Collaborative Training of Data-Driven Remaining Useful Life Prediction Models Using Federated Learning

Wilhelm Söderkvist Vermelin[1], Madhav Mishra[2], Mattias P. Eng[3], Dag Andersson[4], and Konstantinos Kyprianidis[5]

[1, 2, 3, 4] *RISE Research Institutes of Sweden, Mölndal, Västra Götaland, 431 53, Sweden*
wilhelm.soderkvist.vermelin@ri.se
madhav.mishra@ri.se
mattias.eng@ri.se
dag.andersson@ri.se

[1, 5] *Mälardalen University, Västerås, Västmanland, 722 18, Sweden*
konstantinos.kyprianidis@mdu.se

ABSTRACT

Remaining useful life prediction models are a central aspect of developing modern and capable prognostics and health management systems. Recently, such models are increasingly data-driven and based on various machine learning techniques, in particular deep neural networks. Such models are notoriously "data hungry", i.e., to get adequate performance of such models, a substantial amount of diverse training data is needed. However, in several domains in which one would like to deploy data-driven remaining useful life models, there is a lack of data or data are distributed among several actors. Often these actors, for various reasons, cannot share data among themselves. In this paper a method for collaborative training of remaining useful life models based on federated learning is presented. In this setting, actors do not need to share locally held secret data, only model updates. Model updates are aggregated by a central server, and subsequently sent back to each of the clients, until convergence. There are numerous strategies for aggregating clients' model updates and in this paper two strategies will be explored: 1) federated averaging and 2) federated learning with personalization layers. Federated averaging is the common baseline federated learning strategy where the clients' models are averaged by the central server to update the global model. Federated averaging has been shown to have a limited ability to deal with non-identically and independently distributed data. To mitigate this problem, federated learning with personalization layers, a strategy similar to federated averaging but where each client is allowed to append custom layers to their local model, is explored. The two federated learning strategies will be evaluated on two datasets: 1) run-to-failure trajectories from power cycling of silicon-carbide metal-oxide semiconductor field-effect transistors, and 2) C-MAPSS, a well-known simulated dataset of turbofan jet engines. Two neural network model architectures commonly used in remaining useful life prediction, long short-term memory with multi-layer perceptron feature extractors, and convolutional gated recurrent unit, will be used for the evaluation. It is shown that similar or better performance is achieved when using federated learning compared to when the model is only trained on local data.

Wilhelm Söderkvist Vermelin et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 United States License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.
https://doi.org/10.36001/IJPHM.2024.v15i2.3821

1. INTRODUCTION

1.1. Background

A central aspect of robust and reliable preventive maintenance systems is the ability to predict the failure of a component ahead of time. This is often referred to as *predictive maintenance* (PdM)(Meriem, Nora, & Samir, 2023). One such way to predict the end-of-life of a component is to develop systems that can predict the remaining useful life (RUL) of a component. The RUL is defined as the remaining time (measured in some appropriate unit) a system has left until it has failed (Pecht & Kang, 2018). Here failure does not necessarily mean critical failure, which is when the system has failed irreversibly such that one or several subcomponents are broken beyond repair. Failure in this context can also mean either that system degradation has progressed beyond the point that it cannot reliably deliver the specified functionality, or that some measured value has exceeded (or fallen below) some specification threshold. RUL prediction models are either physical; based on mathematical modeling of the physics of the system, data-driven; meaning that the model is developed using data

collected from the system, or hybrid; which means that the model uses some combination of the two former. This paper focuses on data-driven models, specifically deep learning models, a type of machine learning models, designed for the prediction of RUL.

Machine learning (and especially deep learning models) can achieve good performance on a wide range of tasks, given sufficient access to diverse training data (Hestness et al., 2017). However, in many prognostics problems data availability is an issue, limiting the applicability of deep learning models. Another challenge is that when data are available, they are often distributed among several actors. For decentralized and heterogeneous data, it is common that data are sensitive and not possible to share with a third party for prognostics model development. When data sharing is not viable among actors it can hinder the use of machine learning as each individual actor may have insufficient data in the sense of dataset size and diversity. In this scenario, cooperative training of the machine learning model can serve as a means for overcoming the problem of sharing sensitive data. Cooperative training of machine learning models can be achieved through *Federated Learning* (FL), a decentralized machine learning training method. Despite the growing interest in FL, owing much to its privacy-enhancing features as reported in Kairouz et al. (2021), its adoption in the industrial domain, which includes many PHM applications, is still relatively slow. This paper will evaluate FL for PHM in two cases of RUL prediction; simulated turbofan jet engines, and power electronics.

1.2. Federated Learning

FL is a framework for decentralized training of machine learning models. FL is used when, for various reasons, data cannot be centralized at a single location, meaning traditional machine learning training is not possible. The basic structure of FL involves a central *server* and a group of clients forming a so-called *federation*. Each client has a local private partition of data referred to as $\mathcal{P}_k$, where k is one of K clients. The size of the local partition is $\|\mathcal{P}_k\| = n_k$. The full global dataset is denoted as $\mathcal{S}$ such that $\mathcal{S} = \{\mathcal{P}_k\}_{k=1}^{K}$. In the first step, the global model is initialized with random parameters (referred to as weights), w_0, and then communicated to each client. Then, each client trains their model on their own local, private, data ($\mathcal{P}_k$) for E epochs, and thus obtains an updated local model. The updated local models are then communicated back to the central server, which in turn aggregates all updated models from all clients which forms the new global model. Finally, the server communicates the new global model to all clients. This is repeated until the global model has converged. FL has the advantage that no client explicitly has to share their local data, which enhances privacy, and has been used at scale in several domains with good success, as reported in Q. Li et al. (2023).

The algorithm governing aggregation of local updates at the central server is called the FL *strategy*. There are numerous notable strategies, which have different benefits depending on the dataset, model, and objectives (Moshawrab, Adda, Bouzouane, Ibrahim, & Raad, 2023). The first strategy, presented by the original authors of FL, is called *federated averaging* (FedAvg)(McMahan, Moore, Ramage, Hampson, & Arcas, 2017). In this strategy, model updates from the clients communicated to the server are aggregated by averaging the results. The pseudocode for this algorithm is presented in Algorithm 1.

Algorithm 1 Federated Averaging (FedAvg) algorithm. There are K clients indexed by k, m is the batch size and E is the number of local epochs. n_k is the size of the local data partition $\mathcal{P}_k$, $\mathcal{L}(w; b)$ is the loss function for optimizing the model, and η is the learning rate.

Server executes:
Initialize w_0
for each round $t = 1, 2, \ldots, T$ **do**
  $s \leftarrow \max(C \times K, 1)$
  $S_t \leftarrow$ (random set of s clients)
  for each client $k \in S_t$ **in parallel do**
    $w_t^k \leftarrow \text{CLIENTUPDATE}(w_{t-1}^k, k)$
    $s_t \leftarrow \sum_{k \in S_t} n_k$
    $w_t \leftarrow \sum_{k \in S_t} \frac{n_k}{s_t} w_t^k$
  endfor
end for

function CLIENTUPDATE(w, k) ▷ Run on client k
  $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size m)
  for each local epoch i to E **do**
    for each batch $b \in \mathcal{B}$ **do**
      $w \leftarrow w - \eta \nabla \mathcal{L}(w; b)$
    end for
  end for
  return w
end function

The FedAvg strategy is a good baseline strategy and performs well on a wide range of problems and datasets. However, the FedAvg strategy is known to be sensitive to highly non-Independent and Identically Distributed (non-IID) data. This poses a potential issue when applied to RUL prediction as it is not uncommon that clients' assets operate under vastly different operating conditions, leading to local data partitions with distributions that are not representative of the population distribution. To ameliorate the performance of FedAvg on non-IID data, several strategies have been proposed. For example, a strategy referred to as Federated Learning with Personalization Layers (FedPer), initially introduced and described by Arivazhagan, Aggarwal, Singh, and Choudhary (2019). This strategy is similar to FedAvg but with the addition that each client can append custom (personalized) layers to the neural network model which are not communicated with the central server. The weights in the personalized layers for the k:th

client are denoted as w_{P_k}, see Algorithm 2. In this paper, the FedPer algorithm will be implemented as originally described, without any major adaptations.

Algorithm 2 Federated Learning with Personalization Layers (FedPer) algorithm. There are K clients indexed by k, m is the batch size and E is the number of local epochs. n_k is the size of the local data partition $\mathcal{P}_k$, $\mathcal{L}(w; b)$ is the loss function for optimizing the model, and η is the learning rate.

Initialize each client's personal weights:
for each client $k = 1, 2, \ldots, K$ **do**
  Initialize personal weights $w_{P_k,0}$
end for

Server executes:
Initialize base weights $w_{B,0}$
for each round $t = 1, 2, \ldots, T$ **do**
  $s \leftarrow \max(C \times K, 1)$
  $S_t \leftarrow$ (random set of s clients)
  for each client $k \in S_t$ **in parallel do**
    $w^k_{B,t} \leftarrow \text{CLIENTUPDATE}(w^k_{B,t-1}, k)$
    $s_t \leftarrow \sum_{k \in S_t} n_k$
    $w_{B,t} \leftarrow \sum_{k \in S_t} \frac{n_k}{s_t} w^k_{B,t}$
  endfor
end for

function CLIENTUPDATE(w, k) ▷ Run on client k
  $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size m)
  for each local epoch i to E **do**
    for each batch $b \in \mathcal{B}$ **do**
      $(w_B, w_{P_k}) \leftarrow (w_B, w_{P_k}) - \eta \nabla \mathcal{L}(w_B, w_{P_k}; b)$
    end for
  end for
  return w_B
end function

The FedPer algorithm is also presented visually in Figure 1. In the figure, each model consists of a "backbone" and a prediction "head". The backbone is the foundation of the model and shared among all clients, whereas the prediction head constitute the personalized layers. In the FedPer strategy, only model backbones are communicated to the server, prediction heads are not shared and stay at each client.

1.3. Related Work

To explore what research has been performed in the area of federated learning for PHM and RUL prediction, a small-scale literature review has been conducted using the Elsevier Scopus database. To find relevant documents, the following search string was used

```
TITLE-ABS-KEY({federated learning})
AND (TITLE-ABS-KEY ("prognos*")
OR TITLE-ABS-KEY("pred* maint*"))
AND TITLE-ABS-KEY ({remaining useful
life})
```

This search yielded nine results, where seven were published in 2023, and the two other were published in 2021 and 2022,

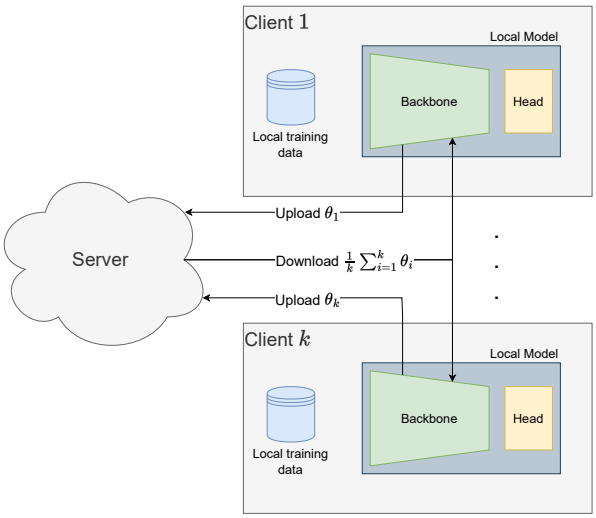


Figure 1. A visual representation of the FedPer algorithm. Each client's model consists of a backbone and a head. The FedPer algorithm relaxes the constraint that the entire model is communicated to the server, and in this protocol only the model backbone is shared.

respectively. Some of the retrieved documents will be discussed below.

Although the research area has not received a lot of attention so far (only nine search results), there is clear evidence for the potential of federated learning in various applications related to PHM, predictive maintenance, and remaining useful life prediction. For example,Dhada, Parlikad, and Palau (2020) proposes the use of federated learning for predicting failures in a simulated turbofan fleet, showing the effectiveness of the (FedAvg) algorithm.

In Arunan, Qin, Li, and Yuen (2023), the authors address data heterogeneity among edge devices, stemming from dissimilar degradation mechanisms and unequal dataset sizes, which poses a critical statistical challenge for developing accurate FL-models. To mitigate this problem the authors propose a FL-based health prognostic model which feature a similarity matched parameter aggregation algorithm.

Similarly,Vibhorpandhare, Jia, and Lee (2021) use federated learning to collaboratively train a federated learning model based on Gaussian mixture models.

The article authored by Bemani and Björsell (2022) presents two FL algorithms for predictive maintenance applications, evaluated using C-MAPSS. The federated learning methods demonstrate enhanced accuracy in anomaly detection and RUL prediction compared to traditional centralized approaches, while preserving privacy via local training.

One paper, by Kamei and Taghipour (2023), explores alternative federated learning strategies to mitigate data heterogeneity

in clients' local data partitions. In particular, the authors use the so called FedProx strategy, first developed in Sahu et al. (2018). The article analyzes prognostics approaches for networks of centralized assets by utilizing sensor data. The paper compares a localized model with both FedAvg and FedProx and using both LSTM and Transformer (Vaswani et al., 2017) architectures to predict RUL, for both centralized and decentralized scenarios using C-MAPSS data, with the Transformer architecture outperforming LSTM overall. The paper explores the effect of the number of clients in the federation on model performance. A separate federation per CMAPSS data subsets (FD001, FD002, FD003, FD004) is evaluated, and the model is trained using one (central training), two, five, and ten clients in the federation. The authors conclude that model performance only slightly degraded in the federated setting, showing that FL is promising for RUL prediction. Their main contribution is showing that FL can still be effective while the number of clients is large (and therefore, the local datasets are small).

In the paper by Du et al. (2023), the authors also investigate the transformer architecture and evaluated its performance for RUL prediction on the C-MAPSS dataset. In addition, the authors proposed a hyperparameter search scheme based on Bayesian optimization.

In the paper by Guo et al. (2023), the authors propose several FL strategies to train feature extraction models based on convolutional autoencoders (CAE). The CAEs are trained to extract low-level features from the data and subsequently each client trains their own local RUL prediction model using local data. In this way only the CAE are trained using FL and the RUL prediction models are trained locally.

In the paper by Abdelli, Cho, and Pachnicke (2021) the authors focused on the collaboration aspect of federated learning, and applied federated learning to RUL estimation of a laser device.

During the development of this manuscript, a paper by X. Chen, Wang, Lu, Xu, and Yan (2023) was published, exploring an idea similar to FedPer for PHM. In this paper, the authors develop a RUL prediction model based on the C-MAPSS dataset, where each clients' model has two parts, a feature extractor (which they refer to as a "local health degradation representation" (LHDR)) and a "unique super-structure module". The LHDR is trained in a federated manner while the unique super-structure module are kept private at each client and not shared with the rest of the federation. This strategy is similar to FedPer in that the "backbone" of the model is shared within the federation, but personalized layers, also called model head, is not shared with the federation. The paper also has a similar setup where clients in the federation are the different subsets of the C-MAPSS dataset. Their paper is limited to investigations on the C-MAPSS dataset, whereas this paper extends the investigation to a dataset from a completely different domain. As this paper also deals with PHM for electronics, a corresponding focused literary analysis was conducted to investigate the current state of research within this area.

Regarding PHM for electronics, there has been some work on PHM for MOSFETs. For example,Ren et al. (2022), used a Long Short-Term Memory (LSTM) model to predict the RUL of thermally stressed MOSFETs. In particular, their approach was evaluated on the public dataset for MOSFET prognostics, introduced in Celaya, Saxena, Saha, and Goebel (2011). Similarly,Haris, Hasan, Jahanzeb Hussain Pirzada, and Qin (2020) applied a Bayesian optimized LSTM for prognostics of thermally aged MOSFETs. In Demus et al. (2019), the authors obtained MOSFET junction temperature measurements and trained a support vector machine to predict the junction temperature, as a proxy for system degradation. However, research on PHM for electronics in a federated setting is lacking.

1.4. Contribution

The main contributions of this paper are twofold.

Firstly, this paper addresses federated learning in a PHM setting where the clients are expected to have non-IID data. To explore and mitigate this challenge this paper evaluates the performance of the traditional FedAvg aggregation strategy in compared to the FedPer strategy. To the authors' knowledge, there has been few or no works exploring personalization layers in federated learning for PHM.

Secondly, this paper investigates federated learning and PHM for electronics, in particular, power electronics in the form of SiC MOSFET devices, an area that has not been extensively explored earlier.

1.5. Datasets

This paper will explore the scenario where the actors (or clients) are companies that have safety critical components that need to be monitored to ensure that they are operating as expected. Specifically, two RUL prediction problems will be addressed; accelerated aging of silicon carbide (SiC) metal-oxide-semiconductor field-effect transistors (MOSFETs) and turbofan jet engines via the C-MAPSS dataset as introduced by Saxena, Goebel, Simon, and Eklund (2008). In each case, the scenario where a company has a local dataset with some number of assets operating under various operating conditions is simulated. In this scenario, each company is interested in modeling the RUL of their assets using deep neural networks, but no company is willing to share data outside their organization. However, they are willing to share models trained on their local data, making the FL strategy an optimal solution. This paper will explore if there are any performance gains when collaboratively training the RUL prediction model using federated learning, versus limiting model training to locally held data.

1.5.1. SiC MOSFET Accelerated Aging Dataset

As complexity in modern electronic devices increases, the risk of failure is also increasing. To ensure reliability and safety of systems and components, monitoring of electronics is becoming more and more important. However, gathering health monitoring data from electronic devices in real-world settings has been linked with inherent risks and substantial expenses. In an effort to mitigate this challenge and stimulate research in the field of PHM for electronic devices, a dataset comprising the degradation histories of 33 wire-bonded SiC MOSFETs subjected to accelerated aging by power cycling until failure was meticulously created. To create a diverse dataset, the SiC MOSFETs were cycled at different currents and on-times (operating conditions), leading to a varying degree of aging rate and hence substantial variation in degradation and life length. More details on this dataset can be found in the paper describing the dataset and its use in data-driven RUL prediction:Söderkvist Vermelin, Lövberg, Misiorny, P. Eng, and Brinkfeldt (2023).

Devices Under Test (DUTs), i.e., the SiC MOSFETs, were cycled until one of the two failure conditions were met:

- measured ON-state voltage is higher than 8 V, or
- recorded MOSFET casing temperature is higher than 100 °C.

The settings for experiment rounds one through five are compiled in Table 1.

Exp. no. (#)	Current (A)	DUTs (#)	ON (s)	OFF (s)
1	25 A	5	10	10
2	23 A	10	10	10
3	24 A	10	10	10
4	28 A	8	10	10
5	25 A	10	15	10

Table 1. SiC MOSFETs dataset, indexed by experiment number.

Recorded data from a typical on/off-cycle, represented by a DUT in round 1, is shown in Figure 2.

Key characteristics for were extracted from the data, resulting in a final dataset consisting of 18 features recorded per cycle. For example, the *cycle count* is the number of cycles the DUT has been cycled and the *life percentage* is the fractional life left until failure (bounded in the interval $[0, 1]$).

Life Percentage (LP) is defined as the amount of life consumed at each point in time:

$$\mathrm{LP}(t) = \frac{t}{T}, \tag{1}$$

such that at time $t = 0$ $\mathrm{LP}(t = 0) = 0$, and at end-of-life $t = T$, $\mathrm{LP}(t = T) = 100\%$. LP is related to RUL (measured in time/cycles) according to

$$\mathrm{RUL}(t) = \frac{t}{\mathrm{LP}(t)} - t \tag{2}$$

where at $t = 0$, $\mathrm{RUL}(t = 0) = T$. Life percentage is also sometimes referred to as lifetime percentage (Ben Ali, Chebel-Morello, Saidi, Malinowski, & Fnaiech, 2015; Lallart, Wu, Li, & Qiu, 2017; Mahamad, Saon, & Hiyama, 2010; Singh, Darpe, & Singh, 2020; Tian, 2009; Tian, Wong, & Safaei, 2010; Wu, Gebraeel, Lawley, & Yih, 2007; Xia et al., 2019; Zhu, Chen, & Shen, 2020). LP is often used for bearing RUL prediction, but has also been used for railway switches (Q. Chen, Nicholson, Ye, Zhao, & Roberts, 2020), and crack-growth in aluminum lugs (T. Li, 2024).

Life percentage can then be translated into so-called *Remaining Useful Life Percentage* (RULP) (Huang, Zhu, Han, & Peng, 2022; Xu, Duan, Chen, Wang, & Fan, 2022) which is defined as

$$\mathrm{RULP}(t) = 1 - \mathrm{LP}(t). \tag{3}$$

LP and RULP prediction, are well established niches in the PHM literature, often used for assets with high variability in life spans and life lengths that span multiple orders of magnitude.

For a detailed description of each feature please refer to the corresponding paper authored by Söderkvist Vermelin et al. (2023). A summary of the derived features is shown in Table 2. The dataset of derived features was used to train a model on the RULP prediction task.

The degradation progression is shown in Figure 3. In this figure, the initial voltage at each cycle is shown, for each DUT in round 1.

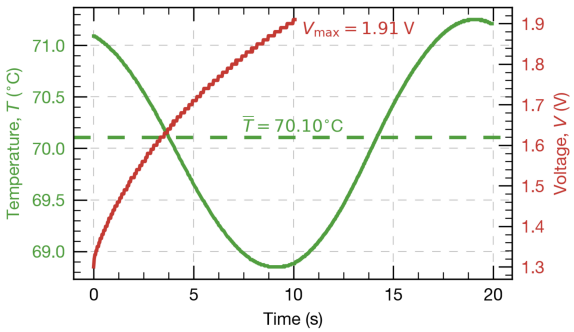


Figure 2. A cycle of data from power cycling, at steady-state and significant degradation has yet to occur. The MOSFET-case temperature is shown in green, and the voltage is shown in red. The average temperature is showed as a vertical dashed line. The cycle has an initial ON-time of 10 seconds where voltage is increasing, and a subsequent OFF-time for 10 seconds (no current is passed and hence no voltage).

Feature	Unit
Cycles	unitless
Life percentage	unitless
End voltage	V
End resistance	mΩ
Residual resistance	mΩ
Cleaned residual resistance	mΩ
Min temperature	°C
Max temperature	°C
Min block temperature 1	°C
Max block temperature 1	°C
Min block temperature 2	∘C
Max block temperature 2	°C
Min water inlet temperature	°C
Max water inlet temperature	°C
Min water outlet temperature	°C
Max water outlet temperature	°C
Mean block temperature	°C
End resistance from mean block temperature	mΩ
Residual end resistance from mean block temperature	mΩ
Cleaned residual end resistance from mean block temperature	mΩ

Table 2. Derived features dataset.

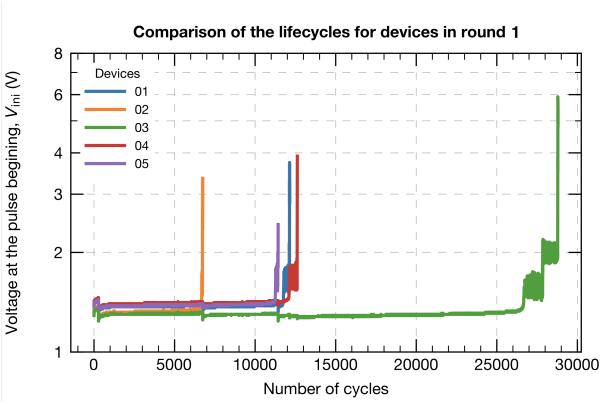


Figure 3. The plot shows the first recorded voltage in each cycle for each DUT in round 1.

For training the machine learning models, devices from each round is randomly selected for training and testing, such that approximately 80 % is allocated to the training dataset, and the rest to the test dataset. In Table 3, the training/testing split, operating conditions and fault modes are shown.

Dataset	Train	Val.	Test	Op. Con.	Fault modes
Round 1	2	1	1	1	Wire-bond lift-off
Round 2	6	2	2	1	Wire-bond lift-off
Round 3	6	2	2	1	Wire-bond lift-off
Round 5	6	2	2	1	Wire-bond lift-off

Table 3. SiC MOSFET dataset details. Op. Con. means "operating condition". Wire-bond lift-off as failure mechanism has not been experimentally verified, but the collected data indicates this fault mode.

When assuming LP is zero at time zero (or equivalently, RULP is one at time zero), one is implicitly assuming there is no initial wear in the devices. Note that RULP prediction is only performed for the SiC MOSFET dataset, whereas RUL prediction is used for the C-MAPSS dataset. The reason no initial wear in the SiC MOSFETs is assumed is because the devices are unused and recently manufactured at the start of the power cycling experiments. Of course, there could be manufacturing defects, and other problems with the devices before they are used, which may affect the life span. Prior to some of the power cycling experiments, a subset of the devices have been analyzed using a Scanning Acoustic Microscope (SAM), to assess the die attach quality in the chips. The hypothesis is that the die attach quality would shorten the life span, however, the results did not show any strong connection between die attach quality and life span. If there had been a link between die attach quality and life length, one could account for it such that initial RULP would be slightly less than one. In general, it is hard to know a priori what will affect the life length of the devices, and as such additional research is needed to assess the "initial wear" of SiC MOSFETs prior to use. In light of this, and the fact that the devices are unused and newly manufactured at the start of experiments LP is *defined* as zero at time zero for each device. In this case, RULP is interpreted as the device specific initial life capacity, going from 100% to 0% as time progresses. When viewed in this way, the initial wear can be included in the initial life capacity, since it does not stem from operation, but external factors.

1.5.2. C-MAPSS Dataset

The Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) dataset, first presented and developed in Saxena et al. (2008), consists of full run-to-failure trajectories using simulated degradation of turbofan jet engines. The dataset consists of flight cycles with recordings from 24 sensors. In addition, it comprises four cohorts of turbofan engines (referred to as *subsets*) and contain various operating conditions and fault modes. The authors have split the data into training and testing data, where the first are used to develop the models and the latter are used to evaluate the model performance. A summary of the data is shown in Table 4.

Dataset	Train	Val.	Test	Op. Con.	Fault modes
FD001	80	20	100	1	HPC degradation
FD002	208	52	259	6	HPC degr.
FD003	80	20	100	1	HPC and fan degr.
FD004	199	50	248	6	HPC and fan degr.

Table 4. C-MAPSS dataset details. Here HPC means high pressure compressor.

For the C-MAPSS dataset, RUL is measured in actual cycles. Due to low initial degradation in the engines, it is common to clip RUL to a certain max value. The most common choices

are setting maximum RUL to 120 or 130 cycles (Doulamis et al., 2020; Zheng, Ristovski, Farahat, & Gupta, 2017). The RUL of the engines in the test dataset remain unchanged, RUL clipping is only performed on the training and validation datasets.

2. METHODOLOGY

In this section the methodology for training and evaluating the models, both using federated learning and local, is described.

2.1. Data-Driven Remaining Useful Life Prediction

RUL prediction involves finding a mapping between sensor measurements $\mathcal{S}$ and RUL, y,

$$f : \mathcal{S} \mapsto y. \tag{4}$$

Here $\mathcal{S}$ is a time series dataset meaning that it is a collection of multivariate sensor readings. $\mathcal{S}$ consists of multiple sub-datasets for each asset $i = 1, 2, \ldots, N$, where N is the total number of assets in the dataset, such that

$$\mathcal{S} = \{\mathcal{D}_i\}_{i=1}^{N}. \tag{5}$$

$\mathcal{D}_i = \{\mathbf{X}_t^{(i)}\}_{t=0}^{T_i}$ where $\mathbf{X}_t^{(i)}$ are s sensor readings at time t between starting time $t = 0$ and end time $t = T_i$ for asset i, where $i = 1, 2, \ldots, N$. T_i is also referred to the *end-of-life* or *life-span* for asset i. This means for each asset indexed by i that $\mathbf{X}_t^{(i)} \in \mathbb{R}^s$ for $t = 0, 1, \ldots, T_i$. When the sensor measurements in $\mathcal{S}$ are run-to-failure histories, the mapping f in Equation (4) can be learned using supervised machine learning. This means that a new dataset $\mathcal{T}$ can be constructed, containing tuples of sensor measurements and the corresponding remaining useful life, for each asset:

$$\mathcal{T} = \left\{ \left(\mathbf{X}_t^{(i)}, y_t^{(i)} \right)_{t=0}^{T_i} \right\}_{i=1}^{N}. \tag{6}$$

Here, again $\mathbf{X}_t^{(i)} \in \mathbb{R}^s$ is an s-dimensional vector containing sensor measurements, and $y_t^{(i)}$ is the RUL at time t for asset i. RUL can be scaled such that for each asset i, $y_t^{(i)} \in [0, 1]$ for $t = 0, 1, \ldots, T_i$, with $y_0^{(i)} = 1$ and $y_{T_i}^{(i)} = 0$. This is often done for assets that have long lifespans that cover several orders of magnitude (as is the case for the devices in the SiC MOSFET dataset). When this scaling is performed, the new quantity is referred to as remaining useful life percentage (RULP) (Huang et al., 2022; Xu et al., 2022), closely related to the concept of life percentage (LP). Life percentage is mentioned in the derived features summary, cf. Table 2. RULP and RUL measured in actual time or cycles can be transformed using the below formula:

$$y_{\mathrm{T};\,t}^{(i)} = t \frac{y_{\mathrm{S};\,t}^{(i)}}{1 - y_{\mathrm{S};\,t}^{(i)}}, \tag{7}$$

where $y_{\mathrm{T};\,t}^{(i)}$ is RUL, i.e., $y_{\mathrm{T};\,t}^{(i)} \in [0, T_i]$ and $y_{\mathrm{S};\,t}^{(i)}$ is RULP, i.e., $y_{\mathrm{S};\,t}^{(i)} \in [0, 1]$. At $t = 0$, RUL is T_i, i.e., $y_{\mathrm{T};\,t=0}^{(i)} = T_i$ and Equation (7) cannot be used. For the rest of the text, all descriptions and discussions regarding RUL will apply to both RUL and RULP, unless explicitly stated. For the C-MAPSS dataset, it is conventional to work in terms of RUL, and this paper will adhere to this convention. For the MOSFET dataset, it is more convenient to work in terms of LP/RULP since the devices have so large life spans, covering many orders of magnitude.

If the mapping f is parameterized by θ, the supervised learning problem can be formulated as follows:

$$\min_{\theta} \ell\left(\mathbf{X}, \mathbf{y}; \theta\right), \tag{8}$$

where $\mathbf{X}$ and $\mathbf{y}$ is shorthand notation for the full sensor dataset and RUL, for all times and all assets; $\mathbf{X} = \{\{X_t^{(i)}\}_{t=0}^{T_i}\}_{i=1}^{N}$, $\mathbf{y} = \{\{y_t^{(i)}\}_{t=0}^{T_i}\}_{i=1}^{N}$, respectively.

To train the data-driven models, in a classical non-federated setting, the optimization problem in (8), is approximately solved using gradient descent

$$\theta_{k+1} = \theta_k - \eta \nabla_\theta \ell\left(\mathbf{X}, \mathbf{y}; \theta_k\right). \tag{9}$$

and when θ_k are the parameters in a neural network, this algorithm is referred to as backpropagation (Goodfellow, Bengio, & Courville, 2016).

2.2. Model Architectures

In this paper two model architectures are explored, one for each dataset. The application of deep neural networks on the C-MAPSS dataset has been studied extensively, and several architectures have been proposed. One such architecture is described in the paper by Chaoub, Voisin, Cerisara, and Iung (2021), the current state-of-the-art architecture for RUL prediction on the C-MAPSS dataset. The proposed architecture is called MLP-LSTM-MLP, consisting of a multilayer perceptron (MLP), a Long Short-Term Memory recurrent neural network (Hochreiter & Schmidhuber, 1997), and finally another MLP outputting the predicted RUL. This model will be described in detail in Section 2.2.1.

For the MOSFET dataset, the Convolutional Gated Recurrent Unit (ConvGRU)(Ballas, Yao, Pal, & Courville, 2016) neural network model is used.

2.2.1. MLP-LSTM-MLP Model

The MLP-LSTM-MLP model described in the paper by Chaoub et al. (2021) has two basic building blocks; the MLP and the LSTM. An MLP consists an input layer, a number of fully connected hidden layers, and an output layer. It is one of the fundamental model architectures, the simplest instance of a neural network with hidden layers.

The LSTM is a type of Recurrent Neural Network (RNN) that is capable of learning long-term dependencies in data. It is commonly used for natural language processing and speech recognition tasks, where the context of the current input depends on the previous inputs.

The basic building block of an LSTM cell is a memory state that keeps track of the information from previous time steps. This memory state is updated at each time step based on the new input and the previous memory state. The update rule takes into account the importance of the previous memory state and the new input, which is determined by a gating mechanism.

There are three main gates in an LSTM cell: the input gate, the forget gate, and the output gate. The input gate controls how much new information from the current input should be passed to the memory state. The forget gate decides how much of the previous memory state should be retained or forgotten. The output gate controls how much of the updated memory state should be passed to the next time step.

The LSTM model has a series of connected cells, where each cell processes a portion of the input sequence. The outputs of the cells are combined to produce the final output of the model.

One of the key advantages of LSTMs over traditional RNNs is their ability to learn long-term dependencies by allowing information to persist in the memory state for longer periods. This enables them to better capture the context of the current input and make more accurate predictions. In addition, LSTMs are capable of learning which information to retain and which to discard through the gating mechanism.

LSTMs have been applied extensively within the area of RUL prediction, and in particular the C-MAPSS dataset, see e.g.,Al-Dulaimi, Zabihi, Asif, and Mohammadi (2019); Che, Wang, Fu, and Ni (2019); Wang, Wen, Yang, and Liu (2019); Zhang, Wang, Yan, and Gao (2018).

The MLP-LSTM-MLP model is, as the name suggests, an MLP, LSTM, and finally MLP in sequence. The first MLP takes as input the sensor readings at time t for asset i: $\mathbf{x}^{(i)}(t)$, and consists of a series of hidden layers, applying nonlinear mappings of the input, and finally an output layer of some dimensionality, d_{out}. The output of the first MLP is fed into the LSTM model which returns the updated hidden state of the LSTM at each time step h_t. The hidden state at each time step is fed into the final MLP which has an one-dimensional output layer, interpreted as the predicted RUL at time t: $\hat{y}^{(i)}(t)$. A full description of the model is presented in Chaoub et al. (2021). The MLP-LSTM-MLP model is shown in Figure 4.

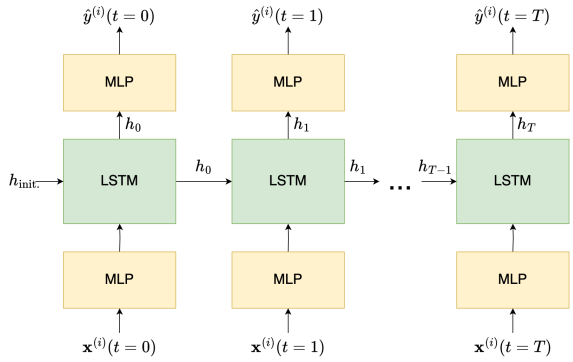


Figure 4. An overview of the MLP-LSTM-MLP model architecture. The sensor readings at time t for asset i ($\mathbf{x}^{(i)}(t)$) is fed into the first MLP. The output of the first MLP is fed into the LSTM, which updates its hidden state. The hidden state is fed into the final MLP which outputs the predicted RUL.

2.2.2. Convolutional Gated Recurrent Unit Model

The Convolutional Gated Recurrent Unit (ConvGRU) model consists of multiple GRU layers, an encoder, and an output layer.

The encoder takes in a slice of time series data (with a predetermined sequence length) and applies a series of convolutional operations to reduce the dimensionality of the data while preserving its important features. After each convolution in the encoder, the output is passed through the Exponential Linear Unit (ELU) activation function (Clevert, Unterthiner, & Hochreiter, 2016) to introduce nonlinearity into the feature extraction procedure. The ELU activation function is defined as follows:

$$\text{ELU}(x;\alpha) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha\left(\exp(x) - 1\right) & \text{if } x \leq 0 \end{cases}, \tag{10}$$

where α was chosen such that $\alpha = 1$.

The output of the encoder is then fed into the GRU layers. The GRU layers consists of several GRU cells, each of which represents one layer in the model. Each GRU cell takes in an input tensor and returns an output tensor based on its internal state. The cells are connected in a way that allows them to update their internal states and outputs as they receive new input data.

The output layer is a fully connected feed-forward neural network with one hidden layer, which takes the output from the GRU layers as input. The hidden activation is the ReLU activation function (Agarap, 2018), defined as:

$$\text{ReLU}(x) = \max(0, x). \tag{11}$$

The output from the output layer is the predicted RULP (a single value). The architecture of the ConvGRU model is

illustrated in Figure 5.

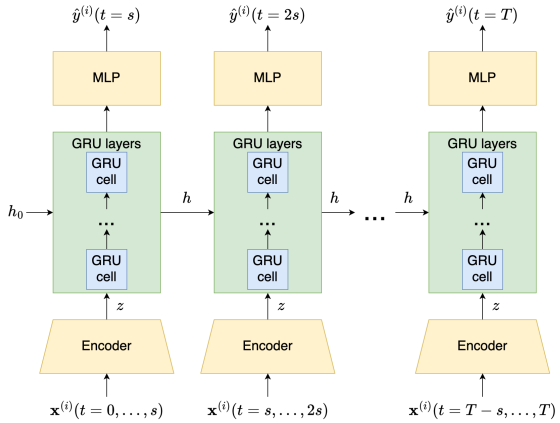


Figure 5. The ConvGRU model architecture. x represents the input time series data. For each sequence, in the time series, the convolutional encoder is applied. The encoded sequence z is then fed into the GRU layers, which can consists of several GRU cells. In this figure, the sequence length is s. For each sequence, the RULP is predicted for the last time step of the sequence. In the final step, the last RULP is predicted.

The ConvGRU has a number of hyperparameters specific to the architecture. In particular, the length of the sequences fed into the encoder, the hidden size of the GRU cells, the number of GRU cells, and the hidden size of the MLP RULP prediction head. In addition, there are hyperparameters specific to the training of the models, such as learning rate, and weight decay for the AdamW optimizer. The hyperparameters are tuned using hyperparameter tuning through the Ray Tune library (Liaw et al., 2018). Tuning is performed for each client for local training, see Section 2.3.2, and globally for the central training, see Section 2.3.3. For the federated learning models, hyperparameter tuning is not used, the best local hyperparameter settings are used.

2.3. Experimental Setup

The experimental setup takes advantage of the natural partitioning present in both datasets. This partitioning is used to artificially create clients in the federation, while also being more realistic than random partitioning of the datasets.

2.3.1. Data Partitioning

The MOSFET dataset consist of four experimental rounds (round 4 is omitted due to too highly accelerated testing) where each experimental round is used to represent a client in the federation. In this scenario, each client has a set of assets operating with some operating conditions. The operating conditions vary among clients but not within the cohort of assets for each client. This is a good proxy for assets within different companies; a company is likely to operate all their assets similarly, whereas another company might operate their assets differently. So in the experimental setup, various experimental rounds represent companies operating their assets. This setup ensures that the clients' data are highly non-IID, as driving current and ON/OFF time is different among clients, see Table 1. The non-IID nature of the clients' data can be seen in Figure 6 where the lifespans of the MOSFET devices are plotted by each experiment round (client).

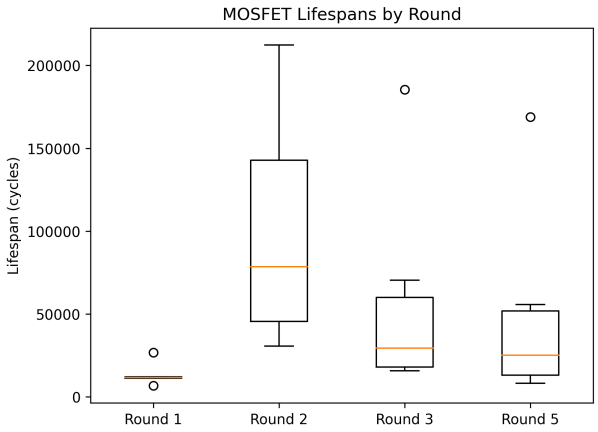


Figure 6. Lifespans of the SiC MOSFETs. It can be observed that there is great variability in lifespans, leading to clients with non-IID data, when each round is used as a client in the FL setup.

The C-MAPSS dataset consists of four "subsets" (FD001, FD002, FD003, and FD004). These subsets are again utilized to create four clients, each with aero engines being operated at different conditions. Since the subsets have different operating conditions, this again leads to clients with non-IID data. For example, the distribution of values for "sensor 12" which is ratio of fuel flow to Ps30, is shown in Figure 7. Here, it can be observed that data is differently distributed among the different subsets, leading to clients with non-IID data.

2.3.2. Local Training

To benchmark the federated learning models, central training and local training will be used. Central training is described in Section 2.3.3.

Each client in the federation (represented by different experimental rounds in the dataset) has a number of assets, that is referred to the client's local assets. As these assets are operated they produce data that will be called the local data.

To benchmark the federated learning approach, each client will also train a model only using their local data, representing a baseline for comparison. The model development is performed on a subset of the local data, called the development set, and is a randomly chosen subset of all assets. The development dataset will also be partitioned into two datasets; the training

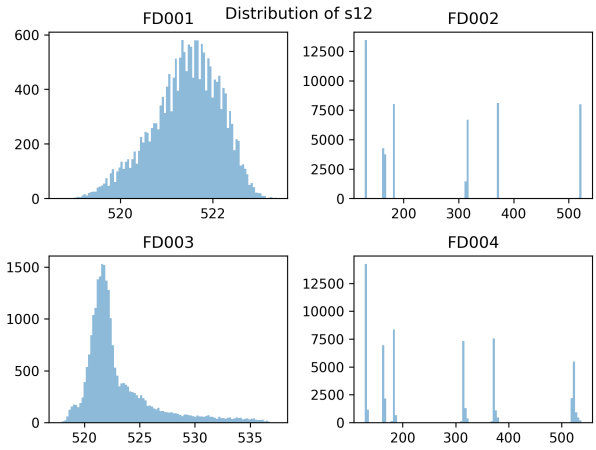


Figure 7. Distribution of "sensor 12" (ratio of fuel flow to Ps30) among the different subsets in the C-MAPSS dataset. A variation among subsets is observed, thus leading to clients with non-IID when subsets are used as clients in FL.

and validation datasets. The training dataset will be used for training the model, and the validation dataset will be used for monitoring the training. A third dataset called the evaluation/testing dataset will be used to evaluate the trained model and is used for comparing locally trained models and models trained using federated learning. In Figure 8 a schematic overview of the partitioning of each client's data is shown. For

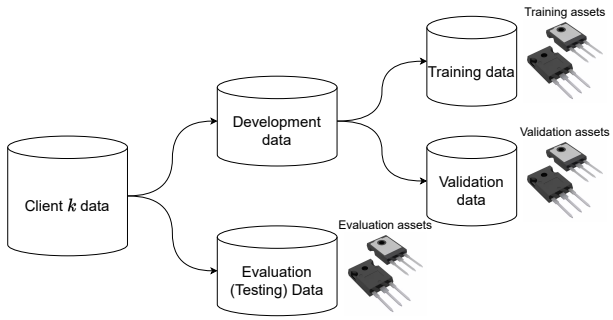


Figure 8. Each client's data are split into development data and evaluation/testing data. The development data are split into training data, data used for training the model, and validation data, used for monitoring the training. The test/evaluation data are held out during training and only used for evaluating the performance of the trained model.

the SiC MOSFET dataset, each device in every experiment round is randomly allocated to either, training, validation or testing datasuch that 60 % is used for training, 20 % is used for validation, and the last 20 % is used for testing.

For the C-MAPSS dataset, a test dataset is provided for each subset and is used for model evaluation. The remaining data is split such that 80 % is used for training and 20 % is used for validation.

Model Component	MLP-LSTM-MLP	ConvGRU
Batch Size	14	1
Sequence Length	-	200
LSTM/GRU Cells	128	32
No. Layers	-	1
Kernel Size	-	9
Learning Rate	0.0002	0.0002
Weight Decay	0.001	0.01

Table 5. Hyperparameter settings for the ConvGRU and MLP-LSTM-MLP models.

The models are updated using gradient descent on the loss function (LeCun, Bengio, & Hinton, 2015), where the loss function used are common loss functions used in regression problems, explained in Section 2.4. The weight update optimizer chosen is the so-called AdamW optimizer (Loshchilov & Hutter, 2019). It is a variation of the Adam optimizer (Kingma & Ba, 2015) with decoupled weight decay regularization. Both models are trained for 300 epochs, and the best models (in terms of validation loss) is selected. In Table 5, the hyperparameters for the ConvGRU and MLP-LSTM-MLP models are presented.

2.3.3. Central Training

Central machine learning, or (centralized machine learning) refers to a system of machine learning in which data is collected and processed in a single, central location (Abdulrahman et al., 2020). This approach is frequently used when working with large datasets that cannot be easily distributed across multiple machines or when real-time predictions are required.

In a central machine learning system, raw data are typically gathered from various sources and sent to a central server or cluster of servers for processing. The data is then cleaned, transformed, and prepared for training, after which a machine learning model is trained on the data using specialized algorithms and hardware. Once the model has been trained, it can be deployed in the same central location to make predictions on new data as it comes in.

Central machine learning has several advantages, including:

- Improved performance: By processing large datasets in a single location, central machine learning systems can take advantage of powerful hardware and software optimizations that would not be possible with distributed systems. This can lead to faster training times and more accurate predictions.
- Greater control: Centralizing the machine learning process allows for greater control over the data and models being used. This can help ensure compliance with regulatory requirements and minimize the risk of errors or inconsistencies in the training process.

- Scalability: Central machine learning systems are often more scalable than distributed systems, as they can easily accommodate larger datasets and more complex models as needed.

However, central machine learning also has some limitations, including:

- Data privacy concerns: Collecting and processing large amounts of data in a central location can raise privacy concerns, particularly if the data includes sensitive information.
- Dependence on infrastructure: Central machine learning systems are dependent on the reliability and availability of the underlying infrastructure. If the central server or cluster goes down, the entire system may be unavailable until it is restored.
- Latency issues: Central machine learning systems can introduce latency in the prediction process, particularly if the data must be transmitted over long distances or through congested networks.

In this study, central training as a limiting benchmark. Given the proper training and hyperparameters, the central model should perform at least equally well or better than the federated models.

In the scenario where clients are different companies training models on their own private data, central training is often infeasible. Companies often have sensitive data and often unable to share data between departments, let alone a third party. In this case, central training should not be considered a realistic alternative to federated learning, but it is an interesting benchmark from a model evaluation perspective, which is why it is included here.

Regarding the implementation details for central training, the same setup as for local training is used (in terms of hyperparameters). One could argue that hyperparameter search should be performed for central training and these are the hyperparameters that should be used in local training, so the proposed procedure needs some justification. The hyperparameters from local training are used for two main reasons. First, an extensive hyperparameter search for the different subsets yielded a small difference in hyperparameters, meaning the choice of hyperparameters found in local training can be used for each client. Secondly, in the real-world scenario, where companies are engaging in FL, central training is infeasible, for the reasons stated earlier. If central training is infeasible, so is central hyperparameter search (since data would need to be pooled to perform central hyperparameter search). In spirit of staying true to the goal of highlighting the difference between *local* and *FL* training, the training procedure should adhere as much as possible to the scenario targeted, meaning central hyperparameter search is omitted in favor of results obtained from local hyperparameter search.

In summary, central training is performed for each dataset separately, by pooling all the subsets for each dataset and training a model based on the pooled data, as one would do in traditional machine learning training.

2.3.4. Federated Training

The federated training is performed using the FedAvg and FedPer algorithms (see Algorithm 1 and Algorithm 2, respectively). As in the previous section, the local model updates are obtained using the local data training dataset. Then the updates are sent to the central server which uses some aggregation to create a new global model. In this case, aggregation is accomplished through the process of averaging the weights of the clients' models. When a new global model is created, it is distributed to each of the clients which resumes local training. An overview of the federated training is illustrated in Figure 9.

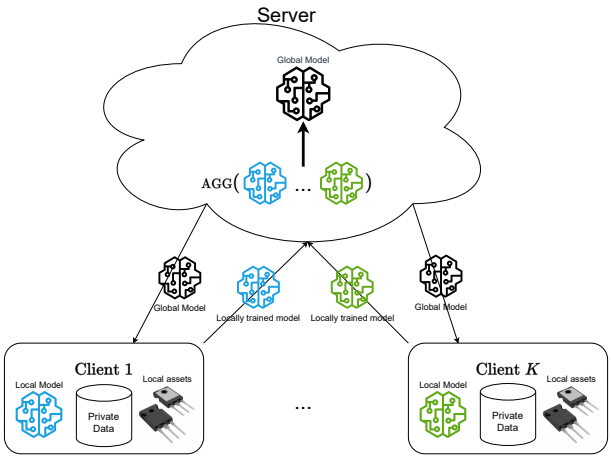


Figure 9. Conceptual illustration of the federated training setup. Each client $1, 2, \ldots, K$ has their local data based on their local assets and trains their local model on this data. This model is sent to the global server which aggregates the local models to form a new global model. The global model is then sent to each of the clients. The above procedure is repeated until convergence.

Federated training has a few hyperparameters specific to federated learning. Namely, one can choose how many epochs are trained locally before communicating the model to the server, referred to as "local epochs". In addition, the number of communications between the server and the clients can be set, called the "federated learning rounds". In this paper, each client trained for a single local epoch, and the number of federated learning rounds was set to 200, to give the models sufficient time to converge.

2.4. Model Evaluation

To evaluate the model performance, various metrics are used. For the MOSFET dataset two loss components are used, Mean Squared-Error (MSE) and ℓ^∞-loss. MSE loss is defined as

$$\ell_{\mathrm{MSE}}\left(\hat{y}, y\right) = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{y}_i - y_i\right)^2, \tag{12}$$

and ℓ^∞-loss is defined as

$$\ell^\infty(\hat{y}, y) = \max|\hat{y} - y|. \tag{13}$$

Hence, the total loss function is given by

$$\ell(\hat{y}, y) = \ell_{\mathrm{MSE}}(\hat{y}, y) + \ell^\infty(\hat{y}, y). \tag{14}$$

The MSE loss term is commonly used in regression problems. The ℓ^∞-loss term is used to penalize the largest output error of the model, leading to a more conservative model with fewer prediction outliers.

For the C-MAPSS dataset, two different metrics are used to monitor and evaluate model performance. To train the model, MSE is used. In the literature, it is common to report the performance of models on C-MAPSS using two metrics, root mean-squared error (RMSE) and a scoring function. RMSE is the square root of MSE:

$$\ell_{\mathrm{RMSE}}\left(\hat{y}, y\right) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\hat{y}_i - y_i\right)^2}. \tag{15}$$

The other common evaluation metric is a scoring function defined in the paper introducing the C-MAPSS dataset (Saxena et al., 2008). The scoring metric penalizes overestimation of RUL more than underestimation. The rationale is that overestimation of RUL is worse since the model is too optimistic about the remaining time left until failure, which means one is at risk of delaying maintenance action until it is too late. On the other hand, a conservative model that tends to underestimate RUL minimizes the risk of failure in favor of excess maintenance which is in many cases better. The scoring function is defined as follows:

$$\ell_{\mathrm{s}}\left(\hat{y}, y\right) = \begin{cases} \sum_{i=1}^{N}\exp\left(-\frac{d_i}{a_1}\right) - 1 & \text{for } d_i < 0 \\ \sum_{i=1}^{N}\exp\left(\frac{d_i}{a_2}\right) - 1 & \text{for } d_i \geq 0 \end{cases}. \tag{16}$$

where $d_i = \hat{y}_i - y_i,\ i = 1, 2, \ldots, N$, N is the number of assets under test, and $a_1 = 10$, $a_2 = 13$.

2.5. Implementation

The models and training framework were developed using the Python programming language (Van Rossum & Drake, 2009). In particular, the deep learning framework PyTorch (Paszke et al., 2019) and a high level wrapper for PyTorch called PyTorch Lightning (Falcon & team, 2019) was used to train the models. For the training the federated learning models, the Flower framework was used (Beutel et al., 2020).

3. RESULTS AND DISCUSSION

In this section, the results on each dataset will be presented in a corresponding subsection. The results include tables illustrating the relative overall performance on the datasets using the various metrics discussed in Section 2.4. In addition, plots of predicted RUL/RULP vs. true RUL/RULP will be presented to show the accuracy and spread in predictions by the various models.

3.1. MOSFET Dataset Results

The results on the MOSFET dataset are presented below. In Table 6 the resulting loss (as given by Equation (14)) is presented, showing that FL, in particular the FedPer strategy, is most performant. A notable exception is round 1 where local is performing the best. In Figure 10 and Figure 11, the predicted vs. actual RULP for round 2, device 6 and device 10 are plotted, respectively.

Round	Central	Local	FedAvg	FedPer
1	0.3547	**0.0070**	0.0823	0.0149
2	0.0482	0.0341	0.0771	**0.0126**
3	**0.0286**	0.0455	0.1519	0.0695
5	**0.0386**	0.0949	0.0664	0.0552
Mean	0.1174	0.0454	0.0944	**0.0381**

Table 6. Local, FedAvg, and FedPer test set loss on the MOSFET dataset. The best (minimal) loss is highlighted in bold. The last row is the mean of the loss.

Additional results are presented in the Appendix, see Figures 13 to 17.

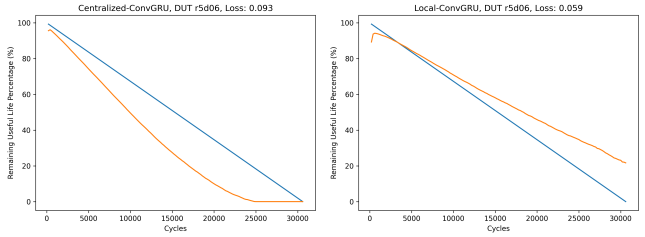
3.2. C-MAPSS Dataset Results

In Table 7 RMSE is calculated per subset on the evaluation dataset (defined in Equation (15)). Similarly, in Table 8, the score according to the scoring function defined in Equation (16) is shown.

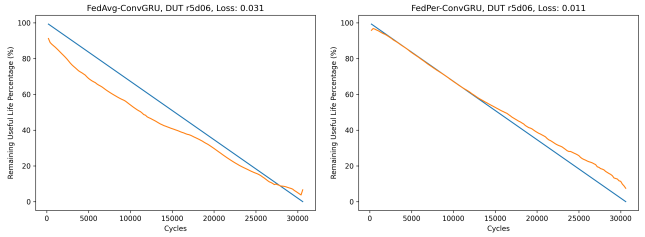
Subset	Central	Local	FedAvg	FedPer
FD001	**13.38**	14.36	13.60	13.44
FD002	23.49	24.20	**23.10**	23.13
FD003	13.14	12.99	**12.71**	14.24
FD004	24.72	33.15	**24.11**	24.16
Mean	18.68	21.18	**18.38**	18.74

Table 7. Local, FedAvg, and FedPer test set RMSE on the C-MAPSS dataset. The best (minimal) RMSE is highlighted in bold.

Additional results are presented in the Appendix, see Figures 18 to 20.

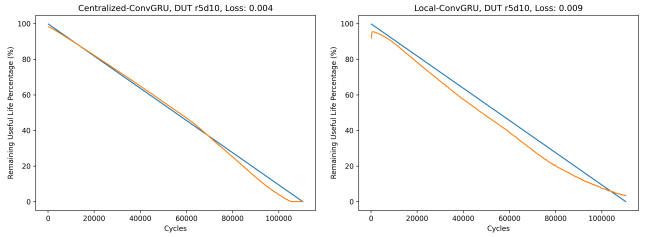


(a) Central model. (b) Local model.

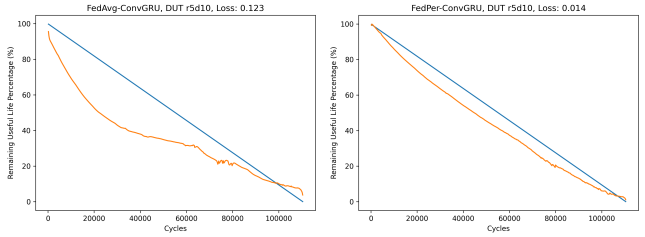


(c) FedAvg model. (d) FedPer model.

Figure 10. RUL prediction results on device 6 in round 2 from models trained locally, using FedAvg, and using FedPer. The number of cycles is shown on the horizontal axis, and the vertical axis shows the RULP. The true RULP is shown in blue and the predicted RULP is shown in orange.



(a) Central model. (b) Local model.



(c) FedAvg model. (d) FedPer model.

Figure 11. SiC MOSFET dataset RULP predictions for round 2, device 10.

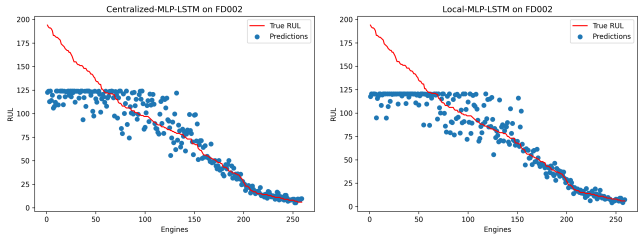
3.3. Discussion

When analyzing the results of the paper, the following guiding principles were used:

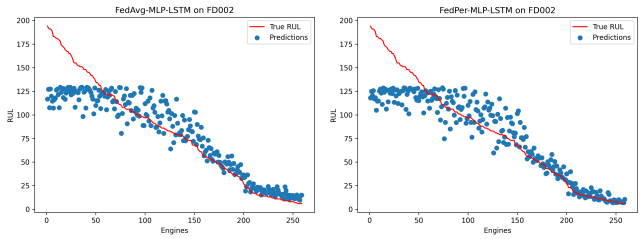
- central training is expected to outperform FL in general, as central training is a special case of FL with only one client. Therefore, in this work, central training serves as a "benchmark", and
- the difference between local training and FL is the most important result, as this represent the realistic dilemma companies are faced with: is there a performance gain of using FL over just training the model locally on private data?

Subset	Central	Local	FedAvg	FedPer
FD001	343.90	**309.78**	364.65	368.68
FD002	4300.42	5036.62	4147.94	**3924.87**
FD003	350.36	434.39	**304.16**	566.55
FD004	5073.86	11803.64	**3370.73**	3375.74
Mean	2517.13	4396.11	**2046.87**	2058.96

Table 8. Local, FedAvg, and FedPer test set score on the C-MAPSS dataset. The best (minimal) score is highlighted in bold.



(a) Central model. (b) Local model.



(c) FedAvg model. (d) FedPer model.

Figure 12. RUL prediction results on subset FD002 from models trained locally, using FedAvg, and using FedPer. The engine ID are shown on the horizontal axis, sorted in descending manner by highest actual RUL. The true RUL is shown as a red line, whereas the predicted RUL is shown as blue points.

With these principles in mind, the results show that various federated learning approaches are, in general, performing better than only training on local data. In particular, it is observed that the FedAvg strategy is performing well, especially on the C-MAPSS dataset. This is probably because the subsets in the C-MAPSS dataset, while still having different operating conditions for each subset, are quite similar (especially regarding lifespans), leading to the FedAvg strategy working well. One should also note that the FedPer strategy works almost as well as the FedAvg algorithm on the C-MAPSS dataset, with results close to (and even exceeding) central training. The FedPer strategy has a consistently high performance on both datasets, especially the MOSFET dataset. One plausible reason is that when using the FedPer strategy, each client is allowed to use personalized layers in the model, which gives the local models more flexibility to handle non-IID data. As discussed earlier,

the MOSFET dataset is quite diverse among the devices in the different rounds, and the FedPer strategy's ability to cope with heterogeneous data could explain the observed results. As companies are likely to have assets that experience a variety of degradation processes and therefore collect highly heterogeneous data, it is important for the federated strategy to handle this in a robust manner.

Central training is the worst performer on the MOSFET dataset, which is an unexpected result. However, the low performance of central training can be attributed to a large incorrect prediction on round 1. Due to the low number of DUTs in round 1, only one device is assigned to the testing dataset (see Table 3). Weighting the prediction results with the number of devices in the testing dataset for each round one can observe that the performance of the central training is not as bad as it seems at first glance and the large contribution of round 1 is more clear (see Table 6).

One can also note that local training is working well and for some clients this model is the top performer. One explanation for this is that the models used were small, in the sense of total number of trainable parameters. Smaller models are less prone to overfitting and can therefore can perform relatively well on smaller datasets.

Regarding the SiC MOSFET dataset results, one can observe in Table 6 that FedPer is outperforming local training except for round 1 and 3 where local training is the best performer, and FedPer is a close second. One possible explanation is that round 1 contains the fewest number of devices, thereby as a client in the federation, is contributing less in the weight-update of the federated learning algorithm, see Algorithm 1 and Algorithm 2.

In addition, one can observe that the models perform well on the device with the longest life span, see Figure 11, but they are less accurate when the life span is shorter. This can occur if the training dataset contains more longer-lived devices, as compared to the test dataset. Indeed, as can be seen in Figure 6, there is a large spread in life-spans and most short-lived devices originate from round 1, which contains the fewest number of devices.

In particular, one interesting observation regarding round 2 (the round with the lowest accelerated aging) is that the local model is preforming well on device 10 which has a long life-span, see Figure 11. The federated learning algorithms have comparable, albeit slightly worse, performance on this device. However, the local model performs poorly on the shorter-lived device 6, see Figure 10, whereas the federated models perform better. This is an indication that the federated models have learned "short-lived" behavior from other clients in the federation, since such examples are more numerous and common among other the other rounds/clients.

For the results on the C-MAPSS dataset, it can be observed in Table 7 and Table 8 that federated learning is performing better, especially FedAvg. In general, it is observed that RUL predictions are quite poor when the remaining life is large, see e.g., Figure 12. This is because, RUL is clipped at 130 cycles in the testing and validation datasets. At high RUL little or no degradation is present in the system, leading to large uncertainty in RUL predictions, and RUL clipping is therefore used in training to stabilize the training process. Naturally, as the model never receives samples with RUL exceeding 130 cycles, it will never predict RUL higher than the clipping value. Of course, the RUL of the test devices are not clipped and RUL values exceeding the threshold will be underestimated close to the clipping threshold. This leads to poor results at high RUL, especially RMSE. As time progresses, the degradation signal in the data becomes more pronounced and consequently, the RUL predictions are more accurate. This is what is observed, the predictions at low RUL are far more accurate than at high RUL.

4. CONCLUSIONS

In this paper, a method for collaborative training of Remaining Useful Life (RUL) prediction models using federated learning was explored, developed, and evaluated. Two federated learning strategies were compared, Federated Averaging (FedAvg) and Federated Learning with Personalized Layers (FedPer), to each other and the baseline case of training models on private locally held data. The results show that the models trained collaboratively using federated learning have similar or better performance as compared to the baseline case.

Future research can further explore and optimize the application of federated learning in this domain to unlock its full potential. Specifically, bridging the gap between hybrid models (machine learning models combined with physics of failure models) with federated learning is a worthwhile future endeavor. Incorporating physical knowledge in machine learning models is a crucial step for regularizing and enhancing data-driven PHM applications. Furthermore, investigating additional privacy enhancing features such as differential privacy is interesting and may help bring federated learning for RUL prediction to safety-critical and privacy-sensitive applications.

Overall, the results suggest that federated learning is a promising avenue for companies and organizations seeking to enhance the overall performance of their prognostics models, collaboratively. The adoption of federated learning techniques in training RUL prediction models holds the potential to improve the accuracy and reliability of predictive maintenance systems, ultimately leading to more effective and efficient maintenance practices in various industrial sectors.

ACKNOWLEDGMENT

This research is supported by the Chips Joint Undertaking and its members, including the top-up funding by the national Au-

thorities of Germany, Belgium, Spain, Sweden, Netherlands, Austria, Italy, Greece, Latvia, Finland, Hungary, Romania and Switzerland, under grant agreement number 101096387. Cofunded by European Union. In particular, the authors thank the Swedish funding authority Vinnova for support. This research work has been funded by the Knowledge Foundation within the framework of the INDTECH (Grant Number 20200132) Research School project, participating companies and Mälardalen University. The authors also give their thanks to the RISE internal project DIGIPROD for supporting this work. Research was supported by XPRES (Centre of Excellence in Production Research) — a strategic research area in Sweden. The authors thank Andreas Lövberg for his input in the initial stage of this research.

REFERENCES

Abdelli, K., Cho, J. Y., & Pachnicke, S. (2021). Secure collaborative learning for predictive maintenance in optical networks [Conference paper]. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *13115 LNCS*, 114 – 130. (Cited by: 0) doi: 10.1007/978-3-030-91625-1_7

Abdulrahman, S., Tout, H., Ould-Slimane, H., Mourad, A., Talhi, C., & Guizani, M. (2020, 10). A survey on federated learning: The journey from centralized to distributed on-site learning and beyond. *IEEE Internet of Things Journal*, *PP*. doi: 10.1109/JIOT.2020.3030072

Agarap, A. F. (2018). *Deep learning using rectified linear units (relu).* Retrieved from http://arxiv.org/abs/1803.08375 (cite arxiv:1803.08375Comment: 7 pages, 11 figures, 9 tables)

Al-Dulaimi, A., Zabihi, S., Asif, A., & Mohammadi, A. (2019). A multimodal and hybrid deep neural network model for remaining useful life estimation [Article]. *Computers in Industry*, *108*, 186 – 196. (Cited by: 167) doi: 10.1016/j.compind.2019.02.004

Arivazhagan, M. G., Aggarwal, V., Singh, A. K., & Choudhary, S. (2019). Federated learning with personalization layers. *CoRR*, *abs/1912.00818*. Retrieved from http://arxiv.org/abs/1912.00818

Arunan, A., Qin, Y., Li, X., & Yuen, C. (2023). A federated learning-based industrial health prognostics for heterogeneous edge devices using matched feature extraction [Article]. *IEEE Transactions on Automation Science and Engineering*, 1–15. (Cited by: 2; All Open Access, Green Open Access) doi: 10.1109/TASE.2023.3274648

Ballas, N., Yao, L., Pal, C., & Courville, A. C. (2016). Delving deeper into convolutional networks for learning video representations. In Y. Bengio & Y. LeCun (Eds.), *4th international conference on learning representations, ICLR 2016, san juan, puerto rico, may 2-4, 2016, conference track proceedings*. Retrieved from http://arxiv.org/abs/1511.06432

Bemani, A., & Björsell, N. (2022). Aggregation strategy on federated machine learning algorithm for collaborative predictive maintenance [Article]. *Sensors*, *22*(16). (Cited by: 11; All Open Access, Gold Open Access, Green Open Access) doi: 10.3390/s22166252

Ben Ali, J., Chebel-Morello, B., Saidi, L., Malinowski, S., & Fnaiech, F. (2015). Accurate bearing remaining useful life prediction based on weibull distribution and artificial neural network. *Mechanical Systems and Signal Processing*, *56-57*, 150-172. Retrieved from https://www.sciencedirect.com/science/article/pii/S0888327014004087 doi: https://doi.org/10.1016/j.ymssp.2014.10.014

Beutel, D. J., Topal, T., Mathur, A., Qiu, X., Parcollet, T., & Lane, N. D. (2020). Flower: A friendly federated learning research framework. *CoRR*, *abs/2007.14390*. Retrieved from https://arxiv.org/abs/2007.14390

Celaya, J. R., Saxena, A., Saha, S., & Goebel, K. F. (2011). Prognostics of power mosfets under thermal stress accelerated aging using data-driven and model-based methodologies. In (Vol. 3, p. 1995). doi: 10.36001/phmconf.2011.v3i1.1995

Chaoub, A., Voisin, A., Cerisara, C., & Iung, B. (2021). Learning representations with end-to-end models for improved remaining useful life prognostics. *CoRR*, *abs/2104.05049*. Retrieved from https://arxiv.org/abs/2104.05049

Che, C., Wang, H., Fu, Q., & Ni, X. (2019). Combining multiple deep learning algorithms for prognostic and health management of aircraft [Review]. *Aerospace Science and Technology*, *94*. (Cited by: 87) doi: 10.1016/j.ast.2019.105423

Chen, Q., Nicholson, G., Ye, J., Zhao, Y., & Roberts, C. (2020). Estimating residual life distributions of complex operational systems using a remaining maintenance free operating period (rmfop)-based methodology. *Sensors*, *20*(19). Retrieved from https://www.mdpi.com/1424-8220/20/19/5504 doi: 10.3390/s20195504

Chen, X., Wang, H., Lu, S., Xu, J., & Yan, R. (2023). Remaining useful life prediction of turbofan engine using global health degradation representation in federated learning. *Reliability Engineering & System Safety*, *239*, 109511. Retrieved from https://www.sciencedirect.com/science/article/pii/S0951832023004258 doi: https://doi.org/10.1016/j.ress.2023.109511

Clevert, D.-A., Unterthiner, T., & Hochreiter, S. (2016). Fast and accurate deep network learning by exponential linear units (elus). In Y. Bengio

& Y. LeCun (Eds.), *Iclr (poster)*. Retrieved from http://dblp.uni-trier.de/db/conf/iclr/iclr2016.html#ClevertUH15

Demus, J., Sysoeva, V., Cheng, Q., Boubin, M., Siraj, A., & Scott, M. (2019). Mosfet junction temperature measurements using conducted electromagnetic emissions and support vector machines [Conference paper]. In (p. 2973 – 2978). Institute of Electrical and Electronics Engineers Inc. (Cited by: 0) doi: 10.1109/ECCE.2019.8912938

Dhada, M. H., Parlikad, A. K., & Palau, A. S. (2020). Federated learning for collaborative prognosis.. Retrieved from https://api.semanticscholar.org/CorpusID:235049866

Doulamis, A. D., Hou, G., Xu, S., Zhou, N., Yang, L., & Fu, Q. (2020). Remaining useful life estimation using deep convolutional generative adversarial networks based on an autoencoder scheme. *Computational Intelligence and Neuroscience*, *2020*, 9601389. Retrieved from https://doi.org/10.1155/2020/9601389 doi: 10.1155/2020/9601389

Du, N. H., Long, N. H., Ha, K. N., Hoang, N. V., Huong, T. T., & Tran, K. P. (2023). Trans-lighter: A light-weight federated learning-based architecture for remaining useful lifetime prediction. *Computers in Industry*, *148*, 103888. Retrieved from https://www.sciencedirect.com/science/article/pii/S0166361523000386 doi: https://doi.org/10.1016/j.compind.2023.103888

Falcon, W., & team, T. P. L. (2019, 3). *Pytorch lightning*. Retrieved from https://www.pytorchlightning.ai doi: 10.5281/zenodo.3828935

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. (http://www.deeplearningbook.org)

Guo, L., Yu, Y., Qian, M., Zhang, R., Gao, H., & Cheng, Z. (2023). Fedrul: A new federated learning method for edge-cloud collaboration based remaining useful life prediction of machines. *IEEE/ASME Transactions on Mechatronics*, *28*(1), 350-359. doi: 10.1109/TMECH.2022.3195524

Haris, M., Hasan, M. N., Jahanzeb Hussain Pirzada, S., & Qin, S. (2020). Bayesian optimized long-short term memory recurrent neural network for prognostics of thermally aged power mosfets [Conference paper]. In S. M., M. M.A., & N. S. (Eds.), . Institute of Electrical and Electronics Engineers Inc. (Cited by: 2) doi: 10.1109/RAEECS50817.2020.9265738

Hestness, J., Narang, S., Ardalani, N., Diamos, G. F., Jun, H., Kianinejad, H., ... Zhou, Y. (2017). Deep learning scaling is predictable, empirically. *CoRR*, *abs/1712.00409*. Retrieved from http://arxiv.org/abs/1712.00409

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Huang, C.-G., Zhu, J., Han, Y., & Peng, W. (2022). A novel bayesian deep dual network with unsupervised domain adaptation for transfer fault prognosis across different machines. *IEEE Sensors Journal*, *22*(8), 7855-7867. doi: 10.1109/JSEN.2021.3133622

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., ... Zhao, S. (2021).

Kamei, S., & Taghipour, S. (2023). A comparison study of centralized and decentralized federated learning approaches utilizing the transformer architecture for estimating remaining useful life [Article]. *Reliability Engineering and System Safety*, *233*. (Cited by: 6) doi: 10.1016/j.ress.2023.109130

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In Y. Bengio & Y. LeCun (Eds.), *3rd international conference on learning representations, ICLR 2015, san diego, ca, usa, may 7-9, 2015, conference track proceedings*. Retrieved from http://arxiv.org/abs/1412.6980

Lallart, M., Wu, B., Li, W., & Qiu, M.-q. (2017). Remaining useful life prediction of bearing with vibration signals based on a novel indicator. *Shock and Vibration*, *2017*, 8927937. Retrieved from https://doi.org/10.1155/2017/8927937 doi: 10.1155/2017/8927937

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. Retrieved from https://doi.org/10.1038/nature14539 doi: 10.1038/nature14539

Li, Q., Wen, Z., Wu, Z., Hu, S., Wang, N., Li, Y., ... He, B. (2023, apr). A survey on federated learning systems: Vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge; Data Engineering*, *35*(04), 3347-3366. doi: 10.1109/TKDE.2021.3124599

Li, T. (2024). Particle filter-based fatigue damage prognosis using prognostic-aided model updating. *Mechanical Systems and Signal Processing*, *211*, 111244. Retrieved from https://www.sciencedirect.com/science/article/pii/S0888327024001420 doi: https://doi.org/10.1016/j.ymssp.2024.111244

Liaw, R., Liang, E., Nishihara, R., Moritz, P., Gonzalez, J. E., & Stoica, I. (2018). Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*.

Loshchilov, I., & Hutter, F. (2019). Decoupled weight decay regularization. In *International conference on learning representations*. Retrieved from https://openreview.net/forum?id=Bkg6RiCqY7

Mahamad, A. K., Saon, S., & Hiyama, T. (2010). Predicting remaining useful life of rotating machinery based artificial neural network. *Computers & Mathematics with Applications*, *60*(4),

1078-1087. Retrieved from https://www.sciencedirect.com/science/article/pii/S0898122110002555 (PCO' 2010) doi: https://doi.org/10.1016/j.camwa.2010.03.065

McMahan, B., Moore, E., Ramage, D., Hampson, S., & Arcas, B. A. y. (2017, 20–22 Apr). Communication-efficient learning of deep networks from decentralized data. In A. Singh & J. Zhu (Eds.), *Proceedings of the 20th international conference on artificial intelligence and statistics* (Vol. 54, pp. 1273–1282). PMLR. Retrieved from https://proceedings.mlr.press/v54/mcmahan17a.html

Meriem, H., Nora, H., & Samir, O. (2023). Predictive maintenance for smart industrial systems: A roadmap. *Procedia Computer Science*, *220*, 645-650. Retrieved from https://www.sciencedirect.com/science/article/pii/S1877050923006178 (The 14th International Conference on Ambient Systems, Networks and Technologies Networks (ANT) and The 6th International Conference on Emerging Data and Industry 4.0 (EDI40)) doi: https://doi.org/10.1016/j.procs.2023.03.082

Moshawrab, M., Adda, M., Bouzouane, A., Ibrahim, H., & Raad, A. (2023). Reviewing federated learning aggregation algorithms; strategies, contributions, limitations and future perspectives. *Electronics*, *12*(10). Retrieved from https://www.mdpi.com/2079-9292/12/10/2287 doi: 10.3390/electronics12102287

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc. Retrieved from http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Pecht, M., & Kang, M. (2018). *Prognostics and health management of electronics : fundamentals, machine learning, and internet of things* (Second edition. ed.). Hoboken, New Jersey: John Wiley & Sons.

Ren, H., Du, X., Yu, Y., Wang, J., Zhou, J., & Peng, Y. (2022). Power mosfet lifetime prediction method based on optimized long short-term memory neural network [Conference paper]. Institute of Electrical and Electronics Engineers Inc. (Cited by: 0) doi: 10.1109/ECCE50734.2022.9947640

Sahu, A. K., Li, T., Sanjabi, M., Zaheer, M., Talwalkar, A., & Smith, V. (2018). On the convergence of federated optimization in heterogeneous networks. *CoRR*, *abs/1812.06127*. Retrieved from http://arxiv.org/abs/1812.06127

Saxena, A., Goebel, K., Simon, D., & Eklund, N. (2008). Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 international conference on prognostics and health management* (p. 1-9). doi: 10.1109/PHM.2008.4711414

Singh, J., Darpe, A. K., & Singh, S. P. (2020, may). Bearing remaining useful life estimation using an adaptive data-driven model based on health state change point identification and k-means clustering. *Measurement Science and Technology*, *31*(8), 085601. Retrieved from https://dx.doi.org/10.1088/1361-6501/ab6671 doi: 10.1088/1361-6501/ab6671

Söderkvist Vermelin, W., Lövberg, A., Misiorny, M., P. Eng, M., & Brinkfeldt, K. (2023). Data-driven remaining useful life estimation of discrete power electronic devices. In *Proceedings of the 33rd european safety and reliability conference (esrel 2023)* (Vol. 33). European Safety and Reliability Conference. Retrieved from https://urn.kb.se/resolve?urn=urn:nbn:se:ri:diva-67109 doi: 10.3850/978-981-18-8071-1-4procd

Tian, Z. (2009). An artificial neural network approach for remaining useful life prediction of equipments subject to condition monitoring. In *2009 8th international conference on reliability, maintainability and safety* (p. 143-148). doi: 10.1109/ICRMS.2009.5270220

Tian, Z., Wong, L., & Safaei, N. (2010). A neural network approach for remaining useful life prediction utilizing both failure and suspension histories. *Mechanical Systems and Signal Processing*, *24*(5), 1542-1555. Retrieved from https://www.sciencedirect.com/science/article/pii/S088832700900377X (Special Issue: Operational Modal Analysis) doi: https://doi.org/10.1016/j.ymssp.2009.11.005

Van Rossum, G., & Drake, F. L. (2009). *Python 3 reference manual*. Scotts Valley, CA: CreateSpace.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In I. Guyon et al. (Eds.), *Advances in neural information processing systems* (Vol. 30). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Vibhorpandhare, Jia, X., & Lee, J. (2021). Collaborative prognostics for machine fleets using a novel federated baseline learner.. Retrieved from https://api.semanticscholar.org/CorpusID:244645140

Wang, J., Wen, G., Yang, S., & Liu, Y. (2019). Remaining useful life estimation in prognostics using deep bidi-

rectional lstm neural network [Conference paper]. In D. P., L. C., Y. S., D. P., & S. R.-V. (Eds.), (p. 1037 – 1042). Institute of Electrical and Electronics Engineers Inc. (Cited by: 121) doi: 10.1109/PHM-Chongqing .2018.00184

Wu, S.-j., Gebraeel, N., Lawley, M. A., & Yih, Y. (2007). A neural network integrated decision support system for condition-based optimal predictive maintenance policy. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, *37*(2), 226-236. doi: 10.1109/TSMCA.2006.886368

Xia, M., Li, T., Shu, T., Wan, J., de Silva, C. W., & Wang, Z. (2019). A two-stage approach for the remaining useful life prediction of bearings using deep neural networks. *IEEE Transactions on Industrial Informatics*, *15*(6), 3703-3711. doi: 10.1109/TII.2018.2868687

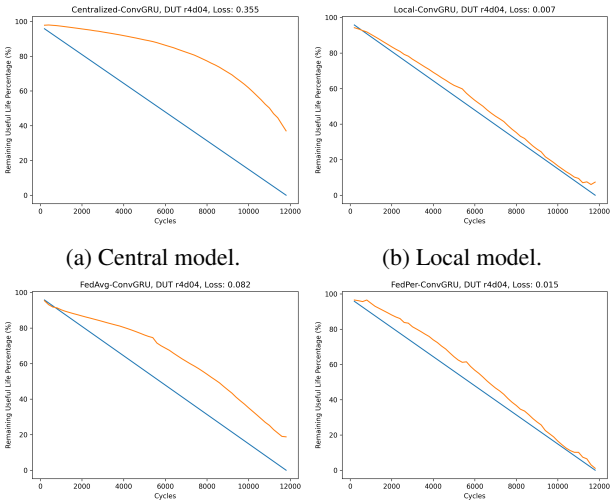
Xu, J., Duan, S., Chen, W., Wang, D., & Fan, Y. (2022). Sacgnet: A remaining useful life prediction of bearing with self-attention augmented convolution gru network. *Lubricants*, *10*(2). Retrieved from `https://www.mdpi.com/2075-4442/10/2/21` doi: 10 .3390/lubricants10020021

Zhang, J., Wang, P., Yan, R., & Gao, R. X. (2018). Long short-term memory for machine remaining life prediction [Article]. *Journal of Manufacturing Systems*, *48*, 78 – 86. (Cited by: 302) doi: 10.1016/j.jmsy.2018.05.011

Zheng, S., Ristovski, K., Farahat, A., & Gupta, C. (2017). Long short-term memory network for remaining useful life estimation. In *2017 ieee international conference on prognostics and health management (icphm)* (p. 88-95). doi: 10.1109/ICPHM.2017.7998311

Zhu, J., Chen, N., & Shen, C. (2020). A new data-driven transferable remaining useful life prediction approach for bearing under different working conditions. *Mechanical Systems and Signal Processing*, *139*, 106602. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0888327019308234` doi: https://doi.org/10.1016/j.ymssp.2019.106602

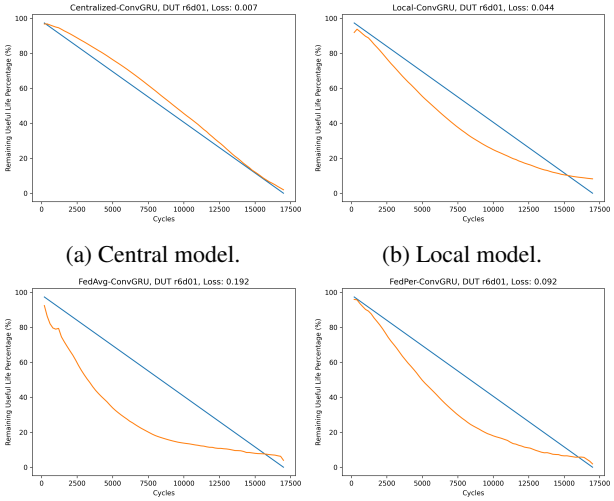
APPENDIX



(a) Central model. (b) Local model.

(c) FedAvg model. (d) FedPer model.

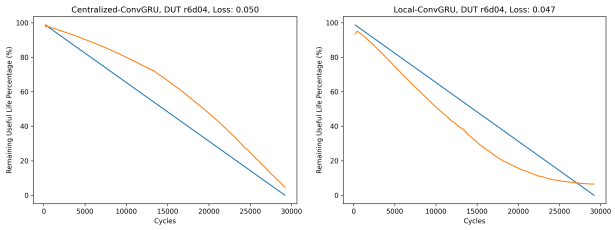
Figure 13. SiC MOSFET dataset RULP predictions for round 1, device 4.



(a) Central model. (b) Local model.

(c) FedAvg model. (d) FedPer model.

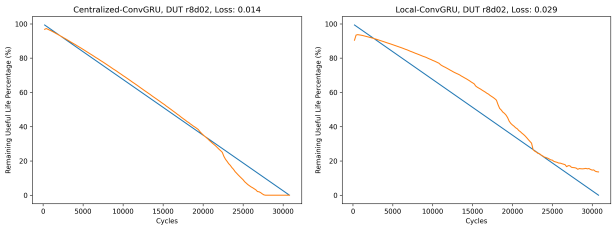
Figure 14. SiC MOSFET dataset RULP predictions for round 3, device 1.



(a) Central model. (b) Local model.

(c) FedAvg model. (d) FedPer model.

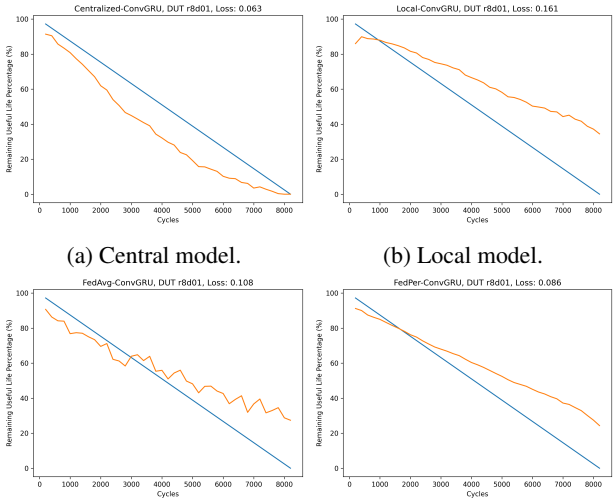
Figure 15. SiC MOSFET dataset RULP predictions for round 3, device 4.



(a) Central model. (b) Local model.

(c) FedAvg model. (d) FedPer model.

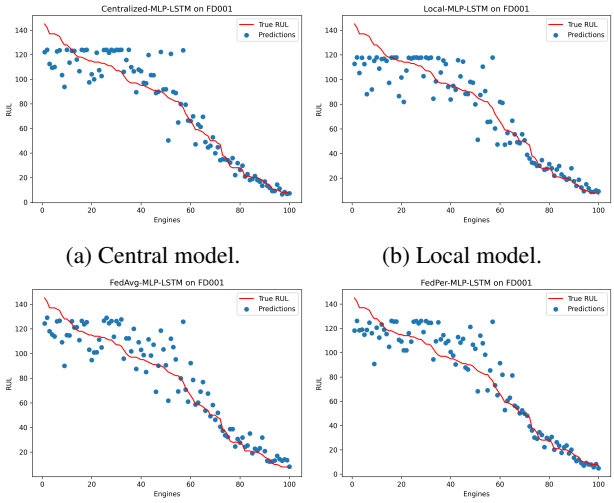
Figure 17. SiC MOSFET dataset RULP predictions for round 5, device 2.



(a) Central model. (b) Local model.

(c) FedAvg model. (d) FedPer model.

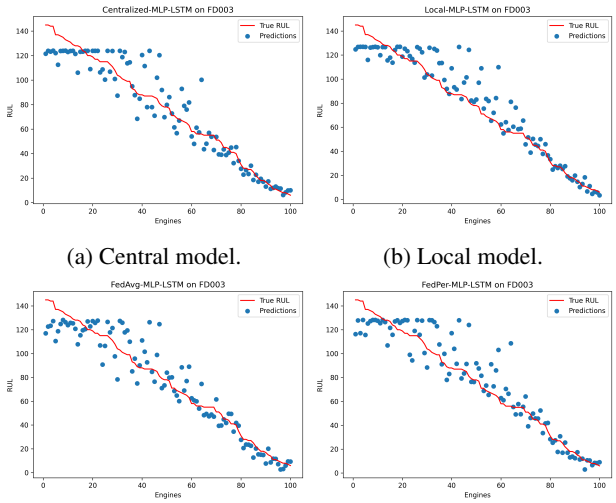
Figure 16. SiC MOSFET dataset RULP predictions for round 5, device 1.



(a) Central model. (b) Local model.

(c) FedAvg model. (d) FedPer model.

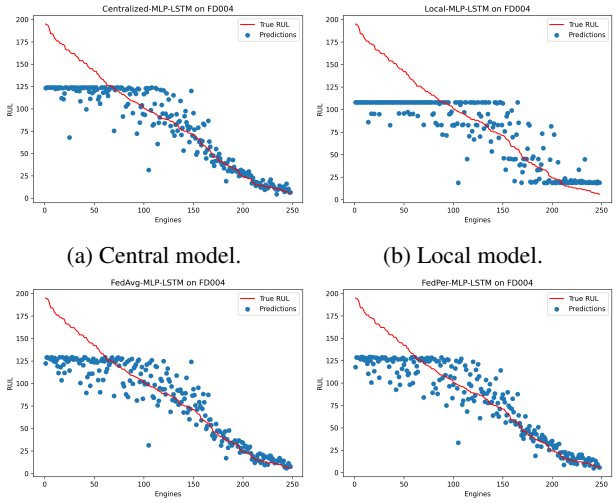
Figure 18. C-MAPSS dataset RUL predictions for FD001 subset.



(a) Central model. (b) Local model.

(c) FedAvg model. (d) FedPer model.

Figure 19. C-MAPSS dataset RUL predictions for FD003 subset.



(a) Central model. (b) Local model.

(c) FedAvg model. (d) FedPer model.

Figure 20. C-MAPSS dataset RUL predictions for FD004 subset.