaches* combine synchronous and asynchronous strategies (Ma et al. 2021; Wu et al. 2020). (Hao, Zhao, and Zhang 2020) introduces a priority function which depends on computing power, communication delay, and data volume but is biased towards clients with large data volumes. (Chen, Mao, and Ma 2021) presents an algorithm where the first stage equals FedAvg while the second stage is asynchronous with individual local epochs. The disadvantage of semi-asynchronous approaches is that the problems with synchronous approaches persist.

Determining a stopping criterion for FL remains a topic requiring attention (Kairouz et al. 2021). Most approaches for heterogeneous clients suggest a fixed number of iterations or communication rounds (Wang et al. 2022). (Jeon et al. 2020) suggest a synchronous stopping criterion which does not apply to asynchronous settings. Therefore, we present an asynchronous stopping mechanism.

## Conclusion

In this paper, we present DAAFL, a data disparity and temporal unavailability aware asynchronous federated learning framework. DAAFL mitigates bias seen in existing asynchronous federated learning algorithms, favouring frequent updates from clients with limited data volumes. The central innovation is the novel aggregation parameter rule, which balances data volume and update frequency versus aggregation weighting. Our approach improves the prediction in two predictive maintenance use cases, surpassing five federated learning baselines. Remarkably, DAAFL closely matches or rivals classical centralised machine learning models.

While we address system heterogeneity for predictive maintenance here, we will extend the approach to other settings. We will explore further asynchronous federated learning aspects in future. Uncovered areas include robustness against attacks, e.g., the potential restoration of private training data from parameter updates.

## References


Bemani, A.; and Björsell, N. 2022. Aggregation Strategy on Federated Machine Learning Algorithm for Collaborative Predictive Maintenance. *Sensors*, 22(16): 6252.

Chen, M.; Mao, B.; and Ma, T. 2019. Efficient and robust asynchronous federated learning with stragglers. In *International Conference on Learning Representations*.

Chen, M.; Mao, B.; and Ma, T. 2021. FedSA: A staleness-aware asynchronous federated learning algorithm with non-IID data. *Future Generation Computer Systems*, 120: 1–12.

Chen, Y.; Ning, Y.; Slawski, M.; and Rangwala, H. 2020. Asynchronous online federated learning for edge devices with non-iid data. In *2020 IEEE International Conference on Big Data (Big Data)*, 15–24. IEEE.

Chen, Z.; Liao, W.; Hua, K.; Lu, C.; and Yu, W. 2021. Towards asynchronous federated learning for heterogeneous edge-powered internet of things. *Digital Communications and Networks*, 7(3): 317–326.

Hao, J.; Zhao, Y.; and Zhang, J. 2020. Time efficient federated learning with semi-asynchronous communication. In *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*, 156–163. New York, NY, USA: IEEE.

Jeon, Y.-S.; Amiri, M. M.; Li, J.; and Poor, H. V. 2020. A compressive sensing approach for federated learning over massive MIMO communication systems. *IEEE Transactions on Wireless Communications*, 20(3): 1990–2004.

Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. 2021. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2): 1–210.

Koetsier, C.; Fiosina, J.; Gremmel, J. N.; Sester, M.; Müller, J. P.; and Woisetschläger, D. 2021. Federated cooperative detection of anomalous vehicle trajectories at intersections. In *Proceedings of the 4th ACM SIGSPATIAL International Workshop on Advances in Resilient and Intelligent Cities*, 13–22. New York, NY, USA: ACM.

Li, T.; Hu, S.; Beirami, A.; and Smith, V. 2021. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning*, 6357–6368. PMLR.

Li, T.; Sahu, A. K.; Talwalkar, A.; and Smith, V. 2020. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3): 50–60.

Li, Y.; Tao, X.; Zhang, X.; Liu, J.; and Xu, J. 2022. Privacy-Preserved Federated Learning for Autonomous Driving. *IEEE Transactions on Intelligent Transportation Systems*, 23(7): 8423–8434.

Liu, Y.; Zhao, R.; Kang, J.; Yassine, A.; Niyato, D.; and Peng, J. 2021. Towards communication-efficient and attack-resistant federated edge learning for industrial internet of things. *ACM Transactions on Internet Technology (TOIT)*, 22(3): 1–22.

Lu, Y.; Huang, X.; Zhang, K.; Maharjan, S.; and Zhang, Y. 2020. Blockchain empowered asynchronous federated learning for secure data sharing in internet of vehicles. *IEEE Transactions on Vehicular Technology*, 69(4): 4298–4311.

Ma, Q.; Xu, Y.; Xu, H.; Jiang, Z.; Huang, L.; and Huang, H. 2021. FedSA: A semi-asynchronous federated learning mechanism in heterogeneous edge computing. *IEEE Journal on Selected Areas in Communications*, 39(12): 3654–3672.

Manias, D. M.; and Shami, A. 2021. Making a case for federated learning in the internet of vehicles and intelligent transportation systems. *IEEE Network*, 35(3): 88–94.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and Arcas, B. A. y. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Singh, A.; and Zhu, J., eds., *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, 1273–1282. Ft. Lauderdale, FL, USA: PMLR.

Saxena, A.; Goebel, K.; Simon, D.; and Eklund, N. 2008. Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 international conference on prognostics and health management*, 1–9. IEEE.

Vahidian, S.; Morafah, M.; Wang, W.; Kungurtsev, V.; Chen, C.; Shah, M.; and Lin, B. 2023. Efficient distribution similarity identification in clustered federated learning via principal angles between client data subspaces. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 8, 10043–10052.

Wang, T.; Cheng, W.; Luo, D.; Yu, W.; Ni, J.; Tong, L.; Chen, H.; and Zhang, X. 2022. Personalized federated learning via heterogeneous modular networks. In *2022 IEEE International Conference on Data Mining (ICDM)*, 1197–1202. IEEE.

Wang, W.; Fida, M. H.; Lian, Z.; Yin, Z.; Pham, Q.-V.; Gadekallu, T. R.; Dev, K.; and Su, C. 2021. Secure-Enhanced Federated Learning for AI-Empowered Electric Vehicle Energy Prediction. *IEEE Consumer Electronics Magazine*, 1–1.

Wu, W.; He, L.; Lin, W.; Mao, R.; Maple, C.; and Jarvis, S. 2020. SAFA: A semi-asynchronous protocol for fast federated learning with low overhead. *IEEE Transactions on Computers*, 70(5): 655–668.

Xu, C.; Qu, Y.; Luan, T. H.; Eklund, P. W.; Xiang, Y.; and Gao, L. 2022. An Efficient and Reliable Asynchronous Federated Learning Scheme for Smart Public Transportation. *IEEE Transactions on Vehicular Technology*.

Xu, C.; Qu, Y.; Xiang, Y.; and Gao, L. 2023. Asynchronous federated learning on heterogeneous devices: A survey. *Computer Science Review*, 50: 100595.

Yang, Q.; Liu, Y.; Chen, T.; and Tong, Y. 2019. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2): 1–19.

Yang, Z.; Zhang, X.; Wu, D.; Wang, R.; Zhang, P.; and Wu, Y. 2022. Efficient Asynchronous Federated Learning Research in the Internet of Vehicles. *IEEE Internet of Things Journal*, 10(9): 7737–7748.

Zhang, H.; Bosch, J.; and Olsson, H. H. 2021. Real-time end-to-end federated learning: An automotive case study. In *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, 459–468. New York, NY, USA: IEEE.

Zhang, Q.; Gu, B.; Deng, C.; and Huang, H. 2021. Secure bilevel asynchronous vertical federated learning with backward updating. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 12, 10896–10904.

Zhou, C.; Tian, H.; Zhang, H.; Zhang, J.; Dong, M.; and Jia, J. 2021. TEA-fed: time-efficient asynchronous federated learning for edge computing. In *Proceedings of the 18th ACM International Conference on Computing Frontiers*, 30–37.