

## Label synchronization for Hybrid Federated Learning in manufacturing and predictive maintenance

Llasag Rosero, Raúl; Silva, Catarina; Ribeiro, Bernardete; Santos, Bruno F.

**DOI**

[10.1007/s10845-023-02298-8](https://doi.org/10.1007/s10845-023-02298-8)

**Publication date**

2024

**Document Version**

Final published version

**Published in**

Journal of Intelligent Manufacturing

**Citation (APA)**

Llasag Rosero, R., Silva, C., Ribeiro, B., & Santos, B. F. (2024). Label synchronization for Hybrid Federated Learning in manufacturing and predictive maintenance. *Journal of Intelligent Manufacturing*, 35(8), 4015-4034. <https://doi.org/10.1007/s10845-023-02298-8>

**Important note**

To cite this publication, please use the final published version (if applicable).  
Please check the document version above.

**Copyright**

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

**Takedown policy**

Please contact us and provide details if you believe this document breaches copyrights.  
We will remove access to the work immediately and investigate your claim.



# Label synchronization for Hybrid Federated Learning in manufacturing and predictive maintenance

Raúl Llasag Rosero<sup>1</sup> · Catarina Silva<sup>1</sup> · Bernardete Ribeiro<sup>1</sup> · Bruno F. Santos<sup>2</sup>

Received: 8 March 2023 / Accepted: 2 December 2023  
© The Author(s) 2024

## Abstract

Artificial Intelligence (AI) is transforming the future of industries by introducing new paradigms. To address data privacy and other challenges of decentralization, research has focused on Federated Learning (FL), which combines distributed Machine Learning (ML) models from multiple parties without exchanging confidential information. However, conventional FL methods struggle to handle situations where data samples have diverse features and sizes. We propose a Hybrid Federated Learning solution with label synchronization to overcome this challenge. Our FedLabSync algorithm trains a feed-forward Artificial Neural Network while alerts that it can aggregate knowledge of other ML architectures compatible with the Stochastic Gradient Descent algorithm by conducting a penalized collaborative optimization. We conducted two industrial case studies: product inspection in Bosch factories and aircraft component Remaining Useful Life predictions. Our experiments on decentralized data scenarios demonstrate that FedLabSync can produce a global AI model that achieves results on par with those of centralized learning methods.

**Keywords** Federated Learning · Machine Learning · Hybrid Federated Learning · Artificial Neural Network · Stochastic Gradient Descent

## List of symbols

$D$	Dataset distributed among parties	$Y$	Labels of $D$
$M$	Number of features of $D$	$Y^j$	Labels of the $j$ th party
$N$	Number of samples of $D$	$\hat{Y}^j$	Predictions of the $j$ th party
$J$	Number of parties	$X_i$	Input data of the $i$ th sample
$j$	The $j$ th party	$N^j$	Dataset size of the $j$ th party
$X\{p_1, \dots, p_M\}$	$D$ 's input data of $M$ features	$M^j$	Number of the dataset features of the $j$ th party
$p$	The $p$ th feature	$A^{N \times J}$	Prediction matrix of $N$ rows and $J$ columns
		$M_{lbs}$	Labels Matrix
		$M_{sync}$	Label Synchronization Matrix
		$B$	Mini-batch size
		$K$	Number of model aggregations
		$\kappa$	The $\kappa$ th global model computation
		$w$	Weights of global AI model
		$w^*$	Optimal weights of global AI model
		$F_\kappa(w)$	HFL cost function of global AI model at $\kappa$ th global model computation
		$\xi(w)$	VFL cost function of global AI model
		$T$	Number of training iterations in the local training step
		$w_t^j$	Weights of the $j$ th party at $t$ th time

✉ Raúl Llasag Rosero  
rosero@dei.uc.pt

Catarina Silva  
catarina@dei.uc.pt

Bernardete Ribeiro  
bribeiro@dei.uc.pt

Bruno F. Santos  
B.F.Santos@tudelft.nl

<sup>1</sup> Department of Informatics Engineering, CISUC - Centre for Informatics and Systems of the University of Coimbra, Polo II - Pinhal de Marrocos, 3030-290 Coimbra, Portugal

<sup>2</sup> Faculty of Aerospace Engineering, Delft University of Technology, Kluyverweg 1, Delft, South Holland 2629 HS, The Netherlands

$F_t^j(w^j)$	HFL cost function of the $j$ th party at $t$ th time
$\sigma_i(w)$	Loss function of VFL collaborative prediction at the $i$ th sample
$\phi(\cdot)$	Activation function (e.g. relu)
$f_i^j(w^j)$	Loss function of the $j$ th party at the $i$ th sample
$\eta$	Step-size of Stochastic Gradient Descent algorithm

## Introduction

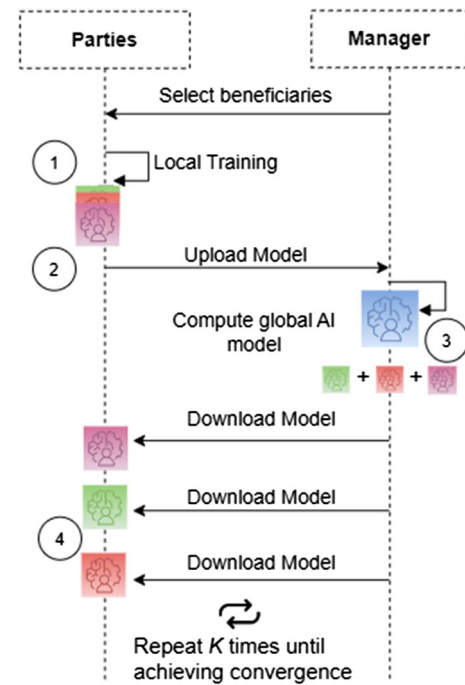
The future industry heavily relies on Artificial Intelligence (AI) (Nguyen et al., 2021; Pharm et al., 2021). AI is a broad field of computer science that focuses on creating computer systems capable of performing tasks that require human-like thinking (Kofi et al., 2022). In various industrial applications, such as predictive maintenance, fault diagnostics, failure prediction, and manufacturing process analysis, AI and the Internet of Things (IoT) are integrated to analyze large amounts of data and improve operations (Ahmad et al., 2021; Peng et al., 2022).

Federated Learning (FL) has become a popular solution to address privacy concerns in distributed Machine Learning (ML) (Kairouz, 2019; Kallista et al., 2022; Li et al., 2019; Zhang et al., 2020). In distributed scenarios, FL enables the training of AI models locally, ensuring that sensitive data remains with its owners and is not transmitted in raw form (Abdulrahman et al., 2021). By doing so, FL provides a decentralized and privacy-preserving approach to machine learning.

Figure 1 presents the main components of federated environments, including:

- **Parties:** devices or organizations that own data and AI models and are the *beneficiaries* of FL applications.
- **Manager:** a computational server communicating with parties and usually storing the global AI federated model.
- **Communication-computation framework:** the FL algorithm that trains the global AI model (Kallista et al., 2022).

Once the federated participants (parties) are selected, the FL process follows a series of key loop steps, executed  $K$  times, until the convergence of the global model is achieved. As depicted in Fig. 1, the standard steps include (1) local training, (2) uploading the model, (3) computing the global model, and (4) distributing it to the parties. Researchers are constantly working to overcome challenges in federated networks, such as improving communication efficiency (Li et al., 2019), addressing system heterogeneity (Abdulrahman



**Fig. 1** Federated learning overview: components (parties, manager, and communication-computation) and the key steps of the approach

et al., 2021), and ensuring data privacy (Abdulrahman et al., 2021; Li et al., 2019). They are also exploring ways to distribute data in federated environments effectively (Li et al., 2019; Liu et al., 2018).

Federated applications are built based on three data partitioning settings: Horizontal (HFL), Vertical (VFL), and Hybrid Federated Learning (Hybrid-FL) (Abdulrahman et al., 2021; Li et al., 2019). These settings differ in how  $N$  data examples and  $M$  features are naturally distributed among  $J$  parties. Samples and features that could be united to construct a centralized dataset  $D$  of shape  $\{X_i \in \mathbb{R}^M, Y_i\}_{i=1}^N$  in case of data privacy not being a concern (Zhang et al., 2020).

Table 1 formally describes data partitioning settings referring to the number of samples and features of a single party  $j$ . Those dimensions differ from the centralized dataset  $D$  depending on if horizontal or/and vertical partitioning are adopted. In HFL, parties differ in sample space but share the same features space  $M$  (McMahan et al., 2016; Sahu et al., 2018; Wang et al., 2020). While a single party  $j$  has a different number of samples  $N^j$ , the size of  $D$  is denoted by the union of the beneficiaries' samples  $\cup_{j=1}^J N^j$ . In VFL, the feature map of parties may overlap but share the sample indexes (Chen et al., 2020; Dai et al., 2021; Novikova et al., 2022; Yang et al., 2019). In other words, the feature map of  $D$  is denoted by the union of the beneficiaries' features  $\cup_{j=1}^J M^j$ . Finally, in Hybrid-FL, parties may differ in sample space and feature space (Hiessl et al., 2020; Zhang et al., 2020). In the hybrid data partitioning setting,  $D$  is composed of the

**Table 1** Distribution of samples and features according data partitioning settings: Horizontal, Vertical, and Hybrid Federated Learning

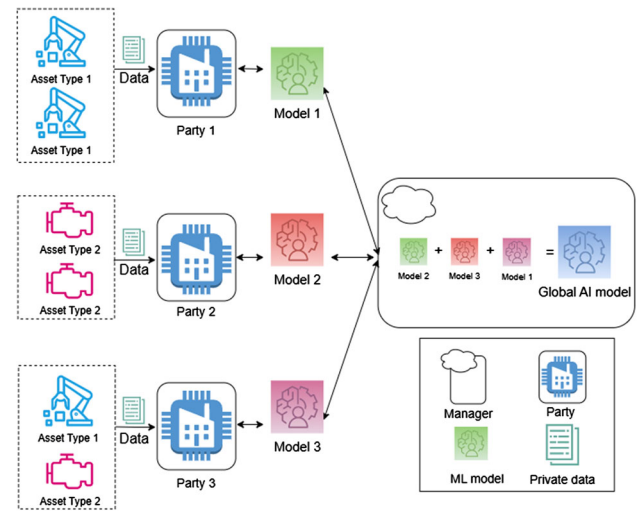
Data partitioning setting	Number of samples		Number of features		Condition for FL
	Per party $j$	Dataset $D$	Per party $j$	Dataset $D$	
Horizontal FL	$N^j$	$\cup_{j=1}^J N^j$	$M$	$M$	Same ANN structure
Vertical FL	$N$	$N$	$M^j$	$\cup_{j=1}^J M^j$	$\exists_j \in J   Y^j \therefore Y$
Hybrid FL	$N^j$	$\cup_{j=1}^J N^j$	$M^j$	$\cup_{j=1}^J M^j$	$\exists Y_{i,i+1,\dots,L} s.t. \sim_{j=1}^J Y^j$

union of the features and samples of the beneficiaries. Therefore, horizontal and vertical partitioning settings are special cases of Hybrid-FL (Zhang et al., 2020). However, most FL algorithms were designed to overcome those special cases separately due to the complexity of dealing with hybrid data partitioning settings.

In the case of FL algorithms compatible with Artificial Neural Networks (ANNs), one of the most used ML algorithms in industrial applications (Kofi et al., 2022), HFL and VFL solutions used to be conditioned beneficiaries to share the same ANN structure or label space, respectively. Recent studies aggregated knowledge of models differing in structure by distilling knowledge from a public dataset (Mora et al., 2022). However, these studies require parties to share a common label space for implementing a sample index synchronization technique initially proposed in VFL (Yang et al., 2019). A technique that does not fit with hybrid data partitioning settings.

The aggregation of knowledge from models exhibiting diverse structures due to the hybrid data partition formed by their training data needs attention within the FL context. This is particularly crucial as hybrid data partitioning frequently emerges in various real-world applications, including domains like manufacturing. For instance: quality prediction of things manufactured in similar stations; product inspection based on failure predictions in different stations (Ning et al., 2022); Remaining Useful Life (RUL) estimation of aircraft components in airlines (Rosero et al., 2022); monitoring of surface structures of coal mines using Electromagnetic Radiation Intensity (ERI) time series data of different producers (Yao et al., 2019); and among others.

This paper proposes a Hybrid-FL algorithm capable of aggregating knowledge of multiple sources whose data configure a hybrid data partitioning (Hiessl et al., 2020; Zhang et al., 2020). As far as our understanding extends, the presented algorithm FedLabSync stands as the pioneering formulation capable of aggregating insights from AI models (e.g. ANNs) of varying structures if at least they are compatible with the widely recognized Stochastic Gradient Descent (SGD) optimization algorithm. This is possible thanks to implementing a label synchronization strategy (Yang et al., 2019) capable of handling non-i.i.d. (identically and independently distributed) data distributing settings (Li et al., 2019).

**Fig. 2** Example of FL environment

Experimental results demonstrated that FedLabSync algorithm could achieve a global AI model with competitive results compared to a model trained in a data-centralized approach. Besides, the proposal presents performance improvements compared to models instructed using data from individual parties.

The main contributions of this paper can be summarized as follows:

1. FedLabSync is an algorithm that trains feed-forward ANNs and other AI models compatible with the SGD algorithm using a collaborative and penalized optimization approach. Besides FedLabSync being able to aggregate models differing in structure, it reduces communication costs because it shares predictions instead of model parameters.
2. The competitiveness of FedLabSync is demonstrated by conducting a set of evaluations in two industrial application scenarios: product inspection based on failure prediction in Bosch<sup>1</sup> production lines; and the RUL of aircraft components.<sup>2</sup> Experiments on each industrial sce-

<sup>1</sup> <https://www.kaggle.com/c/Bosch-production-line-performance>.

<sup>2</sup> <https://www.nasa.gov/content/prognostics-center-of-excellence-data-set-repository>.

nario compare the performance obtained by AI models trained in HFL, VFL, Hybrid-FL and a data-centralized setting without data privacy.

This article is organized as follows. “[Federated Learning](#)” section presents a literature review regarding major components of FL, settings and data partition approaches. “[Label synchronization for Hybrid Federated Learning](#)” section details the proposed Hybrid FL algorithm based on label synchronization. “[Experimental Setup](#)” section describes a manufacturing processes analysis and a predictive maintenance case study. “[Experimental results and analysis](#)” section describes the results obtained with the proposed algorithm, comparing them with HFL, VFL and a data-centralized scenario in which the predictive model is trained using data of whole parties. Finally, conclusions and future works are presented in “[Conclusions and future work](#)” section.

## Federated Learning

In contrast to a data-centralized learning approach where raw data is shared to form a global dataset, FL is a privacy-conscious alternative that utilizes models instead of data. FL aggregates models trained locally on individual parties and updates a global AI model through an iterative process (Abdulrahman et al., 2021; Li et al., 2019). This method ensures the privacy of raw data while still learning from it (Abdulrahman et al., 2021).

The interaction diagram of Fig. 2 illustrates the collaboration of three parties through a manager intervention. Depending on the number of selected parties, their computational resources and availability in aggregation iterations  $K$ , two FL settings have received particular attention, namely, cross-device FL and cross-silo FL (Li et al., 2019):

- *Cross-device* considers the participation of numerous beneficiaries in business-to-consumer (B2C) transactions whose participants are sometimes unavailable for every aggregation iteration, for example, mobile or IoT devices.
- *Cross-silo* considers a small number of parties, usually organizations or companies, fitting a business-to-business (B2B) transaction whose participants own powerful machines providing high availability (Kairouz, 2019).

The transaction type and the chosen data partitioning configuration significantly impact the initial stages of implementing federated applications. In other words, these factors heavily influence the FL key steps outlined in Fig. 1. Horizontal, vertical and hybrid data partition settings can be generalized considering that a dataset  $D$  of  $N$  data samples (composed of inputs  $X$  and labels  $Y$ ),  $M$  features and shape

$\{X_i \in \mathbb{R}^M, Y_i\}_{i=1}^N$  is distributed along  $J$  parties. Here, the feature dimension of the  $i$ th input sample corresponds to  $M$ , the number of its parameters  $\{p_1, p_2, \dots, p_M\}$ , therefore,  $X_i \in \mathbb{R}^M$ .

Considering a federated application of three participants, horizontal, vertical and hybrid data partitioning settings are illustrated in Fig. 3 and defined in Table 1. In this figure, coloured boxes represent inputs  $X$  of different parties while grey boxes represent their labels  $Y$ . In Table 1,  $N^j$  and  $M^j$  represent the number of samples and features of the  $j$ th party. Therefore, its dataset  $D^j$  takes a shape of  $\{X_i^j \in \mathbb{R}^{M^j}, Y_i^j\}_{i=1}^{N^j}$ .

In Horizontal Federated Learning (HFL), each party  $j$  owns different samples but shares the same feature space  $M$  with the other beneficiaries. In this sense, the global dataset  $D$  comprises the union  $\cup_j^J N^j$  of  $J$  partial datasets of size  $N^j$ .

In Vertical Federated Learning (VFL), the  $j$ th party dataset  $D^j$  of shape  $\{X_i^j \in \mathbb{R}^{M^j}, Y_i^j\}_{i=1}^{N^j}$  is composed of  $N$  samples. While the  $j$ th party has an input space  $X^j \in \mathbb{R}^{M^j}$  of dimension  $M^j$ , the dimension of the global dataset  $D$  corresponds to the union of the input space  $\cup_j^J M^j$  of  $J$  parties.

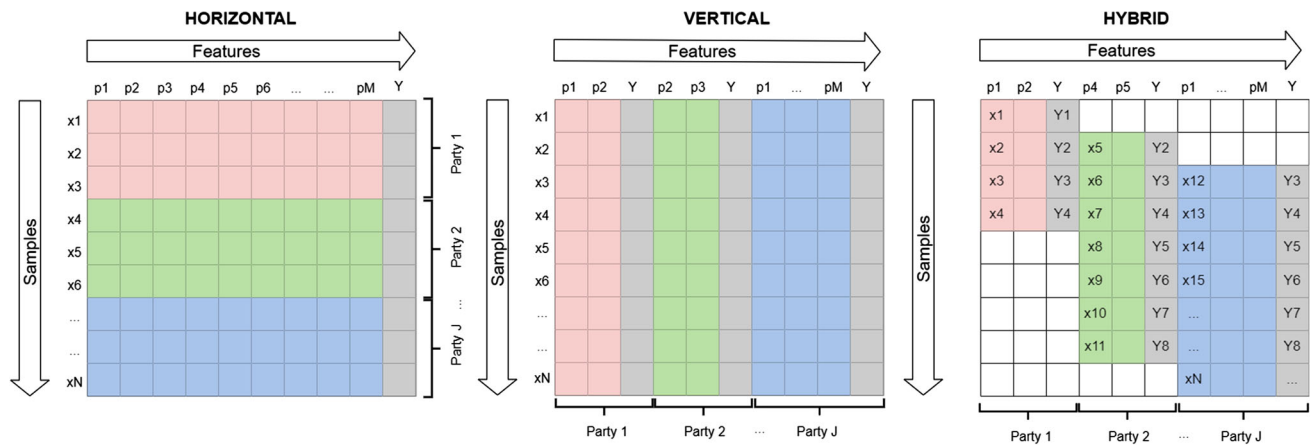
Finally, in Hybrid Federated Learning (Hybrid-FL), the  $j$ th party dataset  $D^j$  of shape  $\{X_i^j \in \mathbb{R}^{M^j}, Y_i^j\}_{i=1}^{N^j}$  is composed of  $N^j$  samples with an input space  $X^j \in \mathbb{R}^{M^j}$  of dimension  $M^j$ . Horizontal and vertical data partitions are exceptional cases of the hybrid one because it differs not only in sample space but may differ in input feature space.

## Horizontal Federated Learning

The predominant focus within the literature concerning Federated Learning (FL) algorithms has gravitated towards horizontal-based approaches, primarily propelled by the remarkable outcomes demonstrated by the pioneering FL algorithm, Federated Averaging (FedAvg) (McMahan et al., 2016). In dealing with non-i.i.d data distribution scenarios, HFL algorithms have aggregated knowledge of AI models with the same and different architectures by sharing model parameters and predictions (Mora et al., 2022; Reddi et al., 2020).

Most of the FL algorithms based on parameter aggregation utilize the SGD algorithm to orchestrate collaborative training of ANNs with the same architecture (Reddi et al., 2020), for example, Federated Stochastic Variance Reduced Gradient (FSVRG) (Konevny et al., 2016), FL via Momentum techniques (Felbab et al., 2019; Konevny et al., 2016; Liu et al., 2019), Federated Proximal Term (FedProx) (Sahu et al., 2018), Federated Stochastic Block Coordinate Descent (FedBCD) (Liu et al., 2019). On the other hand, FL algorithms able to aggregate knowledge of models differing in architecture utilize Knowledge Distillation (KD) techniques





**Fig. 3** Data partitioning settings considered in Federated Learning (Color figure online)

that require sharing model parameters, logits or intermediate features, for example, FedDistill (Jiang et al., 2020), MHAT (Hu et al., 2021) and FedDM (Gong et al., 2021).

Algorithms based on KD achieve their effectiveness by fine-tuning local models through the exchange of information derived from processing a publicly available dataset. When there is not possible to define a public dataset, algorithms SGD-based execute four key steps  $K$  times to minimize a global cost function  $F(w)$  (e.g. binary-cross entropy) and achieve the optimal weights  $w^*$  for the global AI model. Those steps, executed within a loop, are illustrated in Fig. 1.

In the  $\kappa$ th local training step, each party  $j$  minimizes a local cost function  $F^j(w^j)$  to get the optimal local parameters  $w^j$  for the local model. These optimal weights  $w^j$  at  $\kappa$  time are obtained by evaluating how accurately the model predicts the label of the  $i$ th data sample. Considering that  $N^j$  samples are composing the  $j$ th party dataset  $D^j$ , the local problem at  $\kappa$  time can be defined as follows:

$$w^j \triangleq \arg \min F^j(w^j) = \frac{1}{N^j} \sum_{i=1}^{N^j} f_i(w^j). \quad (1)$$

Calculating the  $f_i^j$  loss function (e.g. binary entropy) at each  $i$  sample is impractical to update local weights  $w^j$ . In this sense, best practices suggest evaluating mini-batches via cost function adoption. Even if mini-batches are not adopted, the core of FL algorithms SDG-based refers to solving the problem in Eq. (1) by updating  $w_t^j$  penalized by the step-size  $\eta$  of SGD algorithm:

$$w_{t+1}^j = w_t^j - \eta \nabla F^j(w_t^j). \quad (2)$$

Algorithms, such as FSVRG, FL via Momentum, FedProx, FedBCD, and others, propose alternatives for Equation (2) used by FedAvg. At  $\kappa$  time, after parties update their models, this latest algorithm computes the global AI model by simply

averaging their weights:

$$w_\kappa = \frac{\sum_{j=1}^J N^j w_\kappa^j}{N}. \quad (3)$$

After computing  $w_\kappa$ , weights are distributed to all parties to repeat the process. By performing  $K$  model aggregations, the problem of finding the optimal weights  $w^*$  for the global cost function  $F(w)$  can be solved:

$$w^* \triangleq \arg \min F(w). \quad (4)$$

FL algorithms SGD-based aim to improve the global model performance in a short number of model aggregations and guarantee learning convergence (Felbab et al., 2019; Konexny et al., 2016; Liu et al., 2019; McMahan et al., 2016; Sahu et al., 2018). In this way, FL algorithms based on other optimization algorithms modify the ANN architecture at each  $\kappa$  time, for example, Federated Matching Averaging FedMA (FedMA) (Wang et al., 2020). Alternatively, some frameworks use ML algorithms commonly used in VFL or Hybrid-FL, for example, SecureBoost (Cheng et al., 2019), SimFL (Li et al., 2019), and Support Vector Machines FL (Smith, 2017).

## Vertical Federated Learning

When parties do not share the same feature space but have the same sample identifiers, VFL considers synchronizing these identifiers (sample indexes) for training the global AI model. In this sense, various algorithms emerged: Federated Stochastic Block Coordinate Descent (FedBCD) (Novikova et al., 2022), Feature Distributed Machine Learning (FDML) (Hu et al., 2019), Heterogeneous Neural Networks (HeteroNN) (Yang et al., 2019), SecureBoost (Cheng et al., 2019), Vertical Asynchronous FL (VAFL) (Chen et al.,

2020), Asynchronous Federated Stochastic Gradient Descent (AFSGD-VP) (Gu et al., 2022), and others.

For some ML algorithms such as Support Vector Machines (SVM), linear and logistic regression, and ANNs, the problem of a single party  $j$  corresponds to finding the optimal weights  $w^j$  by minimizing the loss function  $f^j(w^j)$  (see Eq. 1). The sample indexes synchronization strategy ensures that the  $i$ th sample of each party dataset  $D^j$ , whose  $M$  parameters are distributed among parties, shares the same label  $Y_i$ . For that reason, it is needed at least one party sharing  $Y$  (see the condition in Table 1).

In a collaborative prediction, a global loss function  $\sigma(\cdot)$  evaluates how well the aggregation of  $J$  local predictions weighted by  $\alpha$  predicts  $Y_i$  (Chen et al., 2020; Dai et al., 2021). In other words,  $\sigma(\sum_{j=1}^J \alpha \hat{Y}_i^j, Y_i)$  evaluates the distance between collaborative prediction and ground truth in the  $i$ th sample. Since evaluating  $N$  samples in separate is impractical, the following global cost  $\xi(w)$  function is considered:

$$\xi(w) = \frac{1}{N} \sum_{i=1}^N \sigma \left( \sum_{j=1}^J \alpha \hat{Y}_i^j, Y_i \right) \quad (5)$$

In this sense, finding the optimal weights  $w^*$  for the global AI model could be simplified to the following expression:

$$w^* \triangleq \arg \min \xi(w) \quad (6)$$

Among the VFL algorithms that have adopted Equation (5) to train ANNs are VAFL (Chen et al., 2020), AFSGD-VP (Gu et al., 2022) and Vertical Federated Deep Learning (Dai et al., 2021; Gu et al., 2022). Remarkably, the last one presents deeper details of how the cost function  $\sigma(w)$  is minimized by considering block-wise coordination. The learning procedure of this approach is based on privately exchanging the party's predictions through the intervention of a computational server, which broadcasts current predictions to beneficiaries by constantly updating a prediction matrix  $A^{N \times J}$  of  $N$  rows and  $J$  columns.

The need for updating a prediction matrix  $A$  in a computational server is noticed in Algorithm 1. This synchronous VFL algorithm inspired proposals such as VAFL, AFSGD-VP and ours, presented in “[Label synchronization for Hybrid Federated Learning](#)” section. Algorithm 1 considers the availability of  $J$  parties during the learning process. Here, the server communicates with parties through pull and push requests, guaranteeing that parties receive only the necessary information privately. Considering that parties execute Algorithm 1 synchronously, the problem is solved within the global training loop of Line 12.

At each time  $\kappa$ , every party downloads corresponding weights  $w_\kappa^j$  from the global model  $w$ , performs and uploads

### Algorithm 1 Synchronous VFL

---

**Require:**  $J, D^j, \eta, w^j, A^{N \times J}, K$  and  $B$   
**Ensure:**  $w_K^j$  and  $w$

```

1: if Server Pull Request then
2:   if Batch Indexes  $B$  then
3:     Return batch sample indexes
4:   end if
5:   if Prediction Matrix  $A^{N \times J}$  then
6:     Return  $A^{N \times J}$ 
7:   end if
8: end if
9: if Server Push Request then
10:  Update  $A^{N \times J}$ 
11: end if
12: for  $\kappa \leftarrow 2$  to  $K$  do // Global training loop
13:  Pull  $w_\kappa^j$ 
14:  Predict  $\hat{Y}^j$ 
15:  Push  $\hat{Y}^j$ 
16:  Pull  $A^{N \times J}$ 
17:  for batch  $\leftarrow 2$  to Batches of size  $B$  do
18:    Pull Batch Indexes  $B // \hat{Y}$  in  $A^{B \times J} //$ 
19:     $\xi(w_\kappa) = \frac{1}{B} \sum_{i=1}^B \sigma(\sum_{j=1}^J \alpha \hat{Y}_i^j, Y_i)$ 
20:     $\nabla F^j(w_\kappa^j) = \phi^{L-1} \phi^{L'} \frac{\partial \sigma}{\partial \phi^L}$ 
21:     $w_{\kappa+1}^j = w_\kappa^j - \eta \nabla F^j(w_\kappa^j)$ 
22:  end for
23:  Push to server  $w_{\kappa+1}^j$ 
24: end for

```

---

predictions to the server  $\hat{Y}^j$ , updates local weights  $w_{\kappa+1}^j$  using mini-batches of size  $B$  and pushes them. The core steps of this algorithm are located within the nested loop of Line 17. These steps refer to the local training main step of Fig. 1.

The local training step is described in Line 33, where the loss function is derived w.r.t. the activation function of the last layer  $L$  of the local model  $\phi^L$  and multiplied by the remained chain rule derivatives to back-propagate the error. The resulting gradient of this operation is used to calculate the weights  $w_{\kappa+1}^j$  for the next  $\kappa$  global model computation time until achieving the problem convergence.

### Hybrid Federated Learning

In many engineering and industrial applications, parties not only differ in sample space but may also differ in feature space because industrial systems monitor different types of assets (Ning et al., 2022; Su & Lau, 2021; Yao et al., 2019). Independently from the application scenario, due to data distribution along parties usually corresponding to a non-independent and identical distribution (non-i.i.d.) scenario, some applications adopted KD techniques (e.g. Fine-Tuning) to fit models developed with datasets of other organizations under a hybrid data partition approach (Li et al., 2019). In these applications, encryption techniques (Agrawal et al., 2021; Cheng et al., 2019; Li et al., 2019) (e.g. Homomorphic encryption) helped to perform a Multi-Party Computation (Liu et al., 2018). In the context of Deep Learning (DL),

an algorithm based on Block Coordinate Descent named Hybrid Federated Matched Averaging (HyFEM) uses distance functions (e.g. Euclidean norm) and a Hungarian matching algorithm to solve a closed-form problem using Convolutional Neural Networks (CNNs) (Zhang et al., 2020).

Since Fine-Tuning and Deep Learning were primarily conceived for classification problems, Yao et al. proposed a model aggregation and fusion of features of multiple sensor signals for regression problems Yao et al. (2019). Despite the success of the model and feature aggregation, this approach loses valuable information because knowledge integration merges features of multiple signals before aggregating them using the well-known FedAvg algorithm.

As an alternative, the training of a Multitier-partitioned Neural Networks architecture was proposed. This architecture adopts the well-done Primal-Dual transform (Tran-Dinh & Zhu, 2019) and the Stochastic Gradient Descent Ascent (SGDA) algorithm (Deng & Mahdavi, 2021; Lin et al., 2019; Sebbouh et al., 2021) to decompose the problem in sample and feature spaces. Unfortunately, the applications for this alternative are limited to using AI models compatible with the primal-dual transformation, for example, logistic regression (Tran-Dinh & Zhu, 2019). Therefore, we propose a novel algorithm for AI models compatible with the SGD algorithm that can effectively tackle classification and regression problems under hybrid data partitioning settings.

## Label synchronization for Hybrid Federated Learning

Our proposal focuses on cross-silo Federated Learning settings, as it is suitable for business-to-business (B2B) transaction scenarios. In these scenarios, a limited number of isolated stations (silos) own high-performance machines, ensuring high availability during collaboration processes. The collaboration procedures, represented by the key steps in Fig. 1, are managed by a computational server. The parties communicate with the server through pull and push requests to train global and local AI models.

The design of our Hybrid-FL algorithm is motivated by the prevalent use of ANNs in industrial applications. We employ the widely used SGD algorithm to train feed-forward ANNs that may differ in architecture. Our approach is further influenced by Algorithm 1, which trains an ANN using the sample index synchronization technique. A technique that reduces communication costs given model parameter exchanges and allows the knowledge aggregation of multiple AI models differing in architecture and input feature space.

Sample indexes synchronization of VFL (Chen et al., 2020; Dai et al., 2021) can not be adopted because parties differ in sample space. Alternatively, collaboration processes may be conducted by synchronizing samples at the label level

(label synchronization) because samples of  $J$  parties may overlap in label space  $Y$ . The condition for parties collaborating on this hybrid data partitioning approach establishes that:

*Condition:* Given the labels  $Y^j$  of  $J$  parties, there is at least one instance of each one of the total sample's labels  $\mathbf{L}$  after performing the concatenation of parties labels  $\exists_{Y_{i,i+1,\dots,L}.s.t} \bigcap_{j=1}^J Y^j$ .

Like in VFL, the problem of Hybrid-FL also refers to finding the optimal weights for a global cost function  $\xi(w)$  (expression 5). A procedure that avoids evaluating a loss function  $\sigma(w)$  at each training sample for reducing computational operations.

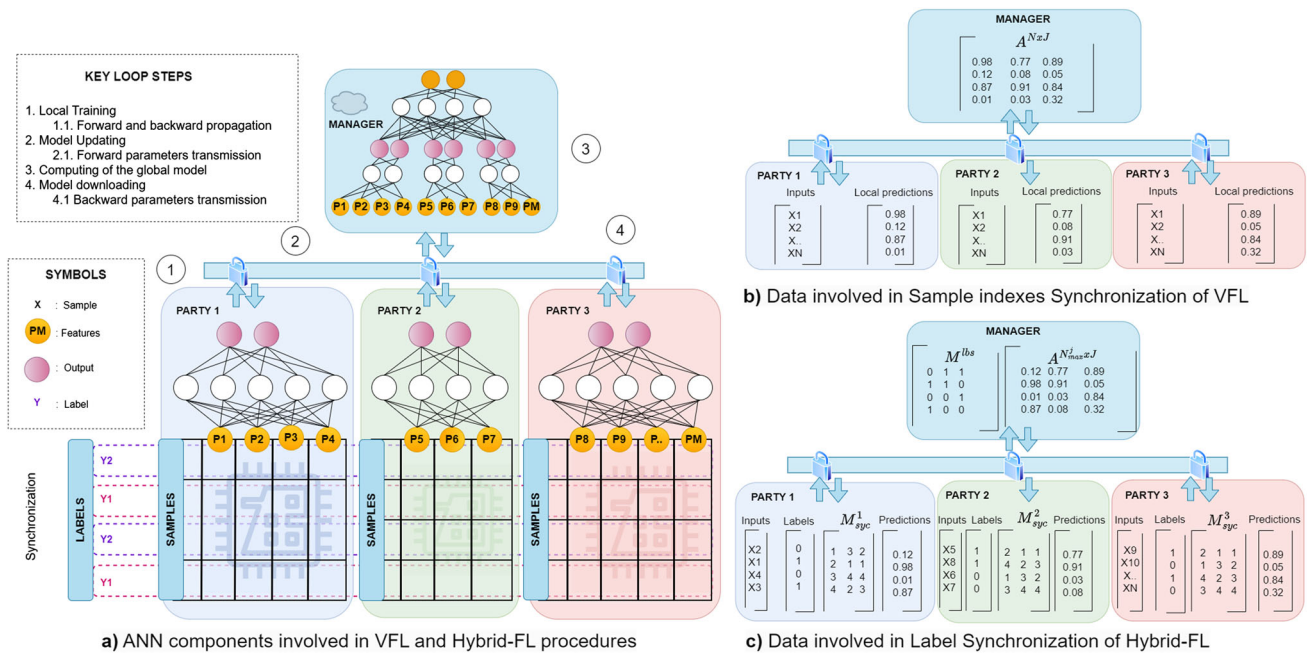
Arguing that parallel training procedures are executed by configuring mini-batches and parties differ in sample space, the block-wise coordination of VFL based on sample indexes fails. Figure 4 illustrates how the arrangement of samples and features in three-party dataset batches shows a similar and distinct ordering of input labels among the parties in VL and Hybrid-FL, respectively. In a parallel synchronous SGD program, parties on a VFL approach only require communicating their local predictions to update the server's prediction matrix  $A^{N \times J}$  because inputs share the same label map  $Y$ . Our Hybrid-FL FedLabSync, described in Algorithm 3, uses label synchronization strategy and extra matrices to find the optimal weights of  $\xi(w)$  based on block-wise coordination. These matrices are illustrated in Fig. 4.

## Synchronization

The proposed Label Synchronization (LabSync) strategy constructs parties synchronization label matrices  $M_{sync}$  by executing Algorithm 2. In this sense, every party  $j$  should firstly share its labels  $Y^j$  and the number of samples  $N^j$  with the server to construct the global label matrix  $M_{lbs}$ . The resulting values of matrix  $M_{sync}$  are then used to update and request local predictions of matrix  $A$ . Those predictions are used to conduct the synchronous SGD program described in Algorithm 3 without requiring parties to train models with whole data samples thanks to configuring mini-batches.

Considering that parties already shared their labels with the server and the  $M_{lbs}$  matrix (labels of parties) is already defined, gathering the label synchronization matrix  $M_{sync}$  for a determined party  $j$  is detailed in Algorithm 2. This algorithm, whose example results are illustrated in the label synchronization strategy of Fig. 4, aims to construct a matrix  $M_{sync}$  for a determined party  $j$  given the labels matrix  $M_{lbs}$  at the central computational server. The resulting matrix  $M_{sync}^j$  for the interested party  $j$  has a shape of  $N_{max}^j \times J$  because it is constructed using the maximum number of samples of  $J$  parties.





**Fig. 4** FL key loop steps, ANN components and data involved in synchronization techniques of Vertical FL and Hybrid-FL

### Algorithm 2 Label Synchronization LabSync

**Require:**  $M_{lbs}$ , interested party  $j$

**Ensure:**  $M_{sync}$

```

1:  $J \leftarrow \text{columns}(M_{lbs})$  // Number of parties
2:  $N_{max}^j \leftarrow \text{rows}(M_{lbs})$  // Maximum samples
3:  $M_{sync} \leftarrow \text{zeros}(N_{max}^j, J)$  // Fill with zeros
4:  $M_{sync}(:, j) \leftarrow (1 : N_{max}^j)^T$  // Set indexes
5:  $[Grs, Lbs] \leftarrow \text{groups}(M_{lbs}, j)$ 
6:  $M_{lc} \leftarrow \text{zeros}(\text{size}(Lbs), j)$ 
7: for  $k$  in  $J$  do
8:   for  $lb$  in  $Lbs$  do
9:      $M_{lc}(lb, k) \leftarrow \sum lb \in M_{lbs}(:, k)$ 
10:   end for
11: end for
12: for  $k$  in  $J$  do
13:   if  $k \neq j$  then
14:      $M_{lc} = \text{ones}(\text{size}(Lbs), J)$ 
15:     for  $lb$  in  $Lbs$  do
16:        $Idx_{lb} = \text{find}(lb \in M_{lbs}(:, k))$ 
17:       for  $i \leftarrow 2$  to  $N_{max}^j$  do
18:          $M_{sync}(i, j) \leftarrow Idx_{lb}(M_{lc}(lb, J))$ 
19:          $M_{lc}(lb, J) \leftarrow M_{lc}(lb, J) + 1$ 
20:         if  $M_{lc}(lb, J) > \text{size}(Idx_{lb})$  then
21:            $M_{lc}(lb, J) = 1$ 
22:         end if
23:       end for
24:     end for
25:   end if
26: end for

```

The core steps of Algorithm 2 perform the following tasks. Lines 5 and 6 group labels and initialize a counter for each label. The number of instances of each label for every party is calculated in the nested loop of Line 7. The nested for loop of Line 12 matches the  $i$ th sample of the interested

party  $j$  with the sample with the same label  $lb$  of the other party  $k$ . In response to non-i.i.d. data distributing settings, the conditional clause for the Line 21 ensures that two parties could have a different number of samples with the same label  $lb$ . It is possible by restarting the label counter  $M_{lc}(lb, j)$  of the  $j$ th party when the number of instances of a determined label  $Idx_{lb}$  is surpassed.

Labels and prediction matrices of Hybrid-FL in Fig. 4 illustrate how predictions of three parties can be accessed through indexes of  $M_{sync}$  matrices. For example, the first party accesses the predictions of matrix  $A$  related to the first instance of  $Y = 0$  by using the pointing indexes of the first row of  $M_{sync}^1$ . In this sense, the first party can privately access the predictions related to the first  $Y = 0$  occurrence of the second and third party in the following positions  $A(3, 2)$  and  $A(2, 3)$ .

The label synchronization process has to be accomplished before training any AI model to achieve the convergence of  $\xi(w)$  using the proposed Algorithm 3 named FedLabSync. Once parties communicate their labels and number of samples and get a matrix  $M_{sync}$ , the key steps enumerated in Fig. 4 are iteratively executed, namely: local training, model uploading, computing of the global model and model download.

### Local training

After label synchronization, local training is the most crucial step of FedLabSync algorithm because it comprises forward and backward propagation of ANNs in a parallel synchronous

SGD program. Considering that a single party  $j$  downloaded weights  $w^j$ , performed predictions via forward propagation  $\hat{Y}^j$  and uploaded them by using push and pull requests to the server, parties can get predictions for a single batch  $\hat{Y}^{B \times J}$  by using matrix  $M_{sys}$ .

The batch of predictions  $\hat{Y}^{B \times J}$  allows entities to calculate error  $\xi^j w^j$  (Line 32 of Algorithm 3) based on block-wise coordination. In Vertical Federated Deep Learning (Dai et al., 2021), the Mean Squared Error (MSE) loss function was adopted to evaluate the collaborative prediction of the  $i$ th sample:

$$\sigma(w) = \frac{1}{2} \left[ \sum_{j=1}^J Y_i^j - Y_i \right]^2, \quad (7)$$

where  $\sum_{j=1}^J Y_i^j$  corresponds to the collaborative prediction and  $Y$  represents the ground truth. Adopting the  $\sigma(w)$  function implies integrating outputs of  $J$  parties by adding extra layers to create a deeper ANN in the manager, as illustrated in Fig. 4.

In this approach, a single party can not achieve  $Y_i$  by itself because Equation (7) treats individual predictions equally. To solve this limitation in regression problems, we propose to adopt the following MSE loss function:

$$\sigma(w) = \frac{1}{2} \left[ \frac{\left[ \sum_{j=1}^J \alpha \hat{Y}_i^j \right] + \hat{Y}_i^j}{2} - Y_i \right]^2. \quad (8)$$

For binary classification problems, we propose to the following binary-cross entropy loss function:

$$\sigma(w) = -Y_i^j \log \left( \frac{\left[ \sum_{j=1}^J \alpha \hat{Y}_i^j \right] + \hat{Y}_i^j}{2} \right) + \dots + (Y_i^j - 1) \log \left( 1 - \frac{\left[ \sum_{j=1}^J \alpha \hat{Y}_i^j \right] + \hat{Y}_i^j}{2} \right) \quad (9)$$

In Eqs. (8) and (9),  $Y_i^j$  corresponds to the collaborative prediction by considering global  $\sum_{j=1}^J \alpha Y^j$  and local  $Y_i^j$  predictions for errors. In forward propagation, we considered a collaborative prediction weighted by  $\alpha$  in case the predictions of a particular party are most valued. Clearly, the collaborative prediction  $\frac{\left[ \sum_{j=1}^J \alpha \hat{Y}_i^j \right] + \hat{Y}_i^j}{2}$  is obtained by an

### Algorithm 3 FedLabSync: Hybrid-FL

---

**Require:**  $J, D^j, \eta, \beta, w^j, A^{N_{max} \times J}, K, B$   
**Ensure:**  $w_K^j$  and  $w$

```

1: if Server Pull Request then
2:   if Batch predictions  $\hat{Y}^{B \times J}$  then
3:     Return  $A^{B \times J}$ 
4:   end if
5: end if
6: if Server Push Request then
7:   if Partial dataset size  $N^j$  then
8:     Calculate  $N_{max}^j$ 
9:   end if
10:  if Partial labels  $Y^j$  then
11:    Creates or update  $M_{lbs}$ 
12:  end if
13:  if All predictions  $\hat{Y}^j$  then
14:    Update  $A^{N_{max}^j \times J}$ 
15:  end if
16:  if Batch predictions  $\hat{Y}^{B \times J}$  then
17:    Update  $A^{B \times J}$ 
18:  end if
19: end if
20: Push  $N^j$ 
21: Push  $Y^j$ 
22:  $M_{sys} \leftarrow \text{Algorithm 2}(M_{lbs}, j)$ 
23: for  $\kappa \leftarrow 2$  to  $K$  do
24:   Pull  $w^j$ 
25:   if  $\kappa = 0$  then
26:     Predict  $\hat{Y}^j$ 
27:     Push  $\hat{Y}^j$ 
28:   end if
29:   for  $batch \leftarrow 2$  to  $Batches$  of size  $B$  do
30:     Calculate batch of size  $B$  from  $M_{sys}$ 
31:     Pull Batch Predictions  $\hat{Y}^{B \times J}$ 
32:      $\xi(w_\kappa) = \frac{1}{B} \sum_{i=1}^B \sigma(\sum_{j=1}^J \alpha \hat{Y}_i^j, Y_i)$ 
33:      $\nabla F^j(w_\kappa^j) = \phi^{L-1} \phi^{L'} \frac{\partial \sigma}{\partial \phi^L}$ 
34:      $w_{\kappa+1}^j = w_\kappa^j - \eta \nabla F^j(w_\kappa^j)$ 
35:     Push  $\hat{Y}^j$  of size  $B \times J$ 
36:   end for
37:   Push to server  $w_{\kappa+1}^j$ 
38: end for

```

---

activation function  $\phi(\cdot)$  that will be used for gradient computation and backward propagation. This process is described in Line 33, where the loss function is derived w.r.t. the activation function of the last layer  $L$  of the local model  $\phi^L$  and multiplied by the remaining chain rule derivatives to back-propagate the error. The resulting gradient of this process is used to calculate the weights  $w_{\kappa+1}^j$  for the next  $\kappa$  global model computation time.

### Model uploading, computing, and downloading

At  $\kappa$  time (see nested loop of Line 23), after parallel updating the weights  $w_{\kappa+1}^j$  of each party  $j$ , weights are pushed to server together with the new predicted values  $\hat{Y}^j$  (see Lines 35 and 37). The weights of all parties are then used to update the global AI model stored by the manager.

Instead of blending ANNs like in HFL, VFL and HybridFL join these neural networks to create a more robust network. For example, in Fig. 4, three parties share their model parameters, allowing the manager to create a deeper neural network, i.e. weights  $w^j$  of a particular party  $j$  update a subspace of the global model.

Distributing weights of the global AI model (e.g. by executing Line 24) allows parties to restore the weights to the  $k$ th time. However, for VFL and Hybrid-FL, global model distribution has to be accomplished by predictions of the beneficiaries. This allows local training procedures to be repeated until the problem convergence is achieved while reducing communication costs.

## Experimental setup

This section presents industrial case studies in a standard experimental setup. While case studies are introduced in “Case study 1: manufacturing processes analysis—failure prediction” and “Case study 2: predictive maintenance—RUL estimation” sections, details about the experimental setup are presented in “Test scenarios” section. The first case refers to manufacturing processes analysis, concretely in failure prediction of objects manufactured along multiple Bosch production Lines (Ning et al., 2022). On the other hand, the second case refers to the predictive maintenance area, specifically in the RUL prediction of aircraft components distributed along aerial fleets (Rosero et al., 2022, 2020). While failure detection corresponds to a binary classification problem, RUL prognosis corresponds to a regression problem. By adopting classification and regression, we make the FedLabSync analysis more robust.

The experimental setup aims to compare the following:

1. The performance of AI models in three data partitioning scenarios (horizontal, vertical and hybrid) and a data-centralized scenario.
2. The model’s performance of each party  $w^j$  trained using partial data  $D^j$  or achieved using the FedLabSync FL algorithm.

### Case study 1: manufacturing processes analysis—failure prediction

The demand for high-quality products forced manufacturing industries to consider new methods and tools that use data modelling, simulation, expert systems, reference models and decision-making support (Hernandez et al., 2006). In this sense, integrating AI and digitalization into manufacturing processes has presented a transformative opportunity for optimizing various production activities (Anghel et al., 2018). Notably, this convergence has elevated the pursuit of

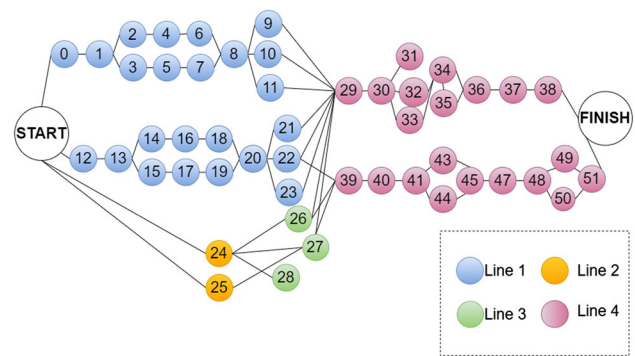


Fig. 5 Stations and production lines of Bosch dataset

quality enhancement to paramount importance (Ning et al., 2022; Zhenyu et al., 2020; Kofi et al., 2022; Hernandez et al., 2006).

## Problem

Anticipating the products that necessitate inspection diminishes the prevalence of defective items and intricately fine-tunes the quality control procedures (Ning et al., 2022). Consequently, determining which products undergo inspection routinely draws upon the bolstering capabilities of ML failure prediction (Kofi et al., 2022). However, prognosticating outcomes for products assembled across numerous workstations has proven to be an intricate challenge (Moldovan et al., 2019). As a result, most predictive methodologies rely on a data-centralized learning approach.

From three of the most illustrative manufacturing processes datasets (SECOM,<sup>3</sup> SEFTI<sup>4</sup> and Bosch<sup>5</sup>), we considered using the last one for constructing FL scenarios. We chose the Bosch production lines dataset (14.3 GB) because it aims to prioritize product inspection based on failure predictions and because products are ensembled in multiple workstations and production Lines (Ning et al., 2022). Therefore, it simulates a real-world FL scenario.

Bosch, one of the leading manufacturing companies, measured and tested the assembling of 1,184,687 and 1,183,748 products (samples), respectively. Each sample has different assembly processes. Figure 5 shows that each sample can present features at a maximum of 52 stations ( $S0-S51$ ) located at four production lines ( $L0-L3$ ).

Each sample has three types of features: numerical, categorical, and date features (Zhenyu et al., 2020). Based on date features, it is possible to get the time stamp of each side the product passes. Studies have constructed Long Short-Term

<sup>3</sup> <http://archive.ics.uci.edu/ml/index.php>.

<sup>4</sup> <http://www.causality.inf.ethz.ch/>.

<sup>5</sup> <https://www.kaggle.com/c/Bosch-production-line-performance/overview>.

**Table 2** Samples and features of Bosch dataset

Samples		Numerical features			
Positive	Negative	L0	L1	L2	L3
1,177,808	6879	168	513	42	245

Memory (LSTM) networks to obtain the long-term dependence on time series data (Carbery et al., 2019). Since our study does not pretend to get time dependency, we used just numerical features as in Ning et al. (2022). We considered using 1,184,687 samples and 968 numerical features (continuous values) distributed along production lines. Table 2 describes the number of features per production line and the positive and negative samples for this case study, namely, samples that passed and did not pass the quality control process, respectively.

### Evaluation metrics

The prediction task of Bosch corresponds to a binary classification problem. Therefore, the output dimension of the decoding model is set to 1 when a product passes the quality control process and set to 0 when a product inspection is needed. Following previous studies in manufacturing monitoring (Moldovan et al., 2019; Ning et al., 2022; Zhenyu et al., 2020), we considered evaluating this case study using accuracy and F-score binary classification metrics:

$$Precision(P) = \frac{TP}{TP + FP} \quad (10)$$

$$Recall(R) = \frac{TP}{TP + FN} \quad (11)$$

$$F\text{-score} = \frac{2 * P * R}{P + R} \quad (12)$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (13)$$

where  $TP$  corresponds to the number of true positives,  $FP$  to false positives,  $TN$  to true negatives and  $FN$  to false negatives.

### Model construction

Assuming that a series of features for each sample is named according to the L production line and the stations S that pass (e.g. L0\_S1\_D1 for the first feature), we can perform the ML development stages. For the current binary classification problem, two main stages are required to develop an ML learning model: data cleaning and feature selection.

According to Ning et al. (2022) and Carbery et al. (2019), only a few workstations collect data from most products, requiring exhaustive data-cleaning procedures. After

**Table 3** Composition of train and test splits of Bosch dataset

Samples	Train 60%	Eval 20%	Test 20%
Positive	197,962	65,987	65,987
Negative	65,329	21,775	21,775

analysing the proportion of missing observations per feature and sample (product ID), Carbery C.M. et al. suggest that data-cleaning has to be produced at two levels: features and samples (Carbery et al., 2019).

At the first level, the 1,183,748 products of the test split are not considered because of the unavailability to identify them as positive or negative samples. According to Zhenyu et al. (2020) and Carbery et al. (2019), cleaning duplicated samples and samples containing more than 142 features with missing values is needed. Finally, features with duplicated names, zero variance, and those with more than 70% of missing values are discarded. These data-cleaning procedures should give a resulting dataset of 1,094,995 observations and 163 features, which allows the conduction of the feature selection stage.

Selecting the variables that influence the outcome employed the Principal Component Analysis (PCA). The application of PCA is performed in two groups of features. While the first group comprises features of L0, L1 and L2 production lines, the second group refers to features of L3. Applying PCA aims to reduce the feature space to 22 in each group because, according to (Carbery et al., 2019; Zhang et al., 2016), the first 22 dimensions of each group can represent more than 95% of the variance.

Since the proportions of positive and negative samples of Table 2 point to an unbalanced dataset, we considered adopting oversampling techniques before training the AI model. We applied the synthetic minority oversampling technique (SMOTE) such as in Carbery et al. (2019). We configured SMOTE to randomly reduce the whole dataset at 10% with a proportion of 1:3 related to negative and positive samples. This oversampling obtained 44 features, 329,936 positive and 108,879 negative samples.

Finally, as described in Table 3, we distributed the data to train and evaluate the construction of the AI model with 60% and 20% of the total samples. The remaining 20% is used to test the model. Details about the ML algorithm and the performance achieved in the experimental setup scenarios are presented in “[Experimental results and analysis](#)” section.

### Case study 2: predictive maintenance—RUL estimation

In the Prognostics and Health Maintenance (PHM) discipline, predictive maintenance is a strategy based on



Condition Monitoring (CM) data that aims to predict the future states of machinery health condition by developing data-driven models (Yu et al., 2021; Rosero et al., 2022). Therefore, reducing maintenance costs and downtime (Luis et al., 2021; San & Young, 2021). Mainly, predictive maintenance has been used to determine the advent of a failure by applying RUL concepts (Khaled & David, 2022; San & Young, 2021), i.e. methods that predict the remaining time an asset is estimated to be able to function without failing (Rosero et al., 2022; Saxena et al., 2008a).

## Problem

In the aviation industry, the interest in predictive maintenance increases due to the need to accomplish strict safety and operational reliability policies (Rosero et al., 2022; Luis et al., 2021). As a response, various methods have been utilized to predict the health of aircraft structures, systems, and components using available sensor data and ML algorithms (Scott et al., 2022). Furthermore, since airlines monitor equivalent aircraft elements, a private collaboration among fleets via Federated Learning gained attention (Rosero et al., 2020).

Given the difficulty in obtaining a significant percentage of failure instances for CM datasets in aerospace, digital twin systems have supported the advances in predictive maintenance. For instance, the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) of The National Aeronautics and Space Administration (NASA<sup>6</sup>) allowed industrial and academics to extend the literature on RUL prediction of turbofan engines. Public datasets of the C-MAPSS simulator comprise data from multiple turbofan engines: Turbofan Engine Degradation Simulation (Saxena et al., 2008a), PHM08 challenge (Saxena et al., 2008b), and Turbofan Engine Degradation Simulation-2 (Arias Chao et al., 2021).

Each engine (aircraft component), identified by a unique number ID in whatever dataset, is monitored along several operating cycles (flight hours). At each cycle, the RUL of the component is related to measurements of 21 sensors and three operating settings: altitude, Mach Number (MN) and Throttle Angle Resolver (TAR).

We considered using the FD004 dataset of Saxena et al. (2008a) to evaluate FL algorithms because it is composed of labeled training and testing data splits (Table 4). The dataset for FD004 consists of run-to-failure trajectories for 249 components in the training split, with a corresponding RUL value provided for each operating cycle. On the other hand, the testing split is comprised of data for 248 components that were monitored up until a few operating cycles prior to their end-of-life. Considering that components of FD004 experienced

**Table 4** Composition of train and test splits of C-MAPSS

Data	Train 42%	Eval 8%	Test 50%
Trajectories	210	39	248

two types of failures after working in six operating regimes, the problem of FD004 refers to how precisely the RUL values in the testing split are after training an AI model using the training data split.

## Evaluation metrics

Following a series of studies in predictive maintenance (Olivares et al., 2019; San & Young, 2021), mainly in those based on C-MAPSS datasets (Rosero et al., 2022, 2020; Saxena et al., 2008a), we evaluate the performance of RUL prediction models using the Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) metrics:

$$MAE = \frac{1}{N} \sum_{i=1}^m |RUL_i - \hat{RUL}_i|, \quad (14)$$

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^m (RUL_i - \hat{RUL}_i)^2}, \quad (15)$$

where  $N$  corresponds to the number of samples,  $RUL$  is the ground truth (label) for the  $i$ th sample and  $\hat{RUL}_i$  is the estimated remaining life.

## Model development

A typical PHM program experiences three primary stages to construct the prediction model: data acquisition, pre-processing, and prognostic model. We constructed the run-to-failure instances in data acquisition by ordering the data by component ID and operating cycle. Next, we adopted the pre-processing data steps from Rosero et al. (2020). Pre-processing data steps correspond to defining a degradation function, normalization and feature space selection.

The degradation function, composed of two health degradation stages, assumes that components experience an imperceptible degradation until crossing an elbow point where the engines degrade abnormally (Rosero et al., 2022). Formally, considering that C-MAPSS datasets provide the RUL value of the last operational cycle per engine  $t_{EoL}$ , and considering an initial constant RUL value  $R_c$ , the RUL of turbofan linearly decreases after reaching a start to failure  $t_{SoF}$  or elbow point. Following studies in FD004 (Rosero et al., 2020; Saxena et al., 2008a), we set  $R_c$  in 120 flight hours and  $t_{SoF}$  as the positive difference between  $t_{EoL}$  and  $R_c$ .

<sup>6</sup> <https://www.nasa.gov/content/prognostics-center-of-excellence-data-set-repository>.



$$f(t) = \begin{cases} Rc & \text{if } 0 \leq t \leq t_{SoF} \\ t_{EoL} - t & \text{if } t_{SoF} \leq t \leq t_{EoL} \end{cases} \quad (16)$$

The data normalization procedure, adopted from Olivares et al. (2019) and Saxena et al. (2008a) applies the K-means algorithm before normalizing the samples per operational regime (e.g. landing and taking off). Clustering involves relating each sample with one of the six operating phase centres of FD004 defined by the combination of altitude, MN and TRA. After relating each sample  $X_i$  with a determined regime  $r$ , the data normalization function  $N(\cdot)$  is applied:

$$N(X_i^{(r,f)}) = \frac{X_i^{(r,f)} - \mu^{(r,f)}}{\sigma^{(r,f)}}, \quad (17)$$

where each sensor  $f$  on regime  $r$ ,  $X_i^{(r,f)}$  represents the sensor data per regime,  $\mu^{(r,f)}$  and  $\sigma^{(r,f)}$  corresponds to the mean and the standard deviation. Then, according to Olivares et al. (2019) and Sahu et al. (2019), the sensors that better represent the degradation of aircraft components are 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17, 20, 21. These 16 sensor measurements (features) and the RUL value (label) are inputs for constructing the AI model.

We considered training and evaluating the construction of the AI model using 210 and 39 run-to-failure trajectories of the training dataset. The 248 run-to-failure trajectories of the test split are used to compute the performance of the resulting prognostic model for an FL or a data-centralized scenario. Details about the ML algorithm and the performance achieved in the experimental setup scenarios are presented in “Experimental results and analysis” section.

## Test scenarios

Three data partitioning scenarios (horizontal, vertical and hybrid) and a data-centralized scenario compose the experimental setup of each industrial case use. “Case study 2: predictive maintenance—RUL estimation” and “Case study 1: manufacturing processes analysis—failure prediction” sections briefly introduced the samples and features used in each case study in a data-centralized scenario, but details about data partitioning scenarios are missing. Depending on the case study adopted, Table 5 describes how samples and

features of a data-centralized scenario can be distributed to configure horizontal, vertical and hybrid data partitions.

In the case of the manufacturing processes case study, the  $N$  samples of the training and testing splits of the Bosch dataset are equally distributed to  $J$  parties. It also happens in features but at a production line level. Since we considered two parties, owing features from L0–L2 and L3 production lines, each party will use 22 PCA features. More details of hyperparameter tuning are presented in Table 6.

Since the predictive maintenance case study analyzes run-to-failure data of aircraft components, we considered a party storing data of an airfleet with multiple aircraft, in consequence, multiple components. In other words, a single party can visualize an entire run-to-failure trajectory of the training dataset. In this sense, sample distribution is performed considering 249 run-to-failure trajectories of dimension  $dim()$ . In the case of feature distribution, each party  $j$  randomly selects 80% of the available features. In other words, 13 sensors are selected without repositioning from 16 sensors.

## Number of parties

The distribution of samples and features of Table 5 is generalized for  $J \in \mathbb{N}$ . However, selecting the number of parties  $J$  depends on the number of samples. In case of the Bosch dataset, we fixed the number of parties at 4 to avoid losing information. However, in the case of the predictive maintenance case study, we considered more than one value for  $J$ .

Considering that a single party  $j$  train and evaluate a model with 2 run-to-failure trajectories, a maximum of 124 parties can compose the FL using the FD004 dataset. However, our experimental setup for this case study considered short values for  $J$  to ensure the convergence of the problem.

The distribution of samples and features for both case studies are detailed in Table 7. While sample distribution considers training and evaluation splits, feature distribution does no consider a split in particular. Sample distribution is performed using the  $Jn$  selection criterion, which means that samples are distributed using the modulo operation after being shuffled. It is noticeable in the C-MAPSS dataset that the mean of run-to-failure trajectories decreases when  $J$

**Table 5** Dimension of samples and features of the experimental setup scenarios at each case study

	Number of samples (train and eval splits)		Number of features	
	Bosch	C-MAPSS	Bosch	C-MAPSS
Horizontal	$N^j = N/J$	$N^j = \dim(249 \text{ run-to-failure trajectories } / J)$	$M^j = 44$	$M^j = 16$
Vertical	$N$	$N$	$M^j = 22$	$M^j = 13$
Hybrid	$N^j = N/J$	$N^j = \dim(249 \text{ run-to-failure trajectories } / J)$	$M^j = 22$	$M^j = 13$
Data-centralized	$N = 351,053$	$N = \dim(249 \text{ run-to-failure trajectories } )$	$M = 44$	$M = 16$

**Table 6** Bosch hyperparameter tuning

Hyperparameter	Search range
Hidden layers	2, 3, 4
Nodes per layer	5, 10, 15, 20, 25
Activation function	<b>relu</b> , tanh, sigmoid
Step-size $\eta$	0.2, 0.1, <b>0.01</b> , 0.001
Batch size $B$	<b>32</b> , 64, 96

increases, the reason for which we considered experimenting with a maximum of eight parties.

### Selection of features

Feature distribution is also different in both case studies. While the manufacturing process case study aims to distribute features of two production line groups (L0–L2 and L3), the feature selection of C-MAPSS corresponds to getting a subgroup of 13 input sensors. The number of possible combinations is calculated using the following formula:

$$(13J)C(16J) = \frac{(16 * J)!}{(13 * J)! * ((3)J)!} \quad (18)$$

Since the number of possible combinations exponentially increases w.r.t.  $J$ , evaluations of “[Experimental results and analysis](#)” section considered a subset of 30 combinations.

## Experimental results and analysis

This section uses processed input data from failure prediction and RUL estimation case studies and adopts a feed-forward ANN to construct AI models for the experimental setup scenarios. Then, the performance achieved by AI models in these

classification and regression problems is separately analyzed in “[Case study 1: failure prediction](#)” and “[Case study 2: RUL estimation](#)” sections. All experiments described in this paper were executed on a computer with AMD Ryzen 9 3900X 12-Core processor, 64 GB RAM, NVIDIA GeForce RTX 3080 GPU, Ubuntu 20.04 LTS 64-bit operating system and MATLAB R2021a. Training a single AI model for these experiments took from 5 to 56 min on average, which mainly depends on the quantity of data processed.

### Case study 1: failure prediction

Independently from the experimental setup scenario, the hyperparameters of the ANN constructed to solve this binary classification problem are resumed in Table 8. This configuration was adopted after constructing the AI model of the data-centralized scenario, a process in which hyperparameters of Table 6 were used to conduct a grid search model.

After constructing 30 models with different initial weights  $w_0$ , the mean classification performance was calculated for the data-centralized scenario. The referred performance, in terms of accuracy and F-score, is described in Table 9 and illustrated by the ROC curve of Fig. 6. Since an accuracy of  $0.866 \pm 0.003$  and an F-score of  $0.82 \pm 0.004$  seem to be comparative with previous works (Moldovan et al., 2019; Ning et al., 2022; Zhenyu et al., 2020), the hyperparameters in bold of Table 6 were adopted for the remaining scenarios.

### Data partitioning scenarios

In Federated Learning approaches, the accuracy and F-score values described in Table 9 were also calculated after evaluating the 30 different AI global models with the testing data split. For data partitioning purposes, whose details are described in Tables 5 and 7, we distributed samples of the training data split among four parties using the 4n selection

**Table 7** Samples and features of failure prediction and RUL estimation problems when J varies

Dataset	J	Samples (train and eval splits)		Features	
		Selection criterion	$\mu$	Selection criterion	Combinations
Bosch	4	4n	$\approx 87,763$	[L0–L2, L3, L0–L2, L3]	1
C-MAPSS	2	2n (trajectory)	$\approx 124$ trajectories	13 <b>C</b> 16	$3.65 \pm 10e+32$
	4	4n (trajectory)	$\approx 62$ trajectories	13 <b>C</b> 16	$2.64 \pm 10e+80$
	8	8n (trajectory)	$\approx 31$ trajectories	13 <b>C</b> 16	128 <b>C</b> 104

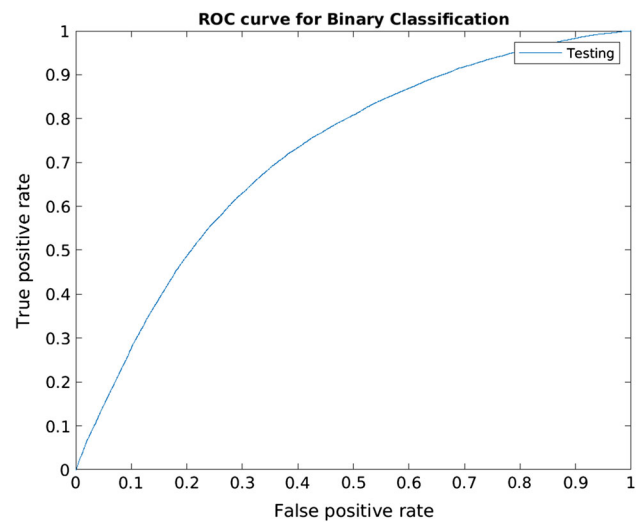
**Table 8** ANN hyperparameters for failure prediction and RUL estimation problems related to Bosch and C-MAPSS datasets

Dataset	Nodes per layer	Activation	Loss function	Metric	$\eta$	$B$
Bosch	(10,10,1)	(relu, relu, relu)	Binary entropy	Accuracy and F-score	0.01	32
C-MAPSS	(10,1)	(sigmoid, relu)	RMSE	RMSE and MAE	0.01	128

**Table 9** Performance of AI models at each experimental setup scenario by training 10 ANNs with different weights  $w_0$ 

J	Data-centralized			Hybrid FL			Vertical FL			Horizontal FL		
	Accuracy	F-score	RMSE	Accuracy	MAE	RMSE	Accuracy	MAE	RMSE	Accuracy	MAE	F-score
Bosch												
<b>4</b>	$0.866 \pm 0.003$	$0.820 \pm 0.004$	$24.39 \pm 2.11$	$0.856 \pm 0.002$	$20.95 \pm 1.51$	$25.06 \pm 1.34$	$0.891 \pm 0.013$	$24.64 \pm 2.05$	$27.01 \pm 1.56$	$0.842 \pm 0.006$	$24.05 \pm 2.48$	$0.788 \pm 0.008$
C-MAPSS												
<b>2</b>	$20.59 \pm 2.23$				$26.99 \pm 0.39$	$30.21 \pm 0.59$		$23.7 \pm 1.22$	$26.06 \pm 0.88$		$23.37 \pm 1.39$	$26.38 \pm 0.98$
<b>4</b>				$26.72 \pm 0.64$		$29.86 \pm 1.01$		$24.56 \pm 1.36$	$26.88 \pm 1.13$		$24.8 \pm 2.61$	$27.86 \pm 2.51$
<b>8</b>												

Number of J in bold refers to experiments in which Hybrid-FL presented the best results

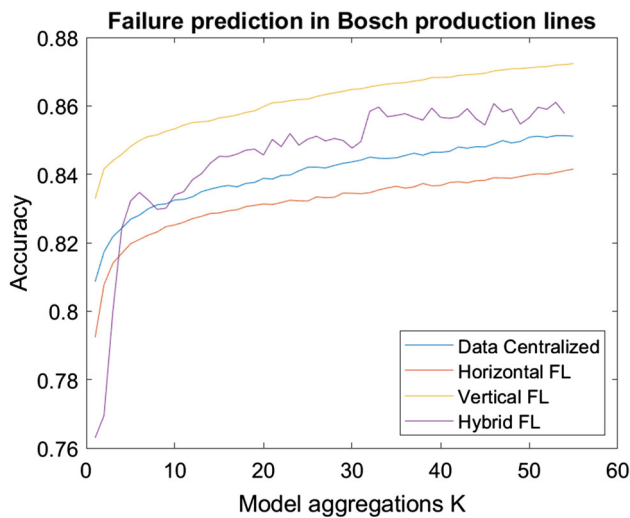
**Fig. 6** ROC curve of failure prediction in Bosch production lines using testing data of Table 2

criterion. Regarding feature distribution, we simulated the collaboration of 4 parties ( $\alpha = 4$ ) sharing features of 2 production lines:  $L0-L2$  and  $L3$ . Since a single party shares approximately 87763 samples and features of one group of production lines, there is no loss of information for the hybrid data partitioning scenario. Thus, a direct comparison among FL algorithms is guaranteed.

We considered comparing the classification performance using FedAvg, Vertical Synchronous VFL and FedLabSync algorithms for horizontal, vertical, and hybrid data partitioning scenarios. The performance on each scenario, detailed in Table 9, was calculated after testing 30 global AI models trained with different initial weights  $w_0$ . All models of this table were constructed by executing 400 model aggregations  $K$  or epochs  $E$  in the case of the data-centralized scenario. Using the evaluating data split, Fig. 7 illustrates the failure detection accuracy calculated at the  $k$ th model aggregation. This figure zooms in on the first 60 first  $K$  model aggregations in which the proposed FedLabSync algorithm learns faster than FedAvg. Since the prediction performance of Hybrid-FL oscillates over the accuracy of the data-centralized scenario, we argue that those oscillations are related to two factors. While the first is related to weightily averaging parties' logits (softmax activation output), the second is related to calculating the collaboratively error using the cross-binary loss function of Eq. 9.

### Performance at each party

Besides comparing the failure prediction performance of global AI models, this paper also compares the performance of isolated models trained using partial data  $D^j$  with the models trained using FL algorithms. After training 30 ANNs with



**Fig. 7** Failure detection accuracy at  $k$ th model aggregation in all data partitioning scenarios (Color figure online)

different initial weights, the mean performance was calculated for every party  $j$ . Then, the results of these experiments were summarized in Table 10. Noticeably, the failure prediction at each production line has been improved by using FL algorithms, with the VFL being the most favourable scenario. For example, the fourth party obtained a classification accuracy of  $0.891 \pm 0.013$ ,  $0.865 \pm 0.001$ ,  $0.842 \pm 0.006$  and  $0.825 \pm 0.011$  in vertical, hybrid, horizontal and partial data  $D^4$ , respectively.

## Case study 2: RUL estimation

A unique model architecture was defined to compare the RUL estimation accuracy obtained in the different data partitioning scenarios. This model architecture, adopted from Rosero et al. (2020) and Olivares et al. (2019), is composed of an ANN followed by a Kalman Filter (KF) used for prediction noise reduction. While Table 8 details the ANN hyperparameters, the KF is described by Algorithm 4.

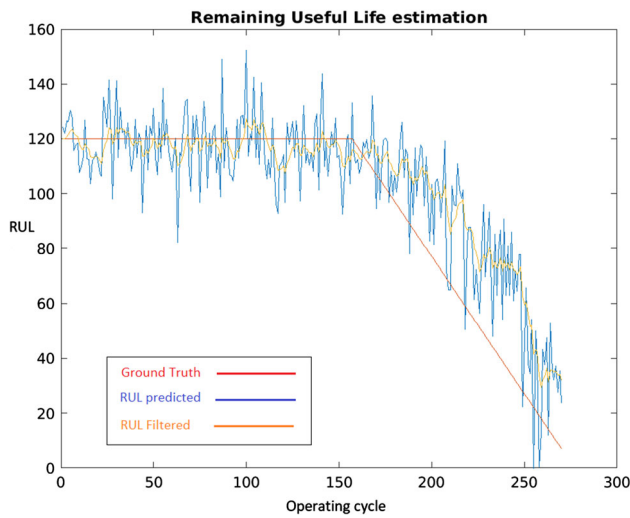
Particularly, this KF is applied to the entire run-to-failure state vector and consists of prediction and update stages (Olivares et al., 2019). Those stages compose the inner loop of Line 2. The prediction stage comprises the Lines 3–5, in which:

- $\hat{Y}_k^-$  is the *priori* estimate of the state vector  $\hat{Y}$  at time  $k$ ,
- $P_k^-$  is the *priori* error estimate matrix,
- $P$  is the *posteriori* error estimate matrix, and
- $Q = 1/209$  is the degradation rate corresponding to 209 operating cycles in average Olivares et al. (2019).

**Table 10** Performance of models trained with partial data  $D^j$  and models trained using FL algorithms

Party $j$	Partial data $D^j$		Hybrid FL		Vertical FL		Horizontal FL	
	Accuracy	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy	F-score
<b>Bosch</b>								
<b>1</b>	$0.821 \pm 0.013$	$0.760 \pm 0.028$	$0.866 \pm 0.003$	$0.831 \pm 0.006$	$0.891 \pm 0.013$	$0.807 \pm 0.157$	$0.842 \pm 0.006$	$0.788 \pm 0.008$
<b>2</b>	$0.822 \pm 0.004$	$0.763 \pm 0.007$	$0.865 \pm 0.005$	$0.821 \pm 0.005$	$0.891 \pm 0.013$	$0.807 \pm 0.157$		
<b>3</b>	$0.822 \pm 0.004$	$0.822 \pm 0.004$	$0.866 \pm 0.006$	$0.822 \pm 0.005$	$0.891 \pm 0.013$	$0.807 \pm 0.157$		
<b>4</b>	$0.825 \pm 0.011$	$0.825 \pm 0.011$	$0.865 \pm 0.010$	$0.830 \pm 0.014$	$0.891 \pm 0.013$	$0.807 \pm 0.157$		
<b>C-MAPSS</b>								
<b>1</b>	$20.77 \pm 0.42$	$24.43 \pm 0.34$	$20.45 \pm 3.84$	$26.5 \pm 2.04$	$24.31 \pm 0.64$	$26.43 \pm 0.65$	$23.37 \pm 1.39$	$26.38 \pm 0.98$
<b>2</b>	$23.27 \pm 0.46$	$26.89 \pm 0.59$	$20.05 \pm 3.43$	$26.01 \pm 2.06$	$24.74 \pm 0.88$	$26.85 \pm 0.76$		
<b>3</b>	$21.83 \pm 0.53$	$25.53 \pm 0.52$	$24.29 \pm 4.03$	$27.79 \pm 3.96$	$23.92 \pm 0.67$	$26.3 \pm 0.56$		
<b>4</b>	$21.25 \pm 0.55$	$25.23 \pm 0.7$	$21.77 \pm 4.52$	$27.39 \pm 2.7$	$24.42 \pm 0.51$	$26.55 \pm 0.54$		

Party  $j$  in bold refers to experiments in which Hybrid-FL improved the performance of models trained using partial data  $D^j$



**Fig. 8** RUL estimation of the first aircraft component in the data-centralized scenario (Color figure online)

At this stage, we assume initial conditions as  $\hat{Y}_0 = 1$  because it corresponds to the normalized RUL  $Y \in [0, 1]$  with initial degradation error  $P_0 = 0$ .

In the update stage, composed of Lines 6–7, the update of the state vector  $\hat{x}$  and the *posteriori* error estimate matrix  $P_k$  depend on the gain  $K$  (see Line 5). We set the estimate of measurement variance  $R = \sigma_z^2$  in 0.09 because a previous heuristic evaluation adopted  $\sigma_z = 0.3$  Olivares et al. (2019). Finally, the estimated  $\hat{RUL}$  results from multiplying the prediction normalized  $\hat{Y}$  with the initial constant  $Rc$ . The benefits of using the KF are visible in Fig. 8, in which we filtered the first aircraft component's predicted RUL (blue curve). Consequently, a smooth RUL (gold curve) is obtained closer to the ground truth (red curve).

---

**Algorithm 4** Kalman Filter for  $n$  run-to-failure trajectories

---

**Require:**  $\hat{Y}$  of  $n$  run-to-failure state vectors

**Ensure:**  $\hat{RUL}$  estimated remaining life

```

1: for  $run2failureTraj = 1, 2, \dots, n$  do
2:   for  $k = 1, 2, \dots, dim(run2failureTraj)$  do
3:      $\hat{Y}_k^- = \hat{Y}_{k-1}$  //Prediction//
4:      $P_k^- = P_{k-1} + Q$  //Prediction//
5:      $K_k = \frac{P_k^-}{P_k^- + R}$  //Prediction//
6:      $\hat{Y}_k = \hat{Y}_k^- + K_k(z_k - \hat{Y}_k^-)$  //Update//
7:      $P_k = (1 - K_k) * P_k^-$  //Update//
8:   end for
9:    $\hat{RUL} = \hat{Y} * Rc$ 
10: end for
```

---

The RUL estimation took an MAE of  $20.59 \pm 2.23$  and an RMSE of  $24.39 \pm 2.11$  when 30 models (trained with different initial weights) were evaluated using the testing data split in a data-centralized scenario. This prediction performance,

documented in Table 9, is comparative with previous studies (Olivares et al., 2019; Rosero et al., 2020). Therefore, we reused the same settings to evaluate the RUL estimation in data partitioning scenarios.

### Data partitioning scenarios

In FL scenarios, we calculate the MAE and RMSE errors described in Table 9 after evaluating 30 different AI global models with the 248 trajectories of testing data split. For data partitioning purposes, whose details are described in Tables 5 and 7, we distributed 249 run-to-failure trajectories of the training and evaluating data splits among  $J$  parties using a systematic sampling criterion  $Jn$  and setting  $\alpha = J$ . This sampling criterion implies that party  $j$  gets run-to-failure trajectories of  $D$  with a step of  $J$ , starting from the  $j$ th and ending in  $n$ . Clearly,  $n$  refers to the number of trajectories of training splits. In the case of feature distribution, each party  $j$  randomly selected 13 without replacement from the 16 available input sensors. Formally, it was defined in Table 7 as a combination 13C16.

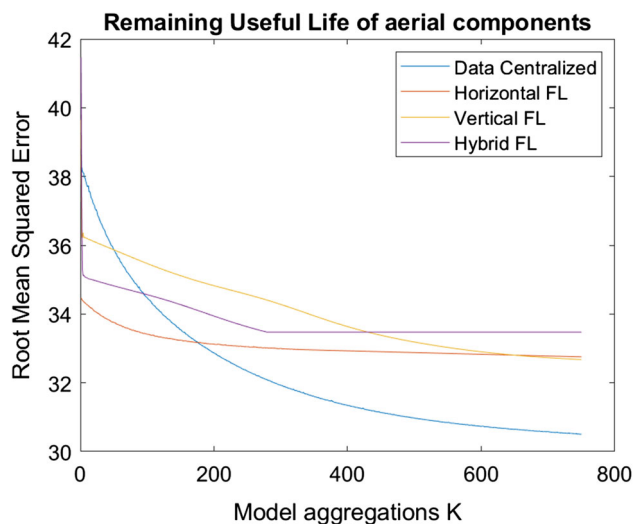
Since this feature selection strategy does not consider the presence of all the features in federated scenarios, performance losses were initially expected. On evaluating data partitioning scenarios of Table 9, in which the number of parties varies, it was easy to notice that the RUL estimation performance decreases when  $J$  increases. This phenomenon is related to the number of run-to-failure trajectories of parties. For example, when the federated applications' beneficiaries have fewer data to train local models, the problem convergence tends to be more challenging.

In Table 9, it is easy to notice that experiments with two beneficiaries demonstrated that the proposed Hybrid-FL algorithm presents performance gains compared to the Vertical and Horizontal FL algorithms. However, in the remaining experiments, when parties have a reduced number of run-to-failure trajectories, the performance losses of the Hybrid-FL algorithm are proportional to the number of parties. Figure 9 illustrates this phenomenon, where the RUL estimation performance of the global AI model (in terms of RMSE) was illustrated for all data partitioning scenarios at every  $\kappa$  model aggregation when  $J = 4$ . In this figure, FedLabSync presents performance losses. However, it learns faster than FedAvg while only sacrificing the precision a little. For instance, by considering the participation of eight parties, in which FedLabSync predicted the RUL with an MAE of  $26.72 \pm 0.64$  and FedAvg did it with  $24.8 \pm 2.61$ , the difference in estimating the remaining time is around two operating hours.

### Performance at each party

Because previous experiments showed that FedLabSync presents performance gains with few parties, we are inter-





**Fig. 9** Experimental setup performance of predictive maintenance case study at each model aggregation  $\kappa$  when  $J = 4$  (Color figure online)

ested in knowing if the performance of party models improves using our Hybrid-FL algorithm. Besides, we know beforehand from Table 9 that the performance of global AI models from the hybrid data partitioning scenario is less than the data-centralized scenario when  $J \neq 2$ . In this sense, we considered evaluating the performance gains of FedLabSync at each party when  $J = 4$ , experiments in which we noticed performance losses on the global AI model.

The MAE and RMSE errors, related to the RUL estimation of four different parties detailed in Table 10, were obtained after testing 30 models for each party and data partitioning scenario. The results of this table show that the performance of some parties improves by using the proposed Hybrid-FL algorithm (e.g., parties 1 and 2). Although the RMSE of models of some (e.g., parties 3 and 4) minimally decreased when  $J \geq 4$ , they do not sacrifice too much in the performance of the global AI model of the Hybrid-FL scenario illustrated in Fig. 9. Naturally, these performance losses may also be related to feature selection sampling errors, mainly because input sensors that better describe the degradation of aircraft components could not be selected using the 13C16 feature selection criterion.

## Conclusions and future work

The proposed Hybrid-FL algorithm, FedLabSync, has been shown to be competitive with a traditional centralized learning approach. FedLabSync executes label synchronization before training a feed-forward ANN through a parallel synchronous SGD program. In addition to label synchronization, FedLabSync offers benefits from the penalized and collabo-

rative optimization problem even when AI models differ in architecture.

FedLabSync, our proposed Hybrid-FL algorithm, operates similarly to the sample synchronization approach of Vertical FL by exchanging information and updating matrices to conduct label synchronization. Like asynchronous VFL, FedLabSync reduces data transmission overhead by minimizing a cost function through block-wise coordination that involves exchanging mini-batches of predictions. As long as the exchange of labels and local predictions between parties is end-to-end secure (e.g. through data encryption and over-the-air computation), the collaborative problem is solved, fulfilling the privacy-preserving principle of FL.

Although our experiments were carried out on a limited sample of the population selected randomly based on a feature selection criterion, we believe that our Hybrid-FL algorithm holds promise in solving collaborative problems involving hybrid data partitioning. This is not only because FedLabSync showed improved performance on party models, but also because it takes into account non-i.i.d. data distribution scenarios through the LabSync algorithm and the penalized optimization imposed by  $\sigma(\cdot)$  and  $\alpha$ .

Our empirical endeavours within authentic industrial scenarios encompassing classification and regression challenges showed that FedLabSync improves collaborative prediction performance after a few rounds of model aggregation. This phenomenon was particularly pronounced in classification tasks, wherein activation functions like softmax rounded the collaborative predictions' outputs. Besides, as our algorithm enhances the performance of the local models for several beneficiaries, we aim to design an asynchronous SGD optimization for Hybrid-FL algorithms in the near future.

**Acknowledgements** This work was partially supported by: (1) The Portuguese Foundation for Science and Technology (FCT) under the project grant FRH/BD/07344/2020, (2) The H2020 KYKLOS 4.0 Project No. 872570 and the H2020 ReMAP Project No. 769288, which the European Commission funds, and (3) The Intelligent Systems Associate Laboratory of the Center of Informatics and Systems of the University of Coimbra CISUC/LASI.

**Author contributions** Conceptualization, RLR, CS, BR and BS; methodology, RLR; software, RLR; validation, RLR, CS, BR and BS; formal analysis, CS; investigation, RLR; resources, RLR, CS, BR and BS; data curation, RLR; writing original draft preparation, RLR; writing review and editing, RLR, CS, BR, and BS; visualization, RLR; supervision, CS, BR, and BS. All authors read and approved the final manuscript.

**Funding** Open access funding provided by FCTIFCCN (b-on).

**Data availability** The data and code used in this study are available upon reasonable request to the author.

## Declarations

**Conflict of interest** The authors declare no conflict of interest.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Abdulrahman, S., Tout, H., Ould-Slimane, H., Mourad, A., Talhi, C., & Guizani, M. (2021). A survey on federated learning: The journey from centralized to distributed on-site learning and beyond. *IEEE Internet of Things Journal*, 8(7), 5476–5497. <https://doi.org/10.1109/JIOT.2020.3030072>
- Agrawal, S., Sarkar, S., Aouedi, O., Yenduri, G., Piamrat, K., Bhat-tacharya, S., Maddikunta, P. K. R., & Gadekallu, T. R. (2021). Federated learning for intrusion detection system: Concepts, challenges and future directions. [arXiv:2106.09527](https://arxiv.org/abs/2106.09527)
- Anghel, I., Cioara, T., Moldovan, D., Salomie, I., & Tomus, M. M. (2018). Prediction of manufacturing processes errors: Gradient boosted trees versus deep neural networks. In *2018 IEEE 16th International conference on embedded and ubiquitous computing (EUC)* (pp. 29–36). <https://doi.org/10.1109/EUC.2018.00012>
- Arias Chao, M., Kulkarni, C., Goebel, K., & Fink, O. (2021). Aircraft engine run-to-failure dataset under real flight conditions for prognostics and diagnostics. *Data*. <https://doi.org/10.3390/data6010005>
- Barari, A., de Sales Guerra Tsuzuki, M., Cohen, Y., & Marco, M. (2021). Editorial: Intelligent manufacturing systems towards industry 4.0 era. *Journal of Intelligent Manufacturing*, 32(7), 1793–1796. <https://doi.org/10.1007/s10845-021-01769-0>
- Carbery, C. M., Woods, R., & Marshall, A. H. (2019). A new data analytics framework emphasising preprocessing of data to generate insights into complex manufacturing systems. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 233(19–20), 6713–6726. <https://doi.org/10.1177/0954406219866867>
- Chen, T., Jin, X., Sun, Y., & Yin, W. (2020). VAFL: A method of vertical asynchronous federated learning. [arXiv:2007.06081](https://arxiv.org/abs/2007.06081)
- Cheng, K., Fan, T., Jin, Y., Liu, Y., Chen, T., Papadopoulos, D., & Yang, Q. (2019). Secureboost: A lossless federated learning framework. [arXiv:1901.08755](https://arxiv.org/abs/1901.08755)
- Dai, M., Xu, A., Huang, Q., Zhang, Z., & Lin, X. (2021). Vertical federated DNN training. *Physical Communication*, 49, 101465. <https://doi.org/10.1016/j.phycom.2021.101465>
- Deng, Y., & Mahdavi, M. (2021). Local stochastic gradient descent ascent: Convergence analysis and communication efficiency. [arXiv:2102.13152](https://arxiv.org/abs/2102.13152)
- Felbab, V., Kiss, P., & Horváth, T. (2019). Optimization in federated learning. In *CEUR workshop proceedings* (Vol. 2473, pp. 58–65).
- Gong, X., Sharma, A., Karanam, S., Wu, Z., Chen, T., Doermann, D., & Innanje, A. (2021). Ensemble attention distillation for privacy-preserving federated learning. In *2021 IEEE/CVF international conference on computer vision (ICCV)* (pp. 15056–15066). <https://doi.org/10.1109/ICCV48922.2021.01480>
- Gu, B., Xu, A., Huo, Z., Deng, C., & Huang, H. (2022). Privacy-preserving asynchronous federated learning algorithms for multi-party vertically collaborative. [arXiv:2008.06233](https://arxiv.org/abs/2008.06233)
- Hernandez, M., Vizan, A., Hidalgo, A., & Rios, J. (2006). Evaluation of techniques for manufacturing process analysis. *Journal of Intelligent Manufacturing*, 17(5), 571–583. <https://doi.org/10.1007/s10845-006-0025-1>
- Hiessl, T., Schall, D., Kemnitz, J., & Schulte, S. (2020). Industrial federated learning—requirements and system design. [arXiv:2005.06850](https://arxiv.org/abs/2005.06850)
- Hu, L., Yan, H., Li, L., Pan, Z., Liu, X., & Zhang, Z. (2021). MHAT: An efficient model-heterogenous aggregation training scheme for federated learning. *Information Sciences*, 560, 493–503. <https://doi.org/10.1016/j.ins.2021.01.046>
- Hu, Y., Niu, D., Yang, J., & Zhou, S. (2019). FDML: A collaborative machine learning framework for distributed features. In *KDD '19: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 2232–2240). Association for Computing Machinery. <https://doi.org/10.1145/3292500.3330765>
- Jiang, D., Shan, C., & Zhang, Z. (2020). Federated learning algorithm based on knowledge distillation. In *2020 International conference on artificial intelligence and computer engineering (ICAICE)* (pp. 163–167). <https://doi.org/10.1109/ICAICE51518.2020.00038>
- Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Eichner, H., El Rouayheb, S., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., ... Zhao, S. (2019). Advances and open problems in federated learning. [arXiv:1912.04977](https://arxiv.org/abs/1912.04977)
- Kallista, B., Peter, K., Brendan, M., & Ramage, D. (2022). Federated learning and privacy. *Communications of ACM*, 65(4), 90–97. <https://doi.org/10.1145/3500240>
- Khaled, A., & David, H. (2022). A dynamic mode decomposition based deep learning technique for prognostics. *Journal of Intelligent Manufacturing*. <https://doi.org/10.1007/s10845-022-01916-1>
- Kofi, N. I., Felix, A. A., Asubam, W. B., & Owusu, N.-B. (2022). Applications of artificial intelligence in engineering and manufacturing: a systematic review. *Journal of Intelligent Manufacturing*, 33(6), 1581–1601. <https://doi.org/10.1007/s10845-021-01771-6>
- Konexny, J., McMahan, H. B., Ramage, D., & Richtárik, P. (2016). Federated optimization: Distributed machine learning for on-device intelligence. [arXiv:1610.02527](https://arxiv.org/abs/1610.02527)
- Li, Q., Wen, Z., & He, B. (2019). Practical federated gradient boosting decision trees. [arXiv:1911.04206](https://arxiv.org/abs/1911.04206)
- Li, Q., Wen, Z., Wu, Z., Hu, S., Wang, N., Liu, X., & He, B. (2019). A survey on federated learning systems: Vision, hype and reality for data privacy and protection. [arXiv:1907.09693](https://arxiv.org/abs/1907.09693)
- Lin, T., Jin, C., & Jordan, M. I. (2019). On gradient descent ascent for nonconvex-concave minimax problems. [arXiv:1906.00331](https://arxiv.org/abs/1906.00331)
- Liu, W., Chen, L., Chen, Y., & Zhang, W. (2019). Accelerating federated learning via momentum gradient descent. [arXiv:1910.03197](https://arxiv.org/abs/1910.03197)
- Liu, Y., Chen, T., & Yang, Q. (2018). Secure federated transfer learning. [arXiv:1812.03337](https://arxiv.org/abs/1812.03337)
- Liu, Y., Kang, Y., Zhang, X., Li, L., Cheng, Y., Chen, T., Hong, M., & Yang, Q. (2019). A communication efficient vertical federated learning framework. [arXiv:1912.11187](https://arxiv.org/abs/1912.11187)
- Luis, B., Paloma, B., Xavier, O., & Floris, F. (2021). Aircraft fleet health monitoring with anomaly detection techniques. *Aerospace*. <https://doi.org/10.3390/aerospace8040103>
- McMahan, H. B., Moore, E., Ramage, D., & Arcas, B. A. (2016). Federated learning of deep networks using model averaging. [arXiv:1602.05629](https://arxiv.org/abs/1602.05629)
- Moldovan, D., Anghel, I., Cioara, T., & Salomie, I. (2019). Time series features extraction versus lstm for manufacturing processes performance prediction. In *2019 International conference on speech*

- technology and human–computer dialogue (*SpeD*) (pp. 1–10). <https://doi.org/10.1109/SPED.2019.8906653>
- Mora, A., Tenison, I., Bellavista, P., & Rish, I. (2022). Knowledge distillation for federated learning: A practical guide. [arXiv:2211.04742](https://arxiv.org/abs/2211.04742)
- Nguyen, D. C., Ding, M., Pathirana, P. N., Seneviratne, A., Li, J., Niyato, D., & Poor, H. V. (2021). Federated learning for industrial Internet of Things in future industries. *IEEE Wireless Communications*, 28(6), 192–199. <https://doi.org/10.1109/MWC.001.2100102>
- Ning, G., Guanghao, L., Li, Z., & Yi, L. (2022). Failure prediction in production line based on federated learning: An empirical study. *Journal of Intelligent Manufacturing*, 33(8), 2277–2294. <https://doi.org/10.1007/s10845-021-01775-2>
- Novikova, E., Doynikova, E., & Golubev, S. (2022). Federated learning for intrusion detection in the critical infrastructures: Vertically partitioned data use case. *Algorithms*, 15(4), 104. <https://doi.org/10.3390/a15040104>
- Olivares, A., Gonzalez, A., Tovar, S. T., & Gorrostieta, E. (2019). Remaining useful life prediction for turbofan based on a multilayer perceptron and Kalman filter. In *2019 16th International conference on electrical engineering, computing science and automatic control—CCE*. <https://doi.org/10.1109/ICEEE.2019.8884495>
- Peng, J., Andreas, K., Wang, D., Zhibin, N., Fan, Z., Wang, J., Xiufeng, L., & Jivka, O. (2022). A systematic review of data-driven approaches to fault diagnosis and early warning. *Journal of Intelligent Manufacturing*. <https://doi.org/10.1007/s10845-022-02020-0>
- Pham, Q. V., Dev, K., Maddikunta, P. K. R., Gadekallu, T. R., & Huynh-The, T. (2021). Fusion of federated learning and industrial Internet of Things: A survey. [arXiv:2101.00798](https://arxiv.org/abs/2101.00798)
- Reddi, S. J., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S., & McMahan, H. B. (2020). Adaptive federated optimization. *CoRR*. [arXiv:2003.00295](https://arxiv.org/abs/2003.00295)
- Rosero, R. L., Silva, C., & Ribeiro, B. (2020). Remaining useful life estimation in aircraft components with federated learning. *International Journal of Prognostics and Health Management*. <https://doi.org/10.36001/phme.2020.v5i1.1228>
- Rosero, R. L., Silva, C., & Ribeiro, B. (2022). Remaining useful life estimation of cooling units via time–frequency health indicators with machine learning. *Aerospace*. <https://doi.org/10.3390/aerospace9060309>
- Sahu, A. K., Li, T., Sanjabi, M., Zaheer, M., Talwalkar, A., & Smith, V. (2018). Federated optimization in heterogeneous networks. [arXiv:1812.06127](https://arxiv.org/abs/1812.06127)
- Sahu, A. K., Li, T., Sanjabi, M., Zaherr, M., Talwalkar, A., & Smith, V. (2019). On the convergence of federated optimization in heterogeneous networks. [arXiv:1812.06127](https://arxiv.org/abs/1812.06127)
- San, K. T., & Young, S. S. (2021). Multitask learning for health condition identification and remaining useful life prediction: Deep convolutional neural network approach. *Journal of Intelligent Manufacturing*, 32(8), 2169–2179. <https://doi.org/10.1007/s10845-020-01630-w>
- Saxena, A., & Goebel, K. (2008a). PHM08 Challenge Data Set. Technical Report, NASA Prognostics Data Repository, NASA Ames Research Center, Moffett Field.
- Saxena, A., & Goebel, K. (2008b). Turbofan engine degradation simulation. Technical report, NASA Ames Prognostics Data Repository, NASA Ames Research Center, Moffett Field.
- Scott, M. J., Verhagen, W. J. C., Bieber, M. T., & Marzocca, P. (2022). A systematic literature review of predictive maintenance for defence fixed-wing aircraft sustainment and operations. *Sensors*. <https://doi.org/10.3390/s22187070>
- Sebbouh, O., Cuturi, M., & Peyré, G. (2021). Randomized stochastic gradient descent ascent. [arXiv:2111.13162](https://arxiv.org/abs/2111.13162)
- Smith, V., Chiang, C.-K., Sanjabi, M., & Talwalkar, A. (2017). Federated multi-task learning. [arXiv:1705.10467](https://arxiv.org/abs/1705.10467)
- Su, L., & Lau, V. K. N. (2021). Hierarchical federated learning for hybrid data partitioning across multitype sensors. *IEEE Internet of Things Journal*, 8(13), 10922–10939. <https://doi.org/10.1109/JIOT.2021.3051382>
- Tran-Dinh, Q., & Zhu, Y. (2019) Non-stationary first-order primal-dual algorithms with faster convergence rates. [arXiv:1903.05282](https://arxiv.org/abs/1903.05282)
- Wang, H., Yurochkin, M., Sun, Y., Papailiopoulos, D. S., & Khazaeni, Y. (2020) Federated learning with matched averaging. [arXiv:2002.06440](https://arxiv.org/abs/2002.06440)
- Yang, L., Yan, K., Xinwei, Z., Liping, L., Yong, C., Tianjian, C., Mingyi, H., & Qiang, Y. (2019). A communication efficient vertical federated learning framework. [arXiv:1912.11187](https://arxiv.org/abs/1912.11187)
- Yao, H., Xiaoyan, S., Yang, C., & Zishuai, L. (2019). Model and feature aggregation based federated learning for multi-sensor time series trend following. *Advances in Computational Intelligence*. [https://doi.org/10.1007/978-3-030-20521-8\\_20](https://doi.org/10.1007/978-3-030-20521-8_20)
- Yu, M., Qianhui, W., Xiu, L., & Bqing, H. (2021). Remaining useful life estimation via transformer encoder enhanced by a gated convolutional unit. *Journal of Intelligent Manufacturing*, 32(7), 1997–2006. <https://doi.org/10.1007/s10845-021-01750-x>
- Zhang, D., Xu, B., & Wood, J. (2016). Predict failures in production lines: A two-stage approach with clustering and supervised learning. In *2016 IEEE International conference on Big Data (Big Data)* (pp. 2070–2074). <https://doi.org/10.1109/BigData.2016.7840832>
- Zhang, X., Yin, W., Hong, M., & Chen, T. (2020). Hybrid federated learning: Algorithms and implementation. [arXiv:2012.12420](https://arxiv.org/abs/2012.12420)
- Zhenyu, L., Donghao, Z., Weiqiang, J., Xianke, L., & Hui, L. (2020). An adversarial bidirectional serial-parallel LSTM-based qtd framework for product quality prediction. *Journal of Intelligent Manufacturing*, 31(56), 1511–1529. <https://doi.org/10.1007/s10845-019-01530-8>

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.