Journal of Parallel and Distributed Computing 167 (2022) 64–76

Contents lists available at ScienceDirect

Journal of Parallel and Distributed Computing

www.elsevier.com/locate/jpdc

Cohort-based federated learning services for industrial collaboration on the edge

Thomas Hiessl[a,c,*], Safoura Rezapour Lakani[a], Jana Kemnitz[a], Daniel Schall[a], Stefan Schulte[b]

[a] Siemens Technology, Siemensstraße 90, Vienna, 1210, Austria
[b] Christian Doppler Laboratory for Blockchain Technologies for the Internet of Things, Hamburg University of Technology, Blohmstrasse 15, Hamburg, 10439, Germany
[c] Vienna University of Technology, Karlsplatz 13, Vienna, 1040, Austria

ARTICLE INFO

Article history:
Received 1 September 2021
Received in revised form 18 February 2022
Accepted 23 April 2022
Available online 2 May 2022

Keywords:
Federated learning
Edge computing
Collaborative AI
Industrial collaboration
Service-based architecture

ABSTRACT

Machine Learning (ML) is increasingly applied in industrial manufacturing, but often performance is limited due to insufficient training data. While ML models can benefit from collaboration, due to privacy concerns, individual manufacturers often cannot share data directly. Federated Learning (FL) enables collaborative training of ML models without revealing raw data. However, current FL approaches fail to take the characteristics and requirements of industrial clients into account.

In this work, we propose an FL system consisting of a process description and a software architecture to provide FL as a Service (FLaaS) to industrial clients deployed to edge devices. Our approach deals with skewed data by organizing clients into cohorts with similar data distributions. We evaluated the system on two industrial datasets. We show how the FLaaS approach provides FL to client processes by considering their requests submitted to the Industrial Federated Learning (IFL) Services API. Experiments on both industrial datasets and different FL algorithms show that the proposed cohort building can increase the ML model performance notably.

1. Introduction

In recent years, Machine Learning (ML) has improved industrial manufacturing and process automation significantly, e.g., in fault classification, quality estimation, and soft sensing [13]. For instance, value-added and ML-based services like condition monitoring for production machines can be used to facilitate timely and cost-efficient maintenance actions throughout the lifetime of a machine [7,1].

To achieve these benefits, high-quality ML models require a significant amount of labeled training and test data. This data is often considered privacy-sensitive and needs to be protected from outside parties [36]. Since these datasets cannot be shared with centralized servers for ML, a privacy-preserving way of knowledge sharing between collaborating devices is desired.

For this, Federated Learning (FL) addresses collaborative and privacy-preserving on-device training, as introduced by McMahan et al. [26]. FL is a recently emerged approach for transferring knowledge as model parameters (e.g., weights of neural networks) between edge devices without revealing raw data. For this, models are trained locally on edge devices and are then uploaded to an aggregation server that fuses model parameters, e.g., by averaging the models. After aggregation, the model is returned to the clients for evaluation. This process is executed repeatedly either until a pre-defined number of communication rounds or a provided level of quality (e.g., classification accuracy) is reached.

FL can be carried out at the edge, which is especially beneficial for industrial scenarios, because of the lower latency and communication costs for bandwidth-intensive applications [38].

To optimize collective model training, various FL algorithms, e.g., [26,39,30,40], have been researched focusing on model aggregation and client selection. Applying this concept for industrial machines that are distributed over multiple factories and facing heterogeneous environmental and operational conditions is referred to as Industrial Federated Learning (IFL) [16].

At this time, still many challenges [18] exist in FL which apply especially to industrial clients [16]. First, FL training processes typically start with selecting clients that are sampled by a server acting as central authority [26,18]. Hence, single servers are used for orchestrating and monitoring the distributed training process managing connected edge devices [27]. So, in the majority of FL

[*] Corresponding author.
E-mail address: hiessl.thomas@siemens.com (T. Hiessl).

https://doi.org/10.1016/j.jpdc.2022.04.021
0743-7315/© 2022 Elsevier Inc. All rights reserved.

approaches the central authority defines the learning task by deciding, e.g., on the used ML model, hyperparameters, and the FL algorithm [18]. However, in industrial applications, users (e.g., machine operators in production lines) have individual requirements regarding business partnerships when it comes to collaborations with other companies on improving and maintaining machine performance [16]. Hence, clients dependent on the learning task definition provided by the server are restricted to centrally managed ML models, therefore applications, and have no influence on selecting partners for FL. For example, pump manufacturers provide their products to customers (clients) from various industries like manufacturing, food and beverage, pharmaceutical engineering or water supply. Different use cases and therefore operational conditions are present in these domains, which may require adaptations of used ML models and hyperparameters for groups of clients. Moreover, even clients from the same field have restrictions to collaborate only with selected partners or with at least a defined minimum number of partners to increase the chance for actual ML model improvements [10].

Hence, there is a need for a service-based system, empowering independent clients to create and submit ML models to the server and thereby enable FL as a Service (FLaaS) [23]. Using a FLaaS approach allows a group of collaborating and independent clients to apply FL on these models and subsequently use it on their machines. Furthermore, considering client restrictions for collaboration partners and the applied FL algorithm on the server provides flexibility to the clients.

Second, model updates for operating machines often need to be triggered explicitly in industry under human supervision. This can be relevant in industry when ML models are used to assist human users [24]. Potential use cases are, e.g., condition classification of factory machines, failure detection, or even optimization of production processes. To update the used ML model with FL, a client may want to explicitly request participation in FL rounds, instead of automatically getting invoked in periodical execution plans by the server. This enables manual testing before deciding whether to use the model in production. Although explicit participation in FL is not unique to IFL, this is significant for industrial applications.

Third, a prominent challenge in FL is the problem of heterogeneous data distributions [18], which is especially present in industrial domains when machines operate with different configurations under varying environmental and operational conditions. For instance, considering different liquids that are pumped in industrial processes, one can observe different error cases occurring over time detected by models using vibration patterns as input data [41]. Typically, one can observe that input data (e.g., vibration data) is varying across clients which is referred to as *feature distribution skew* [17]. Additionally, varying labels (e.g., error cases) can often be observed as well, which is referred to as *label distribution skew*. This phenomenon corresponds to the non-IID (identically and independently distributed) data problem, which is a general issue in FL [18]. In non-IID settings, poor model quality can be observed by individual clients as the model is validated on their local data after FL has been applied for all clients [18]. Therefore, FL systems need to provide mitigation strategies in implemented FL algorithms.

This paper addresses the aforementioned challenges of (i) enabling clients to individually and independently select the used ML models and to define client criteria for collaboration in FL, (ii) enabling clients to explicitly participate in FL on an on-demand basis, and (iii) non-IID data distributions by proposing an FL system. Notably, the contributions are motivated and evaluated using scenarios from the industrial domain, but as discussed above, similar problems also occur in FL in general.

The contributions of this work can be summarized as follows:

Table 1
IFL system entities.

Notation	Description
C	Set of clients $c_i \in C$ participating in IFL
A	Set of assets $a \in A$
M	Set of ML models $m \in M$
$CRIT$	Set of federation criteria $crit \in CRIT$
CB	Set of cohort building approaches $cb \in CB$
ALG	Set of FL algorithms $alg \in ALG$
T	Set of tasks $t_i \in T$ submitted by clients
P	Set of populations $p \in P$
COH^p	Set of cohorts $coh \in COH^p$ of a given population p

- Presentation of an FL process, the *IFL process*, for industrial clients including two algorithms dealing with individual and on-demand requests, and non-IID data.
- Design and implementation of a service-based system, the *IFL system*, covering the IFL process and providing FLaaS.
- Evaluation of the IFL system using two time series-based industrial datasets, providing several physical clients (machines) and derived virtual ones. The results on the model performance after FL are presented and compared in IID and non-IID scenarios.

We show that the IFL system considers on-demand participation of clients and yields significant improvements in classification accuracy applying FL on cohorts of similar clients rather than on the overall population.

The remainder of this paper is organized as follows: We describe the IFL system design in Section 2. In Section 3, we demonstrate the results of the FL experiments, and we review related work in Section 4. We conclude and outline future work in Section 5.

2. System design

In order to present the design of our system, we first introduce the basic notation used in our work in Section 2.1. In Section 2.2, we introduce the IFL process and the algorithms, which are central to our approach of applying FL on cohorts of similar clients. The architecture of the implemented service-based system is described in Section 2.3.

2.1. Basic notation

To describe our IFL system model formally, we introduce the notation presented in Table 1. For this, we consider that a client $c \in C$ manages an asset $a \in A$ (e.g., a concrete heating pump) of a given asset type $type(a)$ (e.g., a centrifugal pump). Every asset type defines a data scheme $u(type(a))$ that needs to match with the data scheme $v(m)$ required by the ML model $m \in M$ that client c wants to train.

To participate in IFL, the i-th client c_i submits a task t_i to the IFL Server. For this, the client needs to specify asset a^{t_i}, model m^{t_i}, the individually selected FL algorithm $alg^{t_i} \in ALG$ for aggregating model weights, a cohort building algorithm $cb^{t_i} \in CB$, and federation criteria $crit^{t_i} \in CRIT$ before submitting t_i to the server.

The federation criteria $crit^{t_i}$ correspond to a set of requirements, where each need to be fulfilled by the system to consider t_i for upcoming FL rounds. So, we consider $t_i \in T$ with $T \subseteq A \times M \times CRIT \times CB \times ALG$.

A population p is a set of tasks that refer to the same asset type $type^p$, model m^p, FL algorithm alg^p and cohort building approach cb^p, where $p \subseteq T$ with $type^p = type(a^{t_i})$, $m^p = m^{t_i}$, $alg^p = alg^{t_i}$, and $cb^p = cb^{t_i}$ holds for all $t_i \in p$.

A cohort coh is a set of tasks with similar data distributions with $coh \subseteq p$. For this, the cohort building approach cb^p is used to

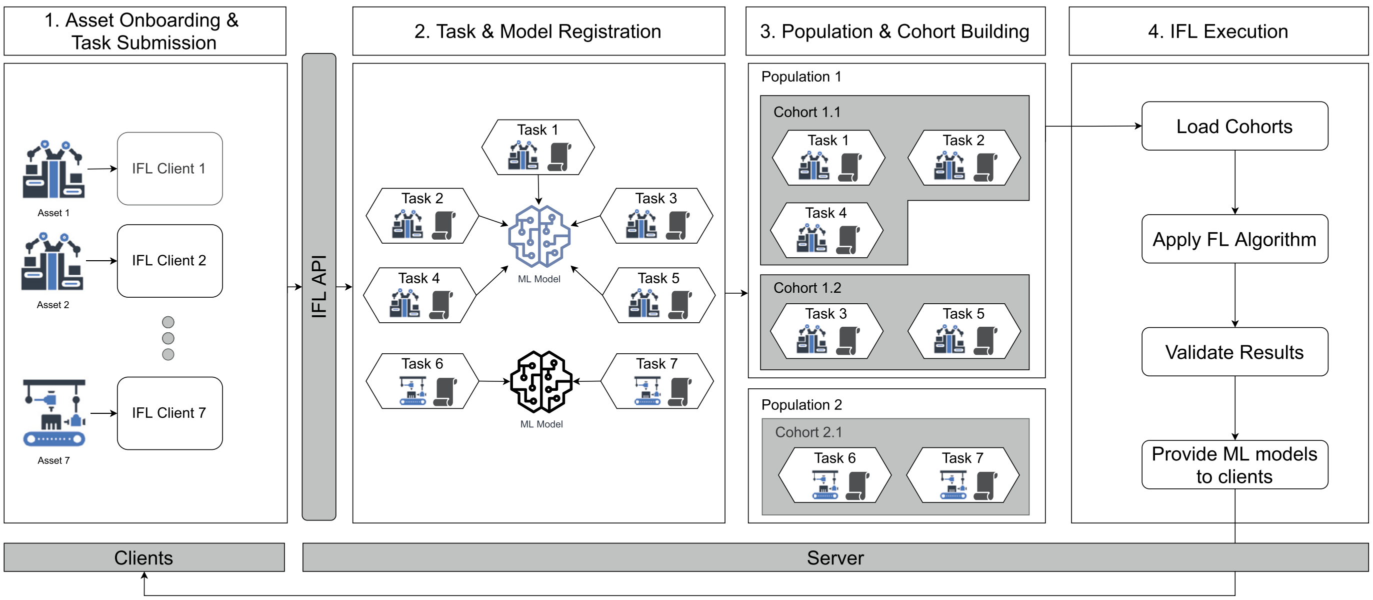


Fig. 1. IFL Process with 4 phases: 1. Clients are connected to their assets and submit tasks using the IFL API to participate in FL. 2. Submitted tasks are registered on the server referring to ML models used as base for FL. 3. Populations of tasks with same asset types are created. Cohorts further split populations into clusters of tasks with similar data distributions. 4. FL is executed for each cohort by applying the algorithm selected by the clients. Finally, validating results and providing the ML model to clients.

assign every task within a population to a cohort. Furthermore, we consider COH^p as the set of all cohorts of population p. Hence, for $coh_1^p, \ldots, coh_{|COH^p|}^p \in COH^p$ it holds that $coh_j^p \subseteq p$ for all $j \in \{1, \ldots, |COH^p|\}$ with $coh_j^p \cap coh_k^p = \{\}$ and $j \neq k$.

Finally, FL is applied on population p using the FL algorithm alg^p to train one model per cohort $coh_j \in COH^p$. For this, we consider m^{coh_j} with $m^{coh_j} = alg^p(coh_j)$ for all $coh_j \in COH^p$.

2.2. IFL process

In our solution, we address the discussed challenges and propose the *IFL process* depicted in Fig. 1. The process is executed by the IFL system consisting of the *IFL Client* and the *IFL Services*. The client can be deployed on edge devices to train and operate ML models based on data generated by connected machines. The IFL Services offer an API to the clients providing knowledge aggregation and distribution on a central server.

To support FL for edge-based industrial clients, the IFL Client and services provide a four-step process depicted in Fig. 1. As a prerequisite, we consider deployed client applications that invoke the IFL Client to establish connectivity to the IFL Services. Data is recorded from the asset and stored on the device. For this, we assume a classification problem with input data provided as matrix $X \subseteq R^{N_S \times N_V}$ with N_V variables and N_S samples, and a N_S-dimensional target vector $y \subseteq R^{N_S}$.

2.2.1. Asset onboarding and task submission

The first step involves the IFL Client that needs to specify metadata that is referenced in a task. This metadata contains the used asset with the corresponding asset type, whereas the asset type can be reused, if other clients have already published this to the server. Similarly, an ML model is created or selected from the server to be applied to the data. This model is created upfront, based on the asset types' corresponding data structure and the ML task that needs to be solved.

Based on that, the client selects a cohort building approach. For this, the IFL Services provide two approaches. The first approach applies a cluster algorithm based on input data X to address potential feature distribution skew. The second approach clusters based on target data y to consider label distribution skew. In both cases, the respective cohort building approach is selected to reduce skewness within cohorts and to improve performance of models that are trained in a cohort by applying FL.

Furthermore, clients specify the knowledge aggregation algorithm, e.g., Federated Averaging [26], and individual federation criteria, e.g., the minimum number of clients in a population or the minimum dataset size. This enables the client to control the knowledge aggregation process, e.g., to avoid that knowledge is transferred only between a small number of clients, which may lead to insufficient training results. Furthermore, this prevents that knowledge from an individual model is transferred to only a few other clients that may not contribute to the global model. To participate in IFL, clients then submit a task with the mentioned specifications to the server using the IFL API.

2.2.2. Task and model registration

In the second step, the server stores the defined assets, uploaded ML models and the tasks. Subsequently, the server verifies that the data scheme of the asset fits to the data scheme required by the referenced ML model, i.e., $u(type(a)) = v(m)$. Hence, it is ensured that clients can participate in FL rounds when the model is trained on their local data.

2.2.3. Population and cohort building

The third step assigns tasks to populations and further splits populations into cohorts. This facilitates FL to be executed within small groups of similar clients. For this, we consider the server to execute Algorithm 1 to assign tasks accordingly as they are submitted by clients. First, to build populations, we consider tasks with equal configurations. Particularly, we search for a population p with the same asset type, ML model, cohort building algorithm and FL algorithm as referred to in task t_i. This is necessary to consider a valid FL setting, whereas the same algorithms and model need to be applied on a common data scheme. If any population matches the task configuration as checked in line 3, the task is added. Otherwise, a new population p_{new} is created and added to the populations store P on the server (lines 13 and 14).

Furthermore, it is required to reach a consensus regarding the federation criteria, i.e., all criteria $crit^{t_i}$ have to hold for all tasks

Algorithm 1 `PopulationCohortBuilding`.

```
Input: new task $t_i$ received from client, populations $P$ stored on the server
    Update populations:
 1: $added_{t_i} \leftarrow False$
 2: for $p \in P$ do
 3:     if $type^p = type(a^{t_i})$, $m^p = m^{t_i}$, $alg^p = alg^{t_i}$, and $cb^p = cb^{t_i}$ then
 4:         $p \leftarrow p \cup t_i$
 5:         $added_{t_i} \leftarrow True$
 6:         if $crit^{t_i}$ holds for every $t_i \in p$ then
 7:             $COH^p \leftarrow CreateCohorts(p)$
 8:             break
 9:         end if
10:     end if
11: end for
12: if not $added_{t_i}$ then
13:     $p_{new} \leftarrow \{t_i\}$
14:     $P \leftarrow P \cup p_{new}$
15: end if

    CreateCohorts(p):
16: Initialize $F$ as $n \times m$ matrix for $n$ tasks and $m$ features
17: for $t_i \in p$ do
18:     if $cb^p$ = Target Distribution then
19:         $r \leftarrow \{mean(y^{t_i}), var(y^{t_i}), skew(y^{t_i}), kurt(y^{t_i})\}$
20:     end if
21:     if $cb^p$ = Input Distribution then
22:         $r \leftarrow \{mean(X^{t_i}), var(X^{t_i}), skew(X^{t_i}), kurt(X^{t_i})\}$
23:     end if
24:     add row $r$ to $F$
25: end for
26: for feature $f \in F$ do
27:     if $std(f) < \epsilon$ then
28:         remove $f$ from $F$
29:     end if
30: end for
31: $k \leftarrow elbow(F)$
32: $COH^p \leftarrow kMeans(F, k)$
33: return $COH^p$
```

$t_i \in p$ as verified in line 6. For instance, we consider a criterion for requiring a minimum number of tasks in a population as a precondition before starting FL. To formally describe this exemplary federation criterion, let $q(t_i)$ be a function with $q(t_i) < |p|$ for all $t_i \in p$, where $q(t_i)$ returns the minimum number of tasks that is required by t_i. Based on that, the server eventually starts cohort creation in line 7 to improve model accuracy when FL is executed.

For this, we consider that data of individual clients can be non-IID. As we identified in [16], non-IID data can be observed when assets operate in heterogeneous industrial environments. Hence, in `CreateCohorts(p)`, the server makes use of aggregated data that describe the clients data distribution.

For this, we consider two cohort building approaches, i.e., *Target Distribution* and *Input Distribution*. Applying the former one, the server requests the client to compute the statistical moments mean, variance, skewness and kurtosis of the target data y^{t_i} in line 19. These measures provide information on the shape of the data's underlying distribution function, i.e., location (mean), dispersion (variance), asymmetry (skewness) and the form of tails (kurtosis). We used these features to capture the target distribution of every client's dataset as precise as possible, which is the basis for accurately assigning the corresponding task to a cohort of tasks with similar target distributions. Therefore, we can reduce label distribution skew in a cohort. In line 24, the features are returned to the server and added as a new row to the feature matrix F.

The *Input Distribution* approach computes the mentioned moments based on all n variables of the input matrix X^{t_i} (line 22) and adds them to F. This feature matrix F consists of $|p|$ rows and u columns, where u is the number of features. Hence, using *Target Distribution* we have $u = 4$, while for *Input Distribution* we consider $u = 4 \times n$. Analogous to *Target Distribution*, using *Input Distribution* we can reduce feature distribution skew.

Since F may contain many features with limited information, we remove all features f from F with $std(f) \le \epsilon$ in line 28, where $std(f)$ computes the standard deviation and ϵ is a pre-configured parameter on the server.

To eventually create the cohorts COH^p, we apply the *k-means* [19] cluster algorithm based on the reduced feature matrix F. The *k-means* cluster algorithm is an iterative approach that considers a fixed number of k clusters, whereas data points are assigned to cluster centers that minimize variance within clusters. Optimally, we want clusters where all the data in a cluster are close to each other, and the distance between two clusters is as large as possible. To identify k, we apply the *elbow* method [37,33] in line 31 to find the optimal value with respect to the *silhouette* [32] score. For this, the elbow method iterates over increasing values of k applying k-means and stopping when improvements of the silhouette score are no longer worth further computation. The silhouette score is maximized if samples (= clients) have a relatively short distance to other samples in the assigned cluster and a relatively large distance to samples from other clusters. In line 32, we finally apply k-means with the identified k to assign all tasks $t_i \in p$ to cohorts COH^p. We used the combination of k-means and the elbow method since it can automatically be applied by the IFL Service without the need for human validation of the number of built cohorts and still avoiding under- and over-fitting of the trained cluster model.

2.2.4. IFL Execution

Algorithm 2 `IFL Execution`.

```
Input: $COH^p$ received from population and cohort building step
    Server execution:
 1: for $coh_j \in COH^p$ do
 2:     initialize $m_0^{t_i}$ for all tasks $t_i \in coh_j$
 3:     for round $r = 1, ..., R$ do
 4:         if $alg^p$ = FedAvg then
 5:             for task $t_i \in coh_j$ do
 6:                 $m_{r+1}^{t_i} \leftarrow ClientUpdate(t_i, m_r^{t_i})$
 7:             end for
 8:             $m_{r+1}^{coh_j} \leftarrow \frac{1}{|coh_j|} \sum_{i=1}^{|coh_j|} m_{r+1}^{t_i}$
 9:         end if
10:         if $alg^p$ = SeqFL then
11:             for task $t_i \in coh_j$ do
12:                 $m_r^{t_{i+1}} \leftarrow ClientUpdate(t_i, m_r^{t_i})$
13:             end for
14:             $m_{r+1}^{coh_j} \leftarrow m_{r+1}^{t_0} \leftarrow m_r^{t_{|coh_j|}}$
15:         end if
16:     end for
17:     validate and send $m_R^{coh_j}$ to all clients $c_i$ of tasks $t_i \in coh_j$
18: end for

    ClientUpdate(t,m) // Run task $t$ on client
19: $B \leftarrow$ ($X^t$ split into batches of size $B$)
20: for epoch $e \in 1..E$ do
21:     for batch $b \in B$ do
22:         $m \leftarrow m - \alpha \nabla l(m, b)$
23:     end for
24: end for
25: return $m$ to server
```

The fourth and final step in the IFL process applies the selected FL algorithm alg^p on all created cohorts COH^p as described in Algorithm 2. To start the execution, the server loops through all $coh \in COH^p$ and initializes the models $m_0^{t_i}$ for all tasks $t_i \in coh_j$ in line 2. For this, the model architecture of the underlying neural network is created by building all layers as defined in the base model m^p of the population.

To actually share knowledge between the involved clients, FL algorithms can be invoked by the server. In this work, we evaluate the integration of two FL algorithms in the IFL Services to be applied on cohorts, i.e., *FedAvg* [26] and *Sequential FL* (*SeqFL*) [8]. We

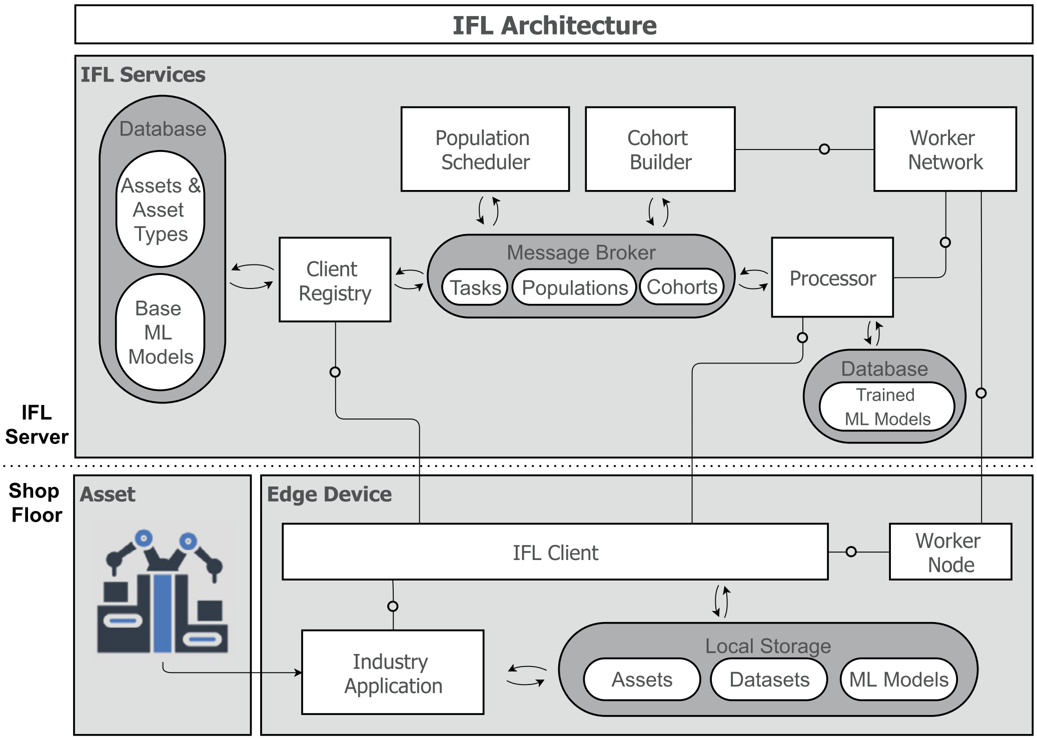


Fig. 2. IFL Architecture: edge-based IFL Client and IFL Services.

selected the two algorithms since they are well-established and often considered as a benchmark for FL.

Both algorithms trigger client training by iterating over tasks t_i of the processed cohort coh_j, and calling `ClientUpdate(t,m)` (lines 6 and 12). In this function (lines 19-25), the dataset X^t is split into batches B and is iterated several times as defined in the number of epochs E. In line 22, the model is updated using one step of gradient descent optimization, considering a loss function l and a learning rate *alpha*. After iterating the defined number of epochs, the optimized model is returned to the server in line 25, to exchange the gained knowledge with other clients.

In FedAvg, this is achieved by summing up model weights from all clients of a given round and dividing it by the number of tasks in the cohort $|coh_j|$ as presented in line 8. In our approach, we integrated a simplified equally weighted averaging, whereas in the original FedAvg approach client models are weighted with an additional factor expressing the proportion of the number of examples N_{S_i} of client c_i with respect to the total number of examples. Using our approach does not reveal this information of the dataset to the server. In the next round, the aggregated model is again distributed to all clients.

In SeqFL, the model is passed sequentially from one client to another, whereas the subsequent client optimizes the model that the previous client has optimized before. The result of the last client in a given round r is used as input for the first client of round $r+1$ (line 14).

After training using either of the aforementioned algorithms, the resulting model $m_R^{coh_j}$ is validated by involved clients on their local test datasets as stated in line 17. This yields validation metrics (i.e., test set accuracies) that are passed to the clients along with the model, which concludes the IFL execution process. As a result, the client is enabled to decide whether to operate the IFL-based model or an individually trained model on the edge device.

2.3. System architecture

To run the IFL process described in Section 2.2, the IFL system provides the service-based architecture depicted in Fig. 2. In this FLaaS approach, we consider two main locations, i.e., the shop floor and the IFL Server.

On the shop floor, assets are operated and generated data (e.g., measured vibrations) are sent to an *Edge Device* to train an ML model that can be used for, e.g., condition monitoring. The IFL Server consists of the *IFL Services* that provide an interface to onboard assets, submit tasks, and apply FL on cohorts of clients. This architecture allows multiple clients from different locations (i.e., shop floors) to use the IFL Services and participate in FL.

2.3.1. IFL Client on Edge Devices

To support the asset onboarding and task submission step in the IFL process as described in Section 2.2.1, the *IFL Client* is deployed to Edge Devices and invoked by the *Industry Application*. The Industry Application can include arbitrary business logic that makes use of FLaaS to train ML models on recorded asset data. To support this, the IFL Client connects to the IFL Services and creates an IFL task. The corresponding metadata (e.g., asset, ML model) is stored along with the used dataset on a local data storage to ensure transparency on the training history.

After submitting a task, the IFL Client starts a local *Worker Node* that can run, e.g., the `ClientUpdate(t,m)` function (see lines 19ff in Algorithm 2) to be invoked by the server.

For this, the IFL Client provides the training and test dataset to the Worker Node to optimize the model m in the respective communication round. As a prerequisite, the Worker Node needs to register at the Worker Network that is located on the IFL Server. This enables that Worker Nodes can be found, as a lookup on the server is initiated to invoke clients. Worker Nodes are relevant in this architecture, since they enable isolated training per task. This training can even be outsourced to trusted Edge Devices

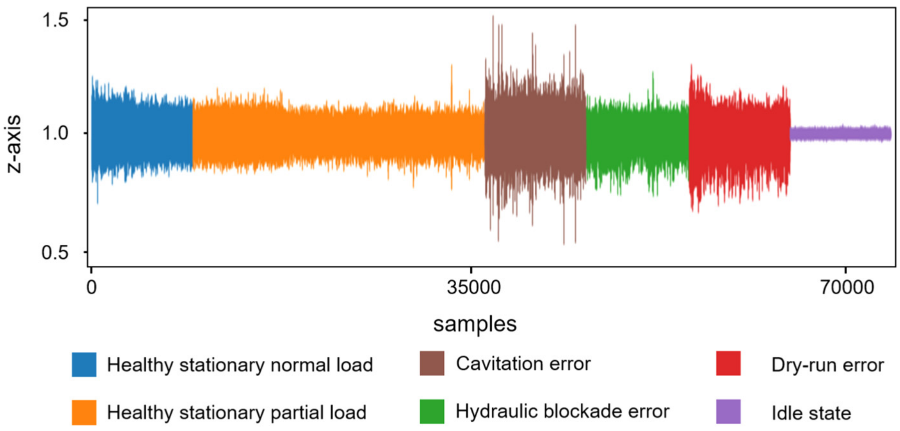


Fig. 3. Pump classification dataset: a schematic view of samples from one sensor.

by spawning Worker Nodes on remote locations if local resources (e.g., memory) are already fully utilized.

2.3.2. IFL Services

To provide FLaaS, the IFL process is supported on the server side, which considers *Client Registry*, *Population Scheduler*, *Cohort Builder*, and the *Processor* as independent services. We further consider a database for assets, associated asset types, and ML models. The latter are considered as base ML models, reflecting a deep learning neural network architecture with untrained weights.

Using the Client Registry, the ML models can be created by clients and used by other clients by referring to them in the submitted tasks. For this, the Client Registry provides an API that accepts assets, asset types, ML models and tasks. After data scheme validation (see Section 2.2.2), the task is forwarded to the *Message Broker*. This broker supports a publish/subscribe messaging architecture to share processed output (i.e., validated tasks, built populations, and cohorts) between the services. This enables loosely-coupled services, whose instances can be scaled up and down considering varying loads of task submissions.

To support the population and cohort building as addressed in Section 2.2.3, the population scheduler consumes tasks from the Message Broker and assigns them to a population. If provided federation criteria hold for all clients, the population is published to the Message Broker for cohort building.

The Cohort Builder queries the Worker Network for registered Worker Nodes of clients that have submitted tasks to the currently processed population. The query mechanism is useful since the selective approach does not invoke clients of other populations blocking their resources. Next, the client statistics are retrieved from the Worker Nodes, to build cohorts (see Algorithm 1).

To address the IFL Execution process step, as described in Section 2.2.4, the Processor service subscribes to created cohorts on the Message Broker and applies the selected FL algorithm. For this, the Processor sends the ML model to respective Worker Nodes and retrieves the updated model after training. After the last communication round, the trained model is validated on test data by the Worker Nodes. To deploy the model to the shop floor, participating clients can download the model as provided by the Processor API.

3. Evaluation

In this section, we show the applicability of the IFL process on different datasets including industrial data. We evaluate our IFL system approach on IID and non-IID datasets.

In the following, we explain our evaluation setup (Section 3.1). The characteristics of the datasets used for our evaluation and the scenarios designed from these datasets are described in Section 3.2. Our experimental design is explained in Section 3.3. Finally, the evaluation results based on the designed scenarios are reported in Section 3.4.

3.1. Experimental setup

All the experiments are run on a Windows machine with 3.0 GHz Intel Xeon Scalable Processor with 8 CPU cores and 32 GiB RAM, hosting the IFL Services. IFL Clients run on a Windows machine with 3.3 GHz Intel Xeon Scalable Processor with 2 CPU cores and 4 GiB RAM.

The implementation of the IFL system uses PySyft,[1] an open source framework for private deep learning to operate on data that resides on remote locations, i.e., the server invokes PySyft to connect to the clients and to train ML models on their local datasets. The Worker Nodes and Worker Network communication, as described in Section 2.3, is implemented using PyGrid.[2] PyGrid is based on PySyft and adds the functionality for registering nodes and searching for nodes in a network to eventually establish connection between the IFL services and the Worker Nodes on the Edge Devices.

3.2. Datasets

We evaluate our proposed approach using two real-world datasets from industry use cases. In this section, we explain the characteristics of each dataset and elaborate on the ML approaches used on these datasets.

3.2.1. Pump condition classification dataset

This dataset contains acceleration data from five pumps. The data is obtained from multiple measurements [20] using an Industrial Internet of Things (IIoT) sensor, namely the *SITRANS multi sensor* [2]. The sensor provides 512 acceleration samples at three dimensions every minute, thus providing a time series representation.

The time series data is labeled based on the conditions of the pumps. Fig. 3 shows a schematic view of the virtual Z-axis from 70,000 samples obtained from one of the sensors. As it can be seen, there are six classes indicating anomalous or healthy conditions. The healthy conditions are healthy stationary normal load (the load of pumped water at the range of $50\frac{m^3}{h}$), healthy stationary partial load (the load is $[12.5\frac{m^3}{h}, 37.5\frac{m^3}{h}]$), and idle state (the

[1] https://github.com/OpenMined/PySyft.
[2] https://github.com/OpenMined/PyGrid.

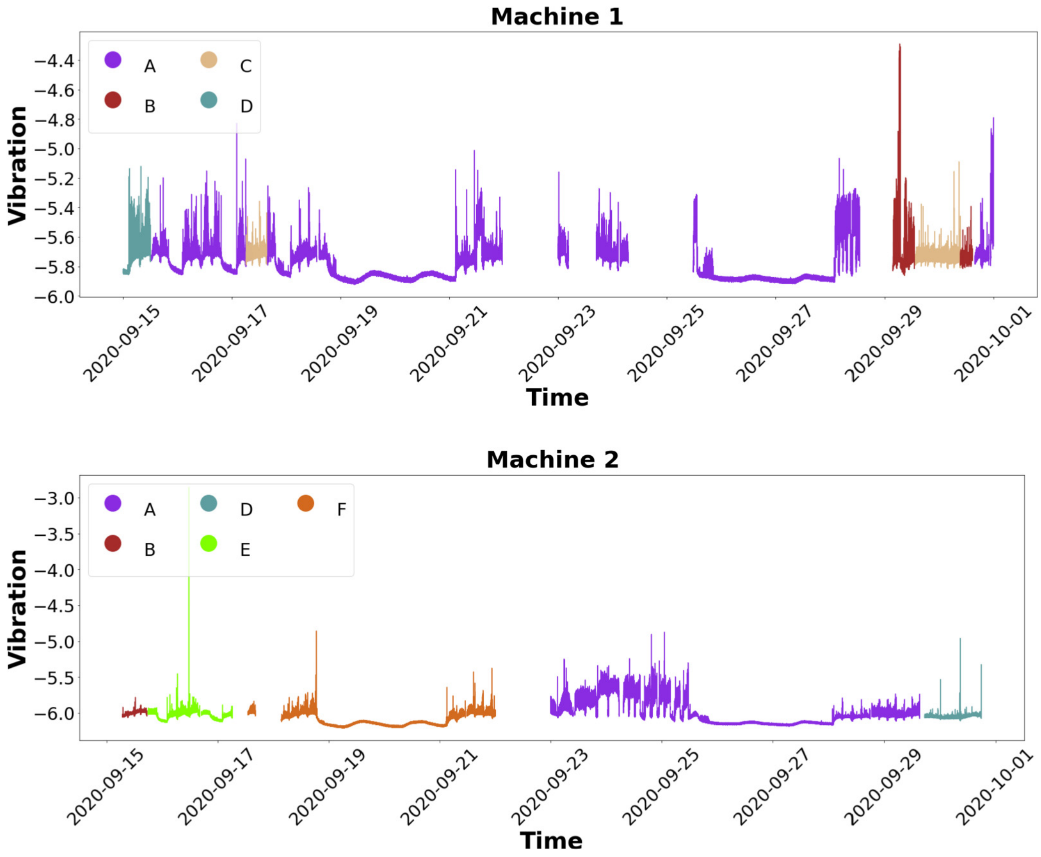


Fig. 4. Material hardness labels for two machine tools. Each row shows vibration time series data for one machine, colors indicate different material hardness.

pump is turned off). The anomalous conditions are hydraulic blockade failure (nothing was pumped at the start of pumping), dry-run error (nothing was pumped at the end of pumping), and cavitation error (the water in a pump turns to a vapor at low pressure).

We follow a sliding window approach for collecting data with a window size of 512 and a step size of 256. We then extract Mel-frequency Cepstral Coefficients (MFCC) as features (20) from the samples in each window. MFCCs are widely used as features for audio classification. We employ it here for acceleration data as both have strong relation due to air-borne and structure-borne sound transmission [9].

We consider two evaluation scenarios for this dataset: 1) *pump classification IID* where the distribution of target data (i.e., pump conditions) is IID, and 2) *pump classification non-IID* with a non-IID target distribution. In both cases, we consider the five pumps as data sources for used clients. For the IID case, we artificially create nine clients, with each client being assigned a subset of data of 1-3 pumps such that all class labels are represented. The clients receive on average 26,554 samples. For the non-IID case, we artificially create ten clients, where each pump dataset is divided between two clients on average with respect to independent and time-separated measurements. These measurements were performed on different days considering even adjustments in between like, e.g., dismantling and rebuilding of pumps, or removing and reattaching screws to address feature and label skews [20]. With this, each of the non-IID clients receives 212,738 samples on average.

As a learning approach, we use an Artificial Neural Network (ANN) with two dense layers with 64 units each using ReLu activations and followed by 40% dropout, and a last output layer without any activation function (5,894 total parameters). We use this dropout-rate to avoid that the model is biased towards a single client. We consider just two dense layers to limit the number of parameters, which increase training time and would require larger datasets. However, to facilitate classification (i.e., linear separability), the number of units per layer (64) is chosen to be significantly greater than the input features (20). This model is then considered as our base model m^p that is uploaded to the server and used by tasks of population p for evaluation.

To train all the clients based on the same scale of data, the input data is normalized using a Gaussian distribution before being fed into the model.

3.2.2. Material classification dataset

This dataset contains vibration data and material hardness labels from two machine tools that process metals. The vibration data is obtained at a frequency of 1 Hz, thus providing 60 samples per minute. The objective here is to classify material hardness based on vibration data. As it can be seen in Fig. 4, there are six material hardness classes in this dataset.

It can be observed that some materials are only machined from one of the machine tools, thus the distribution of hardness labels between the machines is not uniform. Furthermore, the dataset is imbalanced, for example, the material with hardness label *A* has been observed more than the other material hardnesses. FL is reasonable and applicable for this dataset because 1) both machines are similar in construction and 2) non-IID distribution of material hardness labels makes transferring one model for both machines not applicable.

In this use case, we demonstrate how IFL can be applied on a small scale, considering only two assets (i.e., machine tools) generating non-IID data, and still enabling ML model improvements.

The input vibration time series data is split into sequences of maximum two continuous hours during the operation time of the machines. These sequences are normalized and divided into training and test data, keeping 70% of sequence data for training. Our data is obtained from these sequences following a sliding window approach, with a window size of 120 (i.e., samples of every two minutes). For training sequences, we have overlapping windows to cover the entire data with a stepsize of 60, whereas for testing sequences, we do not use any overlapping. We then compute Fast Fourier Transform (FFT) [5] on the samples and consider the magnitude of the computed FFTs as our features. This yields a 120-dimensional feature vector.

We consider one evaluation scenario for this dataset using two clients, each representing a real (physical) machine tool. The target data distribution (i.e., material hardness) is non-IID and the data is imbalanced. We have a total number of 5,985 samples for the first client and 7,136 samples for the second client.

As a learning approach, we employ an ANN with two dense layers, each with 256 units, each using ReLu activations and followed by 40% dropout, and a last output layer without any activation function (98,310 total parameters).

3.3. Experimental design

All the scenarios as discussed in Section 3.2 are provided in a JavaScript Object Notation (JSON) format. The scenario JSON contains configuration for clients, tasks, assets, and the used model.

The client configuration provides settings for the client's name, the path to a client dataset, the tasks associated with a client, and additional organizational descriptions.

The task configuration has settings for the selected FL algorithm alg^{t_i} (FedAvg or SeqFL), the cohort strategy cb^{t_i} (if cohorts of similar clients should be built for federation and the algorithm to be used for building cohorts), and federation criteria $crit^{t_i}$ (e.g., the minimum number of required clients that need to join a population to start the training).

The asset configuration gives settings for name, description, location, and environment description of an asset.

The model configuration provides settings for the path of the base model to be used for federation, and the training parameters (e.g., number of communication rounds, number of epochs per communication round, learning rate, batch size, etc.).

Based on the aforementioned configuration parsed from the scenario JSON, the individual client processes independently create their assets, refer to a common ML model, and eventually submit tasks to the Client Registry API to join for FL.

After FL is finished, resulting ML models are validated using classification accuracy. If applicable, we consider classification accuracy on cohort test data (i.e., test data from all the clients in a cohort) to ensure comparability between models trained only on client data, models trained on central data, and models trained with FL on cohorts. To further compare cohort-based FL with FL not using cohort building (i.e., FL algorithm is applied on the overall population), we consider classification accuracy on the overall population. Finally, the clients download the model using the Processor API to conclude the IFL process which terminates the evaluation scenario.

3.4. Results

In this section, we report on experiments evaluating the performance of our IFL system on multiple scenarios from three datasets (as described in Section 3.2) using FedAvg and SeqFL algorithms.

In all the experiments, our base model for training is an ANN model. We use Pytorch [29] for training and prediction. We compute loss using the cross-entropy loss function [3]. Since we deal with non-IID data, we use a weighted cross-entropy loss function. The loss l is computed for each class c separately and can be written as:

$$\begin{aligned} l(x,c) &= weight[c] \times -\log \frac{\exp x[c]}{\sum_j \exp x[j]} \\ &= weight[c](-x[c] + \log \sum_j \exp x[j]) \end{aligned} \tag{1}$$

where x is an observation (i.e., the output logit of an ANN for a sample whose size is equal to the number of classes), and $weight[c]$ is the class weight computed for each client independently considering a balanced heuristic inspired by [21].

The class weight can be described as:

$$weight[c] = \begin{cases} \frac{N_S}{N_C \times N_{S_c}}, & \text{if } N_{S_c} > 0, \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where N_S is the total number of samples for each client, N_C is the total number of classes provided as a training parameter to each client because there can be classes which are not present for a particular client, and N_{S_c} is the number of samples for this class in a client dataset.

We perform batch optimization for training where training loss is computed on batches of data instead of the entire data due to memory efficiency and overfitting problem. For optimization, we use the Adam optimizer [22] and a batch size of 128. In order to compromise between time and performance, we keep the learning rate $1e-3$ in our experiment. A lower learning rate makes training much slower and a higher learning rate might deteriorate the performance.

3.4.1. Impact of IFL on non-IID data

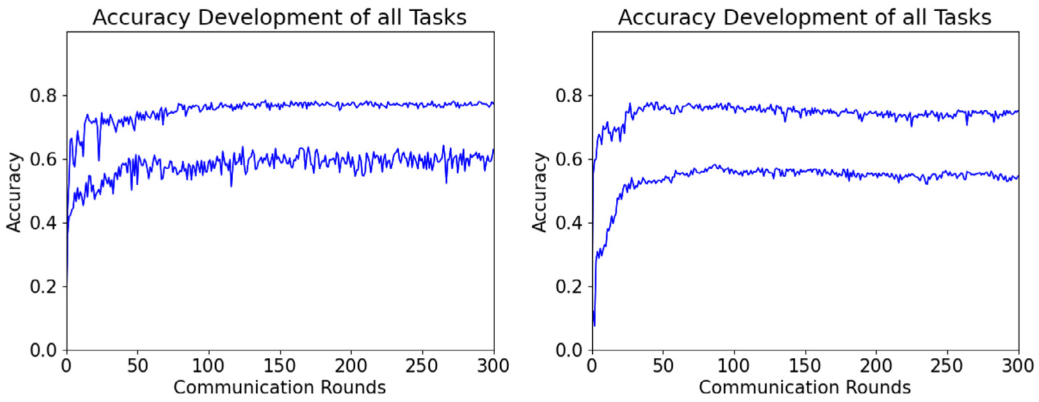
The material classification dataset as discussed in Section 3.2.2 has non-IID data and target distribution. We provided two scenarios for each FL algorithm with a default cohort strategy (when no cohort is built). We trained a federated model with two clients (one for each machine tool). The training is done for 300 communication rounds and one epoch per communication round for each client. The average accuracy on the test data of this dataset is shown in Fig. 5. As it can be seen, the accuracy of the federated model improves after each communication round from initial accuracies of 50% and less. We obtain for both clients a slightly higher accuracy using the FedAvg algorithm. But in both cases, we get the best accuracy of 76% in spite of non-IID data distribution which underlines the usability of our IFL system.

3.4.2. Impact of cohort-based IFL on IID data

We have shown the applicability of our IFL system on industrial data with two clients. Next, we evaluate our approach on the *pump classification IID* dataset (Section 3.2.1) with 9 clients where the data has an IID target distribution. Our federation criterion in both scenarios is the minimum number of clients for starting federation which we set as 9 (the total number of clients).

Fig. 6 shows evaluation results of this scenario after 30 communication rounds and 5 epochs per communication round. The performance of the federated model is compared with three other evaluation approaches:

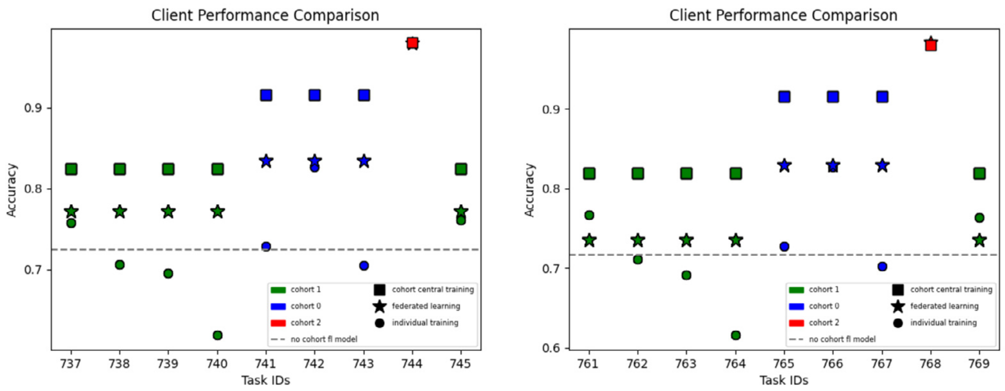
- No cohort FL model: A federated model is trained on the training data of all the clients and tested on the test data of all the clients.
- Cohort central training: Training and test data sets of cohort members are aggregated by the server which results in one joint training and test data set per cohort. For each cohort, one model is trained and tested on the corresponding cohort dataset without applying FL.
- Individual training: A model is trained on each client and transferred to the other clients (i.e., testing on cohort test data).



(a) FedAvg

(b) SeqFL

Fig. 5. Accuracy of the federated model trained on the two clients (one per two machine tool) using the material classification dataset.



(a) FedAvg

(b) SeqFL

Fig. 6. Client performance comparison on the pump classification dataset with IID target distribution. Every client is represented by a task ID and has four accuracy metrics resulting from different ways of model training: federated learning, individual training, cohort central training, and global FL (no cohort fl model).

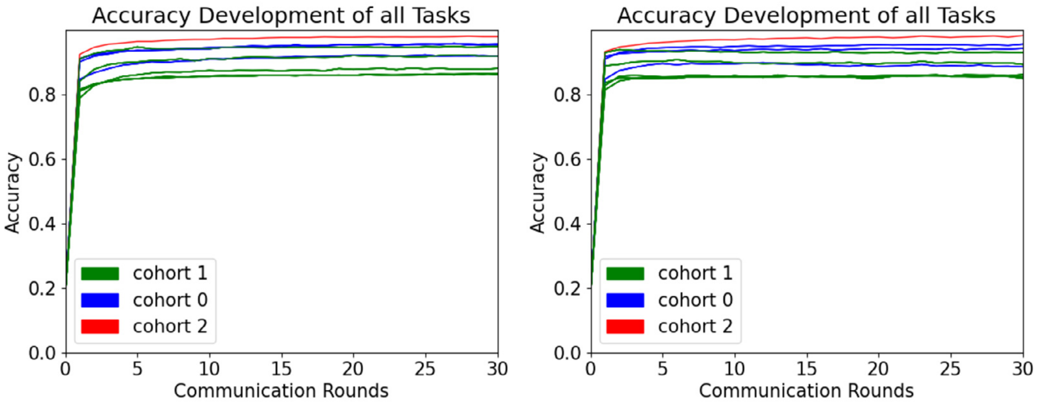
So, for each cohort and approach all clients share one model, except for individual training, which considers one model per client (task id). It can be seen in Fig. 6 that the accuracy of the cohort-based FL model is higher than the global FL model. The cohort building approach in this experiment is based on input data. The result as depicted in Fig. 6 indicates that, however the target distribution is IID, but the input data of pump conditions does not have IID distribution. More precisely, the input data of some pump conditions are more similar to each other than other pump conditions. Therefore, we obtained a higher performance by learning separate FL models on similar groups of clients. This result highlights our contribution using a cohort-based IFL approach with either FedAvg or SeqFL algorithms.

The best result achieved using federation on each cohort is shown in Fig. 6 as cohort central training. It can be observed that we can achieve an accuracy close to cohort central training using our IFL system. In order to show the importance of using FL in this scenario, we compared the accuracy of the federated model on each cohort with the individual training. We can observe that our federated model on average improves the accuracy of each client compared to the individual training. Using FedAvg, this improvement can be seen more clearly than for SeqFL. This emphasizes the applicability of FL for this experiment.

The average accuracy on the test data after each communication round is shown in Fig. 7. It can be seen that the federated model reaches an accuracy above 80% just after a few communication rounds. We can see that cohort 2 has the highest accuracy. The reason is that this cohort has only one client, therefore FL performs like a central model (without federation) for this cohort. The accuracy of FL for the clients in cohort 1 is slightly lower than others. As it can be seen in Fig. 6, the cohort central training accuracy for this cohort is also lower than the other cohorts. Therefore, FL cannot achieve a higher accuracy. One reason for the lower performance of cohort 1 is that it has more clients compared to the other cohorts and some of these clients (e.g., task 740) are not fitting the cohort very well. Therefore, it increases the risk of contributing model updates with below-average quality clients.

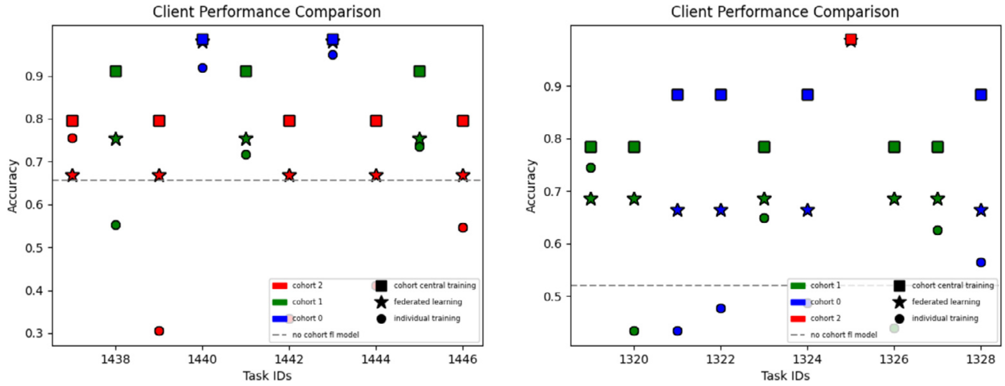
3.4.3. Impact of cohort-based IFL on non-IID data

We showed that cohorts can improve the performance of the IFL system on datasets with IID target distribution. We now go one step further and evaluate the performance of a cohort-based IFL system on the *pump classification non-IID* (Section 3.2.1) with non-IID target distribution with 10 clients. For this experiment, we also provided two scenarios, each considering one FL algorithm. The cohort strategy in both scenarios is based on input data distribution. And the federation criterion in both scenarios is the minimum number of clients for starting federation which is set as 10 (the total number of clients). Fig. 8 shows the evaluation results after 30 communication rounds and 5 epochs per communication round. It can be seen that the federated models achieve a high accuracy compared to their respective cohort central models. Furthermore, the cohort-based approach gives on average a higher performance than the no cohort federated model. This result also emphasizes the impact of finding similarity between clients and using them in federation.



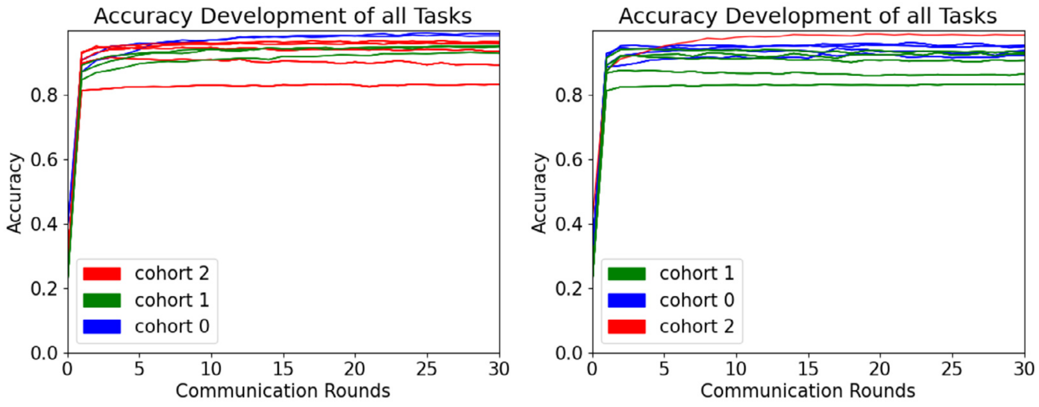
(a) FedAvg (b) SeqFL

Fig. 7. Accuracy of the federated model after each communication round on the pump classification dataset for 9 clients with IID target distribution.



(a) FedAvg (b) SeqFL

Fig. 8. Client performance comparison on the pump classification dataset with non-IID target distribution. Every client is represented by a task ID and has four accuracy metrics resulting from different ways of model training: federated learning, individual training, cohort central training, and global FL (no cohort fl model).



(a) FedAvg (b) SeqFL

Fig. 9. Accuracy of federated model after each communication round on the pump classification dataset for 10 clients with non-IID target distribution.

Fig. 9 shows the average accuracy on the client test data of this dataset after each communication round. As it can be observed, the federated model reaches at least 80% accuracy just after a few communication rounds. It can be seen that in cohorts with more clients such as cohort 2 in Fig. 9(a) or cohort 1 in Fig. 9(b), the accuracy of some clients is lower than the other clients in the same cohort. The reason is that we used the individual client test datasets and the k-means clustering approach for building cohorts. The clusters are initialized randomly and the clients get assigned to the clusters with the minimum mean squared distance. The random cluster initialization may result in outliers during assignment.

3.5. *Discussion*

As we have demonstrated in the evaluation, the IFL process with the integrated cohort building and FL algorithms can be applied to improve ML models for industrial scenarios. Our FLaaS-based approach provides FL to client processes by considering their

IFL tasks and defined metadata. So, the clients explicitly submit the tasks using the Client Registry API, and download the resulting model using the Processor API. This invocation of the service-based architecture and the IFL process addresses the challenges of (i) individual, independent, and (ii) explicit participation in FL with (iii) non-IID data. The individual implications of the presented solutions are discussed in the following:

Impact of datasets on defining clients. The respective experiments are limited to two industrial datasets with a small number of assets. Therefore, the findings may not be generalizable. For example, the pump condition classification dataset only contains acceleration data from five pumps, but resulted in 9-10 different clients. Furthermore, the measurements and conditions were recorded by different experts and due to the asset complexity, partial load and error creation resulted in additional data skews.

Impact of cohort-based FL. Our experimental evaluation proves the applicability of our cohort-based IFL system especially on non-IID data. It can be seen that the accuracy of a cohort-based approach is on average higher than for FL models without considering cohorts. However, a limitation is shown in Figs. 9(a), 9(b). The clusters are initialized randomly and the clients get assigned to the clusters with the minimum mean squared distance. The random cluster initialization may result in outliers during assignment. Therefore, providing a dynamic cohort-assignment approach that enables changes in cohort assignment between communication rounds, is part of our future work.

FL versus model transfer. The results reported on industrial datasets (i.e., pump classification and material classification datasets), indicate that FL is applicable and reasonable for these datasets. Based on our results, transferring a model from one client to the others is not applicable for these datasets and does not give higher accuracy than FL. However, as it can be seen in Fig. 6(b), the individual training result of cohort 1 for task 761 performs slightly better than the FL model. The potential reason might be that this client has either more data or more variety of data in this cohort. Making an adaptive approach, for deciding if FL or model transferring should be used in a cohort is an interesting research question we have not covered yet.

Impact of FL algorithm on model performance. FedAvg performs better than SeqFL for all the evaluated scenarios. The potential reason is that in FedAvg, the parameters of all the clients are averaged at every communication round. Whereas in SeqFL, the model is passed from one client to the other. All the clients contribute, but the federated model of SeqFL is significantly impacted by the training of the last client.

Impact of FL algorithm on fair model sharing. The selection of FL algorithms could also prevent clients from just waiting for federated model updates without individual contributions. If clients select FedAvg, each client needs to provide the locally trained model to the server before a federated model is aggregated and distributed. So, if clients do not contribute, subsequent model updates would not be received after aggregation. In contrast, selecting SeqFL, could lead to transfer models from client A to B, although B might not have contributed something before.

Impact of resource-constrained edge devices on FL. In our evaluation, all the clients are running on machines with similar resource capabilities. Yet, we have not considered heterogeneous resource-constrained edge devices and volatility in the utilization of, e.g., CPU, GPU, memory, and network. This information can be used to avoid inefficient training, e.g., due to overloaded edge devices. Since our current focus was on the applicability of the IFL system on edge devices, we considered optimizing accuracy rather than optimizing resources consumption. Our IFL services will be extended in the future to overcome these limitations.

Impact of communication intensity and latency on edge-based IFL. In Section 3, both evaluated scenarios consider time series data, where a fixed number of acceleration data samples is measured periodically. The size of the data samples per minute is less than 10 kB. Therefore, the communication intensity and latency of these applications are minor. Hence, neglecting privacy aspects, edge-based IFL could be replaced with e.g., central cloud-based ML approaches. However, there are industrial applications that train models based on high-volume and high-frequency data (e.g., images, video streams). For this, using this kind of data locally for training on edge devices and just sharing the model updates with our proposed edge-based approach is beneficial for communication-intensive applications.

Realistic applicability of the IFL system. Although independent clients can always register for FLaaS as provided by our IFL System, they can also drop out due to, e.g., missing resources, volatile connectivity, or intended stopping since the model is already mature enough. Therefore, the remaining clients in the cohort lose potential valuable contributors and the model quality could stagnate, which may cause the server to restart FL with a new cohort organization considering new clients as well. Furthermore, the IFL services do not provide rewards for clients with already mature models to join or stay in the system. This is also relevant in the early stages of the IFL process, as only a limited number of clients might have joined. Since our system creates a subset structure (tasks in cohorts in populations), only a few clients collaborate initially, which can slow down the training progress. For this, introducing a reward system for providing model updates could solve the cold start problem and could prevent drop-outs.

4. Related work

Despite being a very recent research topic, FL has been studied widely in various applications [6,35,12]. The basic idea of FL as proposed in [26] is to federate a global FL model among all the clients. However, one of the main challenges in FL is that the client's data especially in real-world scenarios are non-IID. Thus, a global FL model might not perform well for all the clients.

To overcome this problem, clustered FL approaches are proposed where clusters (or cohorts) of similar clients are identified and the model is personalized for each cluster. Clustered FL approaches can be classified into two categories: centralized approaches where the server identifies a client's assignment to a cluster [14,34], and decentralized approaches where the clients choose and update the model parameters that best fit them [15, 25]. We followed a centralized cohort (clustered) FL approach by considering features (statistical moments) extracted from the client's data. However, the IFL clients can influence the cohort building by defining the cohort building approach and by defining the federation criteria that affect prior population building. In our work, the server also identifies the number of required cohorts. Building cohorts (clusters) on the server in our work is efficient because only few features from clients are used. As it is shown in Section 3.3, a cohort-based FL approach yields a higher performance than a global FL especially for scenarios with non-IID data distribution.

The FedAvg algorithm aggregates model parameters by averaging the value of local parameters from clients in each communication round [26]. Clients often have different dataset sizes and data distributions, thus training an effective global model with a good convergence using FedAvg is challenging. Therefore, the clients' contribution in aggregation operation of FedAvg is weighted either proportional to their local dataset size [26] or using multiple criteria such as the diversity of the dataset, data quality, and dataset size of clients' local datasets [4]. In our work, all clients are weighted equally. This way, we do not use any information about clients' dataset sizes on the server. In our experiments, we still obtained good convergence during training with equal contribution weights.

Selecting edge-based clients for participating in FL rounds is an important aspect for reducing communication cost and potentially long-lasting training times. This can be due to heterogeneous compute capabilities provided on resource-constrained edge devices. Clients may be selected randomly [26,28] or based on pre-defined criteria such as availability of their resources [31]. In our approach, we follow a clustered FL approach, using all the available clients for building an FL population. As discussed in Section 3.1, clients' resources are similar in our evaluation setup. Therefore, we did not need to include any resource-based criteria on the server for selecting the clients. However, in the IFL system, we consider client-defined federation criteria for building FL populations. This could be used as a basis for defining requirements on resources that are provided on edge devices of potential FL partners.

Focusing on edge-based FL, Feraudo et al. [11] provide an architecture that enables asynchronous FL for edge devices using a publish/subscribe pattern. For this, the client registers for FL by sending a message to a message broker, which notifies the server that waits until the end of a pre-defined timespan to consider the start of FL rounds. Similarly, Bonawitz et al. [4] proposed an FL system enabling edge devices to declare their training intention on a central server that starts FL as a required number of clients have joined. Furthermore, the authors consider selection and rejection of clients on the server. Hence, both approaches address the challenge of a varying number of edge devices, particularly unavailable edge devices. Our IFL services enable clients to define, e.g., a minimum number of FL partners, using custom federation criteria. This democratic approach provides more power to the edge clients, e.g., by reducing the required minimum number of FL partners if too little clients joined in previous FL attempts due to potential unavailability.

To provide FLaaS to edge and mobile devices, Kourtellis et al. [23] provide high-level APIs that can be invoked by mobile apps to collaborate on common ML problems. The authors propose a hierarchical scheme for collecting model updates on local device services considering potentially multiple apps. The updates can be forwarded to, e.g., edge nodes or central servers for aggregation. Furthermore, the authors address challenges like the design of collaboration across (hierarchical) network topologies, permission and privacy management, and forms of providing FL-based ML models to clients. However, the approach does not provide a service for building cohorts based on the client data distribution, which is relevant for non-IID datasets.

5. Conclusion

In this paper, we have presented the IFL system, consisting of the IFL services and the IFL client that provide FLaaS for edge-based clients connected to industrial machines. The machine data is used by the IFL client to collaboratively train ML models together with other FL clients. For this, we introduced the IFL process that includes a four-step approach enabling cohort-based FL and therefore addressing the challenge of non-IID data. We proposed a service architecture that supports the IFL process by providing APIs to clients for participating in FL rounds.

As discussed, our IFL system has three main contributions, (i) explicit participation in FL on an on-demand basis, (ii) an architecture supporting individual and independent collaboration on shared ML models, and (iii) handling non-IID data distributions using cohort-building.

To the best of our knowledge, we provided the first cohort-based FL service system considering the characteristics and requirements of industrial clients. Furthermore, using client metadata and statistical moments for cohort building is a unique approach that both considers client preferences and local data distribution to handle non-IID data.

Evaluation on diverse and real-world industrial data is missing in current FL literature. We have shown the potential of our IFL system by applying it on two completely diverse and large real-world industrial datasets, used for pump condition classification and material classification. Our results also show the importance of a cohort-based FL approach, yielding higher accuracies on average compared to executing FL algorithms on the overall population.

The proposed IFL system is an important step towards incorporating FL in industrial applications and opens future directions for FL research based on the discussed limitations (see Section 3.5). For this, future work needs to consider dynamic cohort reorganization and reward management to address client volatility, partially adding model updates from neighboring cohorts to further optimize model performance, and resource-optimized training on heterogeneous edge devices.

CRediT authorship contribution statement

Thomas Hiessl: Conceptualization, Investigation, Project administration, Software, Writing – original draft, Writing – review & editing. **Safoura Rezapour Lakani:** Investigation, Software, Writing – original draft, Writing – review & editing. **Jana Kemnitz:** Data curation, Writing – review & editing. **Daniel Schall:** Conceptualization, Resources, Supervision. **Stefan Schulte:** Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The financial support by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development as well as the Christian Doppler Research Association (Blockchain Technologies for the IoT) is gratefully acknowledged.

References

[1] S. Bagavathiappan, B. Lahiri, T. Saravanan, J. Philip, T. Jayakumar, Infrared thermography for condition monitoring – a review, Infrared Phys. Technol. 60 (2013) 35–55.

[2] T. Bierweiler, H. Grieb, S. von Dosky, M. Hartl, Smart sensing environment – use cases and system for plant specific monitoring and optimization, in: Automation 2019, VDI Verlag, 2019, pp. 155–158.

[3] C.M. Bishop, Pattern Recognition and Machine Learning, Springer New York, NY, USA, 2007.

[4] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecný, S. Mazzocchi, H.B. McMahan, T.V. Overveldt, D. Petrou, D. Ramage, J. Roselander, Towards federated learning at scale: system design, arXiv preprint, arXiv:1902.01046, 2019.

[5] R.N. Bracewell, The Fourier Transform and Its Applications, McGraw-Hill New York, 1986.

[6] T.S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I.C. Paschalidis, W. Shi, Federated learning of predictive models from federated electronic health records, Int. J. Med. Inform. 112 (2018) 59–67.

[7] I.S. Candanedo, E.H. Nieves, S.R. González, M.T.S. Martín, A.G. Briones, Machine learning predictive model for industry 4.0, in: L. Uden, B. Hadzima, I.-H. Ting

(Eds.), Knowledge Management in Organizations, Springer International Publishing, Cham, 2018, pp. 501–510.

[8] K. Chang, N. Balachandar, C. Lam, D. Yi, J. Brown, A. Beers, B. Rosen, D. Rubin, J. Kalpathy-Cramer, Distributed deep learning networks among institutions for medical imaging, J. Am. Med. Inform. Assoc. 25 (2018).

[9] L. Cremer, M. Heckl, Structure-Borne Sound: Structural Vibrations and Sound Radiation at Audio Frequencies, Springer Science & Business Media, 2013.

[10] A.Y. Ding, E. Peltonen, T. Meuser, A. Aral, C. Becker, S. Dustdar, T. Hiessl, D. Kranzlmuller, M. Liyanage, S. Magshudi, N. Mohan, J. Ott, J.S. Rellermeyer, S. Schulte, H. Schulzrinne, G. Solmaz, S. Tarkoma, B. Varghese, L. Wolf, Roadmap for edge AI: a Dagstuhl perspective, arXiv preprint, arXiv:2112.00616, 2021.

[11] A. Feraudo, P. Yadav, V. Safronov, D.A. Popescu, R. Mortier, S. Wang, P. Bellavista, J. Crowcroft, Colearn: enabling federated learning in mud-compliant iot edge networks, in: Third ACM International Workshop on Edge Systems, Analytics and Networking (EdgeSys '20), Association for Computing Machinery, 2020, pp. 25–30.

[12] F. Fioretto, P. Van Hentenryck, Privacy-preserving federated data sharing, in: 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 638–646.

[13] Z. Ge, Z. Song, S.X. Ding, B. Huang, Data mining and analytics in the process industry: the role of machine learning, IEEE Access 5 (2017) 20590–20616.

[14] A. Ghosh, J. Hong, D. Yin, K. Ramchandran, Robust federated learning in a heterogeneous environment, arXiv preprint, arXiv:1906.06629, 2019.

[15] A. Ghosh, J. Chung, D. Yin, K. Ramchandran, An efficient framework for clustered federated learning, arXiv preprint, arXiv:2006.04088, 2020.

[16] T. Hiessl, D. Schall, J. Kemnitz, S. Schulte, Industrial federated learning – requirements and system design, in: Highlights in Practical Applications of Agents, Multi-Agent Systems, and Trust-worthiness. The PAAMS Collection, Springer International Publishing, 2020, pp. 42–53.

[17] K. Hsieh, A. Phanishayee, O. Mutlu, P.B. Gibbons, The non-IID data quagmire of decentralized machine learning, in: 36th International Conference on Machine Learning (ICML), in: PMLR, 2019, pp. 4387–4398.

[18] P. Kairouz, H.B. McMahan, B. Avent, A. Bellet, M. Bennis, A.N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R.G.L. D'Oliveira, H. Eichner, S.E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P.B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konecný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, H. Qi, D. Ramage, R. Raskar, M. Raykova, D. Song, W. Song, S.U. Stich, Z. Sun, A.T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F.X. Yu, H. Yu, S. Zhao, Advances and open problems in federated learning, Found. Trends Mach. Learn. 14 (1–2) (2021) 1–210.

[19] T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, A.Y. Wu, An efficient k-means clustering algorithm: analysis and implementation, IEEE Trans. Pattern Anal. Mach. Intell. 24 (7) (2002) 881–892.

[20] J. Kemnitz, T. Bierweiler, H. Grieb, S. von Dosky, D. Schall, Towards robust and transferable iiot sensor based anomaly classification using artificial intelligence, arXiv preprint, arXiv:2110.03440, 2021.

[21] G. King, L. Zeng, Logistic regression in rare events data, Polit. Anal. 9 (2) (2001) 137–163.

[22] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, arXiv preprint, arXiv:1412.6980, 2014.

[23] N. Kourtellis, K. Katevas, D. Perino, Flaas: federated learning as a service, in: Proceedings of the 1st Workshop on Distributed Machine Learning, DistributedML'20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 7–13.

[24] M. Kritzler, J. Hodges, D. Yu, K. Garcia, H. Shukla, F. Michahelles, Digital companion for industry, in: Companion Proceedings of the 2019 World Wide Web Conference, WWW '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 663–667.

[25] Y. Mansour, M. Mohri, J. Ro, A.T. Suresh, Three approaches for personalization with applications to federated learning, arXiv preprint, arXiv:2002.10619, 2020.

[26] H.B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: 20th International Conference on Artificial Intelligence and Statistics (AISTATS), in: PMLR, 2016, pp. 1273–1282.

[27] V. Mothukuri, R.M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, G. Srivastava, A survey on security and privacy of federated learning, Future Gener. Comput. Syst. 115 (2021) 619–640.

[28] T. Nishio, R. Yonetani, Client selection for federated learning with heterogeneous resources in mobile edge, in: 2019 IEEE International Conference on Communications (ICC), IEEE, 2019, pp. 1–7.

[29] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch, 2017.

[30] K. Pillutla, S.M. Kakade, Z. Harchaoui, Robust aggregation for federated learning, arXiv preprint, arXiv:1912.13445, 2019.

[31] S.A. Rahman, H. Tout, A. Mourad, C. Talhi, FedMCCS: multi criteria client selection model for optimal iot federated learning, IEEE Int. Things J. 8 (2020) 4723–4735.

[32] P.J. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, J. Comput. Appl. Math. 20 (1987) 53–65.

[33] V. Satopaa, J. Albrecht, D. Irwin, B. Raghavan, Finding a "kneedle" in a haystack: detecting knee points in system behavior, in: 2011 31st International Conference on Distributed Computing Systems Workshops, IEEE, 2011, pp. 166–171.

[34] F. Sattler, K.R. Müller, W. Samek, Clustered federated learning: model-agnostic distributed multitask optimization under privacy constraints, IEEE Trans. Neural Netw. Learn. Syst. (2020) 1–13.

[35] M.J. Sheller, G.A. Reina, B. Edwards, J. Martin, S. Bakas, Multi-institutional deep learning modeling without sharing patient data: a feasibility study on brain tumor segmentation, in: International MICCAI Brainlesion Workshop, Springer, 2018, pp. 92–104.

[36] R. Shokri, V. Shmatikov, Privacy-preserving deep learning, in: 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS), ACM, 2015, pp. 1310–1321.

[37] R.L. Thorndike, Who belongs in the family?, Psychometrika 18 (4) (1953) 267–276.

[38] M. Xu, Z. Fu, X. Ma, L. Zhang, Y. Li, F. Qian, S. Wang, K. Li, J. Yang, X. Liu, From Cloud to Edge: A First Look at Public Edge Platforms, Association for Computing Machinery, New York, NY, USA, 2021, pp. 37–53.

[39] D. Yin, Y. Chen, R. Kannan, P. Bartlett, Byzantine-robust distributed learning: towards optimal statistical rates, in: 35th International Conference on Machine Learning (ICML), in: PMLR, vol. 80, Stockholmsmässan, Stockholm, Sweden, 2018, pp. 5650–5659.

[40] P. Yu, L. Wynter, S.H. Lim, Fed+: a family of fusion algorithms for federated learning, arXiv preprint, arXiv:2009.06303, 2020.

[41] X.M. Zhao, Q.H. Hu, Y.G. Lei, M.J. Zuo, Vibration-based fault diagnosis of slurry pump impellers using neighbourhood rough set models, Proc. Inst. Mech. Eng., Part C, J. Mech. Eng. Sci. 224 (4) (2010) 995–1006.

Thomas Hiessl received his diploma degree in computer science from TU Wien in 2017. He then joined Siemens Austria, working on the development of industrial Internet of Things technology and distributed artificial intelligence. He is currently pursuing the Ph.D. in computer science at TU Wien. His research interests include federated learning, cloud computing and edge computing in Internet of Things applications.

Safoura Rezapour Lakani is a data scientist by the Distributed AI System research group at Siemens Technology in Vienna. She received her PhD in computer science from the University of Innsbruck in 2018. Her research interests include machine learning, in particular transfer and federated learning.

Jana Kemnitz is a senior data scientist and machine learning expert at the Distributed AI System research group at Siemens Technology in Vienna. She received her PhD as a Marie Curie fellow from the Paracelsus Medical University. Her main interests are image- and signal processing, machine learning and medical science.

Daniel Schall is head of the Distributed AI System research group at Siemens Technology in Vienna. His main interests are AI systems, platforms, and applications that have real-world impact. He published more than 80 journal and conference papers and 3 books on service-oriented computing. Daniel Schall received his PhD in computer science from TU Wien in 2009.

Stefan Schulte is Associate Professor and head of the Christian Doppler Laboratory Blockchain Technologies for the Internet of Things at the Faculty of Informatics at TU Wien. His research interests span the areas of cloud computing and Internet of Things, and the application and extension of blockchain technologies. Findings from his research have been published in more than 100 refereed scholarly publications.