BACHELOR THESIS

Evaluation and Optimization of Knowledge-Aggregation Algorithms in an Industrial Federated Learning System

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Technical Mathematics

under the supervision of

Assoc. Prof. Dr. techn. Dipl.-Ing. Clemens Heitzinger
Institute of Analysis and Scientific Computing, TU Wien

by

Stephanie Holly
Registration number: 11703485

Vienna, March 2021

Acknowledgement

First, I would like to express my gratitude to my supervisor, Prof. Clemens Heitzinger. Moreover, this thesis would not have been possible without the support of many people. I wish to thank all the colleagues of the research group I had the opportunity to meet: Jana Kemnitz, Thomas Hiessl and Safoura Rezapour Lakani.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Bachelorarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

[illegible]

Wien, März 2021

Contents

1 Introduction

The field of machine learning is constantly evolving and spreading into a growing application sector, such as computer vision, automatic speech recognition and natural language processing [2]. This success of machine learning technologies has been driven by the wealth of available data generated by modern devices such as mobile phones. Mobile phones have access to a large amount of data, however, this data is often privacy sensitive [1] and therefore unsuitable for a central machine learning approach. Conventionally, the data is collected in a centralized storage violating privacy rights and a global model is trained on this collection of data. McMahan et al. [1] introduced an alternative that does not collect the data nor allow a server to access the data and termed this decentralized approach *Federated Learning* (FL).

In the following, we want to characterize FL and describe a typical FL process. We assume that K clients, $K \in \mathbb{N}_{\geq 2}$, with their respective local datasets, wish to collaboratively train a shared machine learning model [2]. In the whole FL process, the local data of a client does not leave that client and the data is not exposed to a server or other clients. On each client, a local model is trained on the respective data. A client can transfer its model to a server. The aim is to encrypt the transferred knowledge such that the data cannot be re-engineered. The server then aggregates the knowledge exchange to build the global model. The performance of this aggregated model should approximate the performance of a single model trained on the collection of all data. However, we allow the FL approach to perform a little less than the corresponding central approach, collecting all data and training a single model.

A typical FL process consists of the following parts [3], see figure 1.1. A central server orchestrates the learning process and repeatedly executes the following steps until a stopping criterion is satisfied:

1. **Client selection:** The server selects a set of clients.
2. **Broadcast:** The server sends the current global model to each of the selected clients.
3. **Client computation:** Each of the selected clients trains the downloaded global model on its local data and updates the model.
4. **Aggregation:** Each of the selected clients sends its updated model back to the server that aggregates these updated models.
5. **Model Update:** The server updates the global model based on the aggregation.

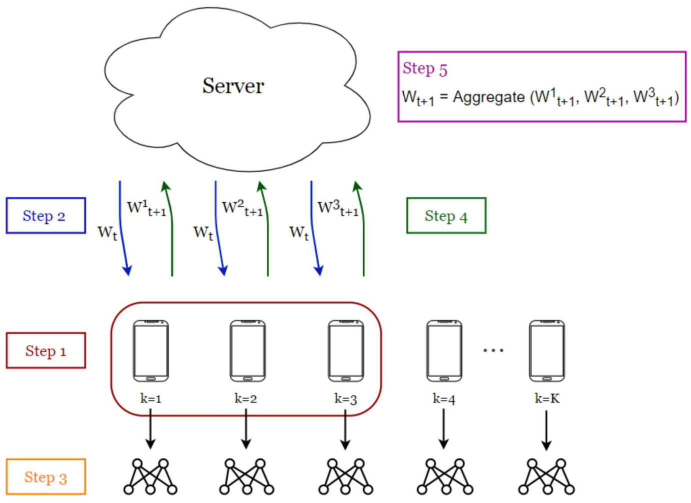

Figure 1.1: Typical FL process

FL has primarily focused on supervised learning tasks where labels are available on each client [3]. Therefore, we will only consider a supervised learning task throughout this work. Supervised learning refers to a machine learning problem on labeled data. Therefore, the data is represented in pairs of observations (x, y) for features x in a sample space $\mathcal{X}$ and labels y in a label space $\mathcal{Y}$. Supervised learning aims at estimating a functional mapping to predict the label y of a given feature x. In the FL setting, we denote

$$D_k := D_{\mathcal{X},k} \times D_{\mathcal{Y},k} \subseteq \mathcal{X} \times \mathcal{Y}$$

as the dataset of client k with $1 \leq k \leq K$.

The performance of a machine learning model strongly depends on the amount of available data. However, in many machine learning applications, it is hard to obtain the amount of data that is required to empower machine learning applications [2]. Modern devices, such as mobile phones, that naturally collect a large amount of data through several sensors, including cameras and microphones [1], provide an opportunity for machine learning applications. However, the data is usually privacy sensitive. The public awareness of data protection is growing and users are increasingly concerned about the misuse of their private information [2]. Since FL decouples model training from the need for direct access to the local data [1], FL allows users to improve the usability of private devices while preserving user privacy and data security [2]. Also, in an industrial context, data is often only available to a limited degree for individual machines motivating organizations to collaborate with industry partners to reach a satisfying level of performance without sharing vulnerable business information [8].

However, FL does not only bring benefits but also faces a number of major challenges, some of which are only present in FL in an industrial context, while others plague the FL setting in general. In a typical FL setting, and especially in an industrial context, we will encounter clients with non-identically distributed data. The performance of FL decreases significantly for non-identically distributed data [12]. For example, industrial assets typically produce different anomalous operating conditions resulting in dissimilarity in the label distribution. Zhao et al. [12] proved that a highly skewed label distribution significantly reduces the accuracy of the aggregated model in FL. In an industrial context, the feature distribution plays a key role in data heterogeneity. For example, variations in machine type, operational- and environmental conditions influence the feature distribution. In 2020, Hiessl et al. [8] identified challenges of FL in an industrial context and introduced *Industrial Federated Learning* (IFL) that adjusts FL to a learning task in an industrial context. In contrast to FL, in step 1 of the typical FL process, the server does not arbitrarily select the clients but only orchestrates knowledge exchange between clients that have sufficiently similar data to prevent a decrease of the aggregated model performance. Hiessl et al. [8] refer to the set of clients that collaboratively train a machine learning model as a *cohort*.

The performance of a machine learning algorithm is highly sensitive to the choice of its hyperparameters. Therefore, hyperparameter selection is a crucial task in the optimization of knowledge-aggregation algorithms. In a FL setting, hyperparameter optimization poses new challenges and is a major open research area. In this work, we want to investigate the impact of different hyperparameter optimization approaches in an IFL system.

We believe that the data distribution influences the choice of the best hyperparameter configuration and suggest that the best hyperparameter configuration for a client might differ from another client based on individual data properties. Therefore, we want to investigate a local hyperparameter optimization approach that – in contrast to a global hyperparameter optimization approach – allows every client to have its own hyperparameter configuration. The local approach allows us to optimize hyperparameters prior to the federation process reducing communication costs. Communication is considered a critical bottleneck in FL. Clients are usually limited in terms of communication bandwidth enhancing the importance of reducing the number of communication rounds or using compressed communication schemes for the model updates to the central server. Dai et al. [4] introduced *Federated Bayesian Optimization* (FBO) extending Bayesian optimization to the FL setting. In FBO, every client locally uses Bayesian optimization to find the optimal hyperparameter configuration. Additionally, each client is allowed to request for information from other clients. Dai et al. [4] proved a convergence guarantee for this algorithm and its robustness against heterogeneity. However, until now, there is no research on the impact of global and local hyperparameter optimization of a FL task with heterogeneous clients. Therefore, we compare a local hyperparameter optimization approach to a global hyperparameter optimization approach, optimizing hyperparameters in the federation process.

The aim of this work is i) to analyse challenges and formal requirements in FL, and in particular in IFL ii) evaluate the performance of an IoT sensor based classification task in an IFL system iii) investigate a communication efficient hyperparameter optimization approach iv) compare different hyperparameter optimization algorithms. Therefore, we want to answer the following questions.

Q1: Does FL work for an IoT sensor based anomaly classification task on industrial assets with non-identically distributed data in an IFL system with a cohort strategy?

Q2: Can we assume that the global and local hyperparameter optimization approach deliver the same hyperparameter configuration in an identically distributed FL setting?

Q3: Can we reduce communication costs in the hyperparameter optimization of a non-identically distributed classification task in context of FL by optimizing a hyperparameter locally prior to the federation process?

Q4: Does Bayesian optimization outperform grid search, both in a global and local approach of a non-identically distributed IoT sensor based classification task?

2 Optimization Algorithms

2.1 Algorithmic Challenges and Formal Requirements

In FL, new algorithmic challenges arise that differentiate the corresponding optimization problem from a distributed optimization problem. In distributed learning settings, major assumptions regarding the training data are made which usually fail to hold in a FL setting [5]. These assumptions, however, play a crucial part in the analysis of optimization algorithms and can have a strong influence on their performances. Moreover, possibly expensive and unreliable communication poses further challenges for optimization algorithms. The optimization problem in FL is therefore referred to as federated optimization emphasizing the difference to distributed optimization [1]. In an IFL setting, additional challenges regarding industrial aspects arise. In this section, we want to formulate the federated optimization problem and discuss the algorithmic challenges of FL in general, and in particular of IFL.

2.1.1 Problem Formulation

We consider a supervised learning task with features x in a sample space $\mathcal{X}$ and labels y in a label space $\mathcal{Y}$. We assume that we have K available clients, $K \in \mathbb{N}_{\geq 2}$, with

$$D_k := D_{\mathcal{X},k} \times D_{\mathcal{Y},k} \subseteq \mathcal{X} \times \mathcal{Y}$$

denoting the dataset of client k with $1 \leq k \leq K$ and $n_k := |D_k|$ denoting the cardinality of the client's dataset. Let $\mathcal{Q}$ denote the distribution over all clients, and let $\mathcal{P}_k$ denote the data distribution of client k. We can then access a specific data point by first sampling a client $k \sim \mathcal{Q}$ and then sampling a data point $(x, y) \sim \mathcal{P}_k$ [3]. Then, the local objective function is

$$F_k(w) := \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{P}_k} [f(x, y, w)]$$

where $w \in \mathbb{R}^d$ represents the parameters of the machine learning model and $f(x, y, w)$ represents the loss of the prediction on sample (x, y) for the given parameters w. Typically, we wish to minimize

$$F(w) = \frac{1}{K} \sum_{k=1}^{K} F_k(w).$$

Let $n := \sum_{k=1}^{K} n_k$ denote the total number of samples distributed over all clients.

2.1.2 Federated Learning

In the next section, we want to discuss the algorithmic challenges of solving a federated optimization problem as described above. One of the major challenges concerns data heterogeneity. In general, we cannot assume that the data is identically distributed over the clients, that is $\mathcal{P}_k = \mathcal{P}_l$ for all $k, l = 1, \ldots, K$. Therefore, F_k might be an arbitrarily bad approximation of F [1]. In real-world problems, the data D_k on a given client k depends on individual conditions, thus this local dataset is not necessarily representative of the dataset D_l of client l.

In the following, we want to analyse different non-identically distributed settings as demonstrated by Kairouz et al. [3] assuming that we have an IoT sensor based anomaly classification task in an industrial context. Given the distribution $\mathcal{P}_k$, let $P^k_{\mathcal{X},\mathcal{Y}}$ denote the bivariate probability function, let $P^k_{\mathcal{X}}$ and $P^k_{\mathcal{Y}}$ denote the marginal probability function respectively. Using the conditional probability function $P^k_{\mathcal{Y}|\mathcal{X}}$ and $P^k_{\mathcal{X}|\mathcal{Y}}$, we can now rewrite the bivariate probability function as

$$P^k_{\mathcal{X},\mathcal{Y}}(x, y) = P^k_{\mathcal{Y}|\mathcal{X}}(y|x)P^k_{\mathcal{X}}(x) = P^k_{\mathcal{X}|\mathcal{Y}}(x|y)P^k_{\mathcal{Y}}(y)$$

for $(x, y) \in \mathcal{X} \times \mathcal{Y}$. This allows us to characterise different settings of non-identically distributed data:

Feature distribution skew

We assume that $P^k_{\mathcal{Y}|\mathcal{X}} = P^l_{\mathcal{Y}|\mathcal{X}}$ holds for all clients k and l. However, $P^k_{\mathcal{X}} = P^l_{\mathcal{X}}$ possibly fails to hold for all clients k and l. Clients that have the same anomaly classes might still have differences in the measurements due to variations in sensor and machine type.

Label distribution skew

We assume that $P^k_{\mathcal{X}|\mathcal{Y}} = P^l_{\mathcal{X}|\mathcal{Y}}$ holds for all clients k and l. However, $P^k_{\mathcal{Y}} = P^l_{\mathcal{Y}}$ possibly fails to hold for all clients k and l. The distribution of labels might vary across clients as clients might experience different anomaly classes.

Same label, different features

We assume that $P^k_{\mathcal{Y}} = P^l_{\mathcal{Y}}$ holds for all clients k and l. However, the conditional distribution might vary across clients. We cannot assume that $P^k_{\mathcal{X}|\mathcal{Y}} = P^l_{\mathcal{X}|\mathcal{Y}}$ holds for all clients k and l. The same anomaly class can have significantly different features for different clients due to variations in machine type, operational- and environmental conditions.

Same features, different label

We assume that $P^k_{\mathcal{X}} = P^l_{\mathcal{X}}$ holds for all clients k and l. However, the conditional distribution might vary across clients. We cannot assume that $P^k_{\mathcal{Y}|\mathcal{X}} = P^l_{\mathcal{Y}|\mathcal{X}}$ holds for all clients k and l. The same features can have different labels due to operational- and environmental conditions, variation in manufacturing, maintenance et cetera.

Quantity skew

We cannot assume that different clients hold the same amount of data, that is $n_k = n_l$ for all $k, l = 1, \ldots, K$. Some clients will generate more data than others resulting in different amounts of local data.

In real-world problems, we expect to find a mixture of these non-identically distributed settings.

In FL, heterogeneity does not exclusively refer to a non-identical data distribution, but also addresses violations of independence assumptions on the distribution $\mathcal{Q}$ [3]. Due to limited, slow and unreliable communication on a client, the availability of a client is not guaranteed for all communication rounds. The availability of a client strongly depends on technical requirements and on local conditions, e.g. a client might lose internet connection at day time and might have a better internet connection at night time. Then, the distribution $\mathcal{Q}$ is not independent [3]. Also, an active client can drop out of training at a given communication round. Similarly, a new client fulfilling the respective technical requirements can participate in training at a given communication round.

Communication is considered a critical bottleneck in FL [3]. In each communication round, the participating clients send a full model update w back to the central server for aggregation. In a typical FL setting, however, the clients are usually limited in terms of communication bandwidth. Consequently, it is crucial to minimize the communication costs by reducing the number of communication rounds or using compressed communication schemes for the model updates to the central server on each client. At the same time, clients have a relatively fast processor and their local dataset D_k is relatively small compared to the total dataset size n [2]. Thus, the obvious approach is to use additional computation to reduce the number of communication rounds.

In traditional approaches, the performance of a machine learning model is most commonly defined as the model accuracy. Since communication in federated learning is much more expensive than computation, it is crucial to minimize communication. Therefore, we define performance as the highest classification accuracy achieved after a given amount of communication. The communication can either be in terms of communication rounds between a server and its clients, or uploaded models from each client [3].

2.1.3 Industrial Federated Learning

In an industrial setting, FL experiences challenges that specifically occur in an industrial context. Industrial assets have access to a wealth of data suitable for machine learning models, however, the data on an individual asset is typically limited and private in nature. In addition to sharing the data within the company, Hiessl et al. [8] propose sharing the data with an external industry partner. FL leaves possibly critical business information distributed on the individual client (or within the company) and simultaneously improves the performance of the machine learning model on the client. However, in section 2.2 we have already seen that heterogeneity is a major challenge in FL. Zhao et al. [12] proved that a highly skewed label distribution significantly reduces the accuracy of the aggregated model in FL. In an industrial context, we expect to find heterogeneous clients due to varying environmental and operational conditions on different assets. Therefore, Hiessl et al. [8] introduced a modified approach of FL in an industrial context and termed it *Industrial Federated Learning* (IFL). IFL does not allow arbitrary knowledge exchange between clients. Instead, the knowledge exchange only takes place between clients that have sufficiently similar data to prevent a decrease of the global model performance. Hiessl et al. [8] refer to this set of clients as a *cohort*. In IFL, individual clients do not only profit from collaboratively training a global model on a larger amount of data, but the data is also sufficiently similar preventing negative knowledge transfer. We expect the federated learning approach in such a cohort to approximate the corresponding central learning approach.

For example, we consider an industrial classification problem. The task is the classification of the operating condition of an industrial asset. The data is measured by a sensor that is attached to the asset. We expect to find differences in the measurements due to variations in sensor and asset type, and therefore differences in the feature distribution. Also, each asset generates healthy data samples and anomalous data samples. In reality, the anomalous conditions differ from asset to asset. Therefore, we expect to find differences in the label distribution.

Apart from the data dissimilarity, another challenge arises in FL in an industrial context. For further analysis, we consider an industrial classification problem: Let $\mathcal{X} \subset \mathbb{R}^n$ denote the sample space and let $\mathcal{Y}$ denote the label space. We assume without loss of generality that $\mathcal{Y} = \{1, \ldots, C\} \subset \mathbb{N}$ for a constant $C \in \mathbb{N}$ where y indicates a condition, such as healthy or different types of anomalies. Let x denote a data point generated by an asset in an industrial context. In a classification problem, we are interested in correctly classifying the exact condition. Typically, a machine learning model is trained to minimize the cross-entropy loss. The standard categorical cross-entropy loss function is given by [10]

$$E(w) := -\frac{1}{N} \sum_{c=1}^{C} \sum_{n=1}^{N} y_n^c \log\big(h_w(x_n, c)\big),$$

where y_n^c is the target label for data sample n for class c, x_n is the input feature for data sample n and h_w is the model with weights w for class c. Naturally, we expect the number of healthy data samples to be significantly larger than the number of anomalous data samples [9]. The rare anomalous samples are more valuable than the numerous healthy samples. A new hyperparameter γ_c allows us to penalize individual classification errors differently. We can then rewrite the weighted categorical cross-entropy loss function as [10]

$$E(w) := -\frac{1}{N} \sum_{c=1}^{C} \sum_{n=1}^{N} \lambda_c y_n^c \log\big(h_w(x_n, c)\big),$$

where λ_c is the weight for class c.

2.2 Federated Averaging

In the following, we will present the Federated Averaging (FedAvg) algorithm according to McMahen et al. [1], a form of stochastic gradient descent (SGD) and the state-of-the-art algorithm in FL. McMahan et al. [1] introduced the FedAvg algorithm for federated optimization and responded to the challenges of FL. In section 2.1.2, we have seen that communication is a critical bottleneck in FL and that an obvious approach for reducing communication costs is to increase computation. McMahan et al. [1] proposed two ways of adding computation: increased parallelism (using more clients working independently between each communication round) and increased computation on each client. The FedAvg algorithm addresses both approaches by controlling the fraction C of clients that perform computation in each communication round r, the number of local epochs E on each client and the local mini-batch size B. By increasing the fraction C we can increase parallelism and by increasing the number of local epochs E and the mini-batch size B we can increase computation on each client.

Let K denote the number of clients, R the number of communication rounds, D_k the local dataset of client k, and η the learning rate. Let w_r^k denote the machine learning model's weight of client k in communication round r and let $w_r = \sum_{k=1}^{K} \frac{n_k}{n} w_r^k$ denote the aggregated model weight in communication round r. Then, the optimization problem is

$$\min_{w \in \mathbb{R}^d} F(w) = \min_{w \in \mathbb{R}^d} \frac{n_k}{n} \sum_{k=1}^{K} F_k(w)$$

with

$$F_k(w) := \frac{1}{n_k} \sum_{i=1}^{n_k} f_i(w),$$

where $f_i(w) := \mathcal{L}(x_i, y_i, w)$ is defined as the loss of the prediction on sample (x_i, y_i) for the given model weight w [2].

In algorithm 1, we give the pseudocode of the FedAvg algorithm. First, the server initializes a model weight $w_0 \in \mathbb{R}^d$. Then, for each communication round r, the server determines a randomly selected subset S_r of $C \cdot K$ clients and sends the latest model weight w_r to all clients $k \in S_r$. Now, each client $k \in S_r$ executes the following steps: The client obtains the latest model weight w from the server. Then the dataset D_k is randomly divided into $\frac{n_k}{B}$ batches. For each epoch, the client updates the model weight $w = w - \eta \nabla \mathcal{L}(b, w)$ for all batches $b \in \mathcal{B}$. The client sends the model update back to the server. The server then aggregates all model weights $w_r = \sum_{k=1}^{K} \frac{n_k}{n} w_r^k$.

Algorithm 1: Federated Averaging

```
Server executes:
initialize $w_0$
for each communication round $r = 1, \dots, R$ do
    $S_r :=$ random set of $C \cdot K$ clients
    for each client $k \in S_r$ do
        $w_r^k :=$ ClientUpdate$(k, w_{r-1})$
    end
    $w_r := \sum_{k=1}^{K} \frac{n_k}{n} w_r^k$
end
Clients execute:
ClientUpdate$(k, w)$ :
$\mathcal{B} :=$ set of batches of size $B$ (randomly divide dataset $D_k$ into batches)
for each epoch $i = 1, \dots, E$ do
    for each batch $b \in \mathcal{B}$ do
        $w := w - \eta \nabla \mathcal{L}(b, w)$
    end
end
return $w$
```

2.3 Hyperparameters

The performance of a machine learning algorithm is highly sensitive to the choice of hyperparameters. Therefore, hyperparameter selection is a crucial task in the optimization of knowledge-aggregation algorithms. Hyperparameters are parameters that control the machine learning algorithm and have to be defined prior to the learning process. Considering the FedAvg algorithm in section 2.2, we have to define prior to the training process the fraction C of clients that performs computation, the number R of communication rounds, the number E of local epochs, the mini-batch size B, the learning rate η, and all hyperparameters regarding the model architecture (the number of hidden layers, dropout rate etc.). In contrast, model parameters are learned during the training process. In the FedAvg algorithm, the weight w of the machine learning model is learned during the training process.

Let us formulate the hyperparameter optimization task: Let $\Lambda \subset \mathbb{R}^N$ denote the hyperparameter search space and let $\lambda \in \Lambda$ denote a hyperparameter configuration. We wish to find $\lambda^* \in \Lambda$ such that

$$\lambda^* = \arg\min_{\lambda \in \Lambda}[\mathcal{L}(\mathcal{D}_{\text{valid}}, \mathcal{A}_\lambda(\mathcal{D}_{\text{train}}))]$$

where $\mathcal{L}$ is defined as the loss function, $\mathcal{D}_{\text{valid}}$ is the validation dataset, $\mathcal{D}_{\text{train}}$ is the training dataset, and $\mathcal{A}_\lambda$ is the machine learning algorithm trained on dataset $\mathcal{D}_{\text{train}}$ with hyperparameter configuration λ. We train the algorithm $\mathcal{A}_\lambda$ on the training dataset $\mathcal{D}_{\text{train}}$ with hyperparameter configuration λ and evaluate its performance by computing the loss of the resulting model on the validation dataset $\mathcal{D}_{\text{valid}}$. Finally, we select the hyperparameter configuration that yields the minimal loss.

Hyperparameter search in a knowledge-aggregation algorithm is a challenging optimization problem. We do not have access to the gradient of the loss function with respect to the hyperparameter configuration and cannot apply gradient methods. Also, since we have to train the model on the training dataset and then evaluate its performance on the validation dataset, evaluation costs can be extremely expensive. Furthermore, we will typically find a mixture of different variable types in a hyperparameter configuration $\lambda = \lambda_1 \times \cdots \times \lambda_N$. A hyperparameter can be a discrete variable, $\lambda_i \in \mathbb{N}$, such as the number of hidden layers, a continuous variable, $\lambda_i \in \mathbb{R}$, such as the learning rate, and a categorical variable, such as the choice of the optimizer. Moreover, a hyperparameter λ_i can depend on other hyperparameters, such as hyperparamters regarding the model architecture.

In this work, we consider the following hyperparameters regarding the FedAvg algorithm 2.2:

- the fraction C of clients that perform computation,
- the number R of communication rounds,
- the number E of epochs,
- the batch size B,
- the learning rate η,
- the number m of hidden layers in the machine learning model,
- the dropout rate d, and
- the weight γ of the loss function.

We note that the number of epochs E is crucial for reducing the risk of overfitting, the effect of fitting the model too closely to the training dataset $\mathcal{D}_{\text{train}}$. Since we train the model on the training dataset, the loss decreases on the training dataset. However, in the case of overfitting, the model does not perform well on the validation dataset, the unknown data to the model. Therefore, we specify an arbitrarily large number E and apply *early stopping*, a method that stops the learning process when the loss is not decreasing on the validation dataset for a certain number of epochs. In FL, the weights of the global model are updated in each communication round on the server by aggregating the weights of the local models. To ensure that the global model performs well, it is therefore particularly important to avoid overfitting of the local models. We consider another hyperparameter for *early stopping*: the number of communication rounds R. In IFL, we expect that the performance of the global model improves with an increasing number of communication rounds R. Therefore, we choose a sufficiently large number of communication rounds and apply *early stopping*. We note that we have restricted this assumption to an IFL setting. In IFL, we limit the knowledge exchange to clients that have sufficiently similar data and therefore prevent negative knowledge transfer. However, in a FL setting, we cannot assume that the data is sufficiently similar. Thus, the performance of the global model might see a significant decrease with an increasing number of communication rounds [12].

2.3.1 Hyperparameter Optimization Algorithms

In this section, we want to present the most common approaches to hyperparameter optimization in machine learning. All of these approaches address challenges of hyperparameter search in knowledge-aggregation algorithms, e.g. no access to the gradient. Therefore, each hyperparameter optimization algorithm selects a hyperparameter configuration in the search space, performs the learning process on the training dataset and evaluates the model's performance on a validation dataset. However, the difference lies in the way of selecting the next hyperparameter for evaluation.

We assume that we wish to optimize the hyperparameters λ_i for $i = 1, \ldots, N$. Let Λ_i denote the search space for hyperparameter λ_i and let $\lambda := \lambda_1 \times \cdots \times \lambda_N$ denote a hyperparameter configuration in the search space $\Lambda := \Lambda_1 \times \cdots \times \Lambda_N$.

Grid search

The simplest and most obvious hyperparameter optimization approach is grid search. We manually specify for each hyperparameter λ_i a discrete subset $G_i \subset \Lambda_i$, form a grid $G := G_1 \times \cdots \times G_N \subset \Lambda$ and evaluate each hyperparameter configuration $\lambda \in G$. Grid search suffers from the *curse of dimensionality*: the number of evaluations $\prod_{i=1}^{N} |G_i|$ grows exponentially and the efficacy decreases with an increasing number of hyperparameters. However, it is optimal for optimizing a hyperparameter configuration $\lambda = \lambda_1 \times \cdots \times \lambda_N$ where N is sufficiently small. In figure 2.1, the grid search algorithm is illustrated. Assuming $N := 2$, we wish to optimize two hyperparameters $\lambda_i, i = 1, 2$. Therefore, we evaluate each configuration $\lambda := \lambda_1 \times \lambda_2$ in that grid.

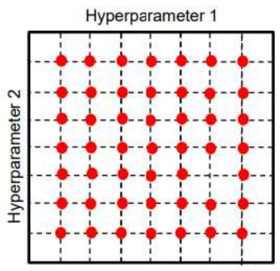

Figure 2.1: Example of grid search in two dimensions

Random search

A more advanced approach is random search. We manually specify a range $R_i \subseteq \Lambda_i$ for each hyperparameter λ_i and then randomly sample a hyperparameter configuration $\lambda \in R := R_1 \times \cdots \times R_N$. Instead of evaluating each hyperparameter configuration λ in a grid, it executes random searches in the search space R. In figure 2.2, the random search algorithm is illustrated. Again, we wish to optimize two hyperparameters $\lambda_i, i = 1, 2$. Now, we randomly select a hyperparameter configuration $\lambda := \lambda_1 \times \lambda_2$ in the search space R.

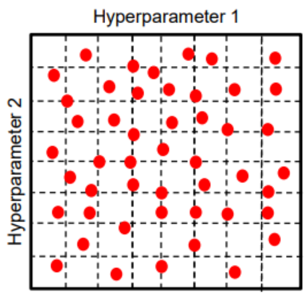

Figure 2.2: Example of random search in two dimensions

Random search addresses the inefficiency of grid search in high-dimensional search spaces. It does not suffer of the *curse of dimensionality* and solves the problem of a *low effective dimensionality*, a hyperparameter λ_i being less sensitive to changes than another [7]. Figure 2.3 illustrates the grid search and the random search algorithm when dealing with a low effective hyperparameter. We wish to optimize $f(\lambda_1, \lambda_2) := g(\lambda_1) + h(\lambda_2)$ for two hyperparameters $\lambda_i, i = 1, 2$. We assume that λ_1 (the important parameter) is more sensitive to changes and has more impact on the performance of the machine learning model f, and λ_2 (the unimportant parameter) is almost insensitive to changes. Therefore, we have $f(\lambda_1, \lambda_2) \approx g(\lambda_1)$. Above each square $g(\lambda_1)$ is shown in green, and left of each square $h(\lambda_2)$ is shown in yellow. In the optimization algorithm, we evaluate $f(\lambda_1, \lambda_2)$ for 9 different hyperparameter configurations $\lambda := \lambda_1 \times \lambda_2$. While grid search evaluates $g(\lambda_1)$ only for 3 distinct values λ_1, random search evaluates $g(\lambda_1)$ for 9 distinct values λ_1.

Bayesian Optimization

The benefits of grid search and random search lie in their simplicity. However, we would like to use past configurations to determine the next configuration in the hyperparameter search space. While grid search and random search do not profit of previous configurations, Bayesian optimization includes past configurations in the decision of choosing the next configuration. Therefore, it avoids unnecessary evaluations and requires fewer iterations to find the best hyperparameter configuration [6]. Bayesian optimization is a sequential search

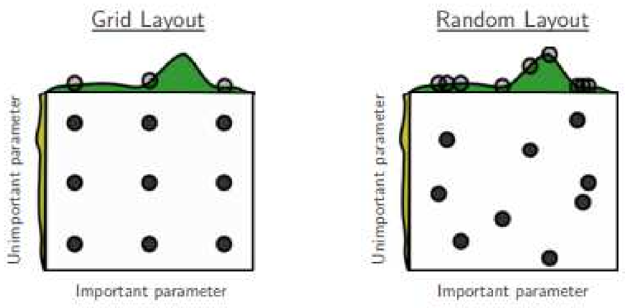

Figure 2.3: Example of grid search and random search in two dimensions with a low effective dimension [7]

framework [11] that balances exploitation and exploration of the search space. The idea is to exploit our knowledge by investigating more closely promising regions. Simultaneously, we want to explore regions of the search space with high uncertainty ensuring we do not miss the global optimum. An overuse of exploitation put us at risk of getting caught in a local optimum, the opposite of leaving the global optimum.

In Bayesian optimization, we construct a posterior distribution of functions that approximates the objective function. A widely used choice are Gaussian processes. We call a stochastic process Gaussian if and only if all finite sub-collections of random variables have a multivariate Gaussian distribution. Similarly to a Gaussian distribution, a Gaussian process $f(x) \sim GP(m(x), k(x,x'))$ is completely defined by a mean function $m : \mathcal{X} \to \mathbb{R}$ and a covariance function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ [13].

A widely used covariance function is the exponential square function

$$k(x,x') := \exp(-\frac{1}{2}\|x-x'\|_2^2)$$

where $\|\cdot\|_2$ denotes the Euclidean norm. By choosing the covariance function, we can make assumptions about the objective function.

Let $X := \{x_i | i = 1, \dots, n\}$ be a set of n training points and let

$$K(X,X) := \begin{pmatrix} k(x_1,x_1) & k(x_1,x_2) & \dots & k(x_1,x_n) \\ k(x_2,x_1) & k(x_2,x_2) & \dots & k(x_2,x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n,x_1) & k(x_n,x_2) & \dots & k(x_n,x_n) \end{pmatrix}$$

denote the $n \times n$ covariance matrix. Furthermore, let X_* denote the set of n_* test points, $K(X_*, X_*)$ denote the $n_* \times n_*$ covariance matrix, and $K(X, X_*)$ denote the $n \times n_*$ matrix of covariances for all pairs of training and test points. Analogously, we define $K(X_*, X)$.

Then, the joint distribution of the training outputs f and the test outputs f_* is given by

$$\begin{bmatrix} f \\ f_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{pmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{pmatrix}\right).$$

We obtain the posterior distribution of functions by conditioning the joint Gaussian distribution [13]

$$f_*|X_*, X, f \sim \mathcal{N}\Big(K(X_*,X)K(X,X)^{-1}f, K(X_*,X_*) - K(X_*,X)K(X,X)^{-1}K(X,X_*)\Big).$$

Then, the posterior distribution is used to determine the next sample point. We choose a sample point according to a criterion. The criterion is represented by an so-called acquisition function. Therefore, in Bayesian optimization, we transform the optimization problem into a cheaper proxy optimization problem, optimizing the acquisition function. The acquisition function balances exploitation and exploration indicating the utility of sample points. We choose the sample point with the highest utility for the next evaluation. Based on the acquisition function, the utility is high for sample points in promising regions (exploitation) and for sample points in unexplored regions (exploration).

Now, we can formulate the Bayesian optimization algorithm [14]:

1. We initialize a Gaussian process.
2. We find a sample point that maximizes the utility of the acquisition function based on the current prior distribution.
3. We evaluate the objective function for the chosen sample point.
4. We update the Gaussian process and obtain the posterior distribution.

We repeat step 2, 3 and 4 for multiple iterations.

2.4 Hyperparameter Optimization Approaches in an IFL System

In a FL setting, hyperparameter optimization poses new challenges and is a major open research area. In traditional machine learning approaches, hyperparameter optimization often only focuses on the improvement of model accuracy rather than communication efficacy, and computing efficacy for each client [3]. However, in section 2.1.2, we have linked the performance of a machine learning model in FL to communication costs. Therefore, Kairouz et al. [3] propose that further research in FL should consider efficient hyperparameter optimization methods.

Kairouz et al. [3] propose an approach to tackle this challenge. The introduction of potential additional hyperparameters (the number of communication rounds, the number of participating clients per communication round etc.) motivates a closer analysis of the hyperparameter space in the context of FL. Kairouz et al. [3] therefore introduce the idea of a separate optimization of hyperparameters. We now want to further investigate this approach.

In an effort to reduce communication costs, a critical bottleneck in FL, we investigate a communication efficient hyperparameter optimization approach. We believe that the data distribution influences the choice of the best hyperparameter configuration and suggest that the best hyperparameter configuration for a client might differ from another client based on individual data properties. Therefore, we want to investigate a local hyperparameter optimization approach that – in contrast to a global hyperparameter optimization approach – allows every client to have its own hyperparameter configuration. The local approach allows us to optimize hyperparameters prior to the federation process reducing communication costs.

Dai et al. [4] investigated a communication efficient local hyperparameter optimization approach and introduced Federated Bayesian Optimization (FBO) extending Bayesian optimization to the FL setting. In FBO, every client locally uses Bayesian optimization to find the optimal hyperparameter configuration. Additionally, each client is allowed to request for information from other clients. Dai et al. [4] proved a convergence guarantee for this algorithm and its robustness against heterogeneity. However, until now, there is no research about the impact of global and local hyperparameter optimization.

In the LocalHPO algorithm 2, we perform local hyperparameter optimization. We optimize the hyperparameter configuration λ^k for each client k. In the GlobalHPO algorithm 3, we perform global hyperparameter optimization. We optimize the hyperparameter configuration λ in the federation process. The LocalOptimization method in the LocalHPO algorithm 2 and the GlobalOptimization method in the GlobalHPO algorithm 3 can be based on any hyperparameter optimization algorithm.

Algorithm 2: LocalHPO

```
Server executes:
initialize $w_0$
for each client $k = 1, \dots, K$ do
    $\lambda^k :=$ LocalOptimization$(k, w_0)$
end
return $(\lambda^k)_{k=1}^{K}$
```

Algorithm 3: GlobalHPO

```
Server executes:
$\lambda :=$ GlobalOptimization()
return $\lambda$
```

We want to differentiate between a hyperparameter λ_i that is globally constant, assuming that all clients have the same value for λ_i, and a hyperparameter λ_i^k that is not globally constant – due to heterogeneity in the data – and whose value depends on a client k. Here, λ_i^k denotes the hyperparameter λ_i on client k. Let λ^k denote the hyperparameter configuration on client k. This motivates the following definition: We call a hyperparameter λ_i

- global, if $\lambda_i^k = \lambda_i^l$ for all $k, l \in \{1, \dots, K\}$,
- local, if there exist $k, l \in \{1, \dots, K\}$ with $k \neq l$ such that $\lambda_i^k \neq \lambda_i^l$.

We notice that this differentiation is only relevant for settings with non-identically distributed clients. In an identically distributed setting, we assume that a hyperparameter configuration that works for one client also works for another client. In our experiments, we verified this assumption for a proxy dataset.

In addition to the classification into local and global hyperparameters, we can distinguish between model hyperparameters, parameters that concern the machine learning model, and federation hyperparameters, parameters that concern the federation process. By definition, a local hyperparameter must be a model hyperparameter. A global hyperparameter, however, can either be a model hyperparameter or a federation hyperparameter. While a federation hyperparameter can only be optimized in the federation process, a model hyperparameter can be optimized locally on each client prior to the federation process. For example, we classify the hyperparameters we considered in section 2.3 in the following way:

C	fraction of participating clients	federation/global
R	number of communication rounds	federation/global
E	number of epochs	model/local
B	batch size	model/local
η	learning rate	model/local
m	number of hidden layers	model/local
d	dropout rate	model/global
γ	weights of loss function	model/local

Table 2.1: Example of a classification of hyperparameters in a non-identically distributed FL setting

3 Data, Algorithms and Experiments

In the next section, we want to make our benchmark design explicit and present our experimental setup. We will present the machine learning tasks including the data partition of the training data, the machine learning models, the optimization algorithms and our experiments. We considered an image classification task and an IoT sensor based anomaly classification task on industrial assets. For the image classification task we have chosen a proxy dataset, the MNIST dataset of handwritten digits. Finally, to demonstrate the effectiveness of the IFL system, we evaluated a real-world problem with a natural partition of data.

3.1 Data

The MNIST dataset consists of handwritten digits and has 60 000 training examples and 10 000 test examples. In order to test the IFL system on this proxy dataset, we still need to specify on how to distribute the data over artificially designed clients. To systematically evaluate the effectiveness of the IFL system and the cohort strategy, we simulated an identical data distribution. This refers to shuffling the data and partitioning the data into 10 clients, each receiving 6 000 examples.

Following the approach of McMahan et al. [1], we applied a convolutional neural network with the following settings:

- 2 convolutional layers with 32 and 64 filters of size 5×5 and a ReLu activation function, each followed by a max pooling layer of size 2×2,
- a dense layer with 512 neurons and a ReLu activation function,
- a dense layer with 10 neurons and a softmax activation function.

The industrial task concerns IoT sensor based anomaly classification on industrial assets. Figure 3.1 illustrates the pump and sensor setup. We considered multiple centrifugal pumps with sensors placed at different positions, in different directions to record three axis vibrational data in a frequency of 6644 Hz. Per minute, 512 samples were collected. The task is the classification of the operating condition of the pump. Therefore, we operated the pumps under 6 varying conditions, including 3 healthy states and 3 anomalous states. More precisely, the measurements include the following operating conditions: the healthy data with full load (50 $\mathrm{m}^3\,\mathrm{h}^{-1}$), the healthy data with partial load ($37.5, 25$ and 12.5 $\mathrm{m}^3\,\mathrm{h}^{-1}$), the healthy data measured in idle state (0 $\mathrm{m}^3\,\mathrm{h}^{-1}$), the anomalous data measured during hydraulic blocking of the pump, the anomalous data measured during dry running of the pump, and the anomalous data measured during cavitation of the pump. In our experiments, we only consider the data measured by sensor 3. A client is either assigned data of an asset in a measurement, or data of several assets in a measurement ensuring that each client sees all operating conditions. However, since in the process of measurement, the assets were turned off and cooled down overnight, the sensors were removed and reattached, the screws were removed an reattached to the asset, and the assets were completely dismantled and rebuilt, we consider the data to be non-identically distributed regarding its feature distribution.

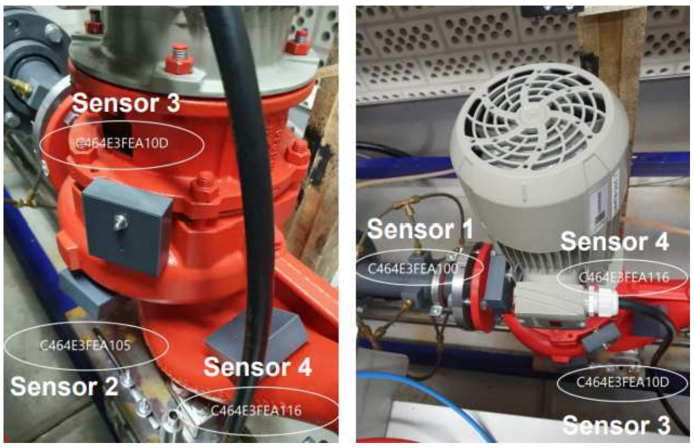

Figure 3.1: Pump and sensor setup

To increase robustness and rotational invariance of the machine learning model, we remapped the 3-dimensional vibrational data from the sensor coordinate system s_i into the world coordinate system w_i using the Kabsch algorithm [15] minimizing the following loss function L for a given rotation matrix C:

$$L(C) = \sum_{i=1}^{n} \|s_i - Cw_i\|_2^2.$$

Then, we applied a sliding window such that the remapped 1×512 input is artificially increased to a 16×256 input with window size of 256 and an overlap of 16 steps. Further, we extracted the Melfrequency cepstral coefficients (MFCCs) and applied the synthetic minority oversampling technique (SMOTE). Finally, we normalized the resulting MFCCs features between $[-1, 1]$ based on a transformation resulting in a gaussian distribution.

We applied an artificial neural network with the following settings:

– a dense layer with 64 neurons and a ReLu activation function,

– a dropout layer with a dropout rate of 0.4,

– a dense layer with 6 neurons and a ReLu activation function,

– a dropout layer with a dropout rate of 0.4,

– a softmax activation function.

3.2 Algorithms

Our evaluations include the FedAvg algorithm 1, and the hyperparameter optimization approaches LocalHPO 2 and GlobalHPO 3 presented in section 2.4. We implemented these approaches based on grid search and Bayesian optimization. In this section, we give their pseudocode. We searched for the learning rate η with fixed C, R, E, and B.

In algorithm 4, we give the pseudocode of the LocalOptimization method in LocalHPO 2 based on the grid search algorithm with a fixed grid G. We iterate through the grid G, train the model on the training data of client k based on the ClientUpdate method used in the FedAvg algorithm 1 with the learning rate η as an additional argument, and validate the performance of the model w_η on the validation data $\mathcal{D}^k_{\text{valid}}$ of client k. Finally, the learning rate that yields the highest accuracy A_η on the validation data is selected. Here, w_η denotes the resulting model trained on the training data with learning rate η and $A(\mathcal{D}^k_{\text{valid}}, w_\eta)$ denotes the accuracy of the model tested on the validation data $\mathcal{D}^k_{\text{valid}}$ of client k.

Algorithm 4: Local Grid Search

```
LocalOptimization($k, w_0$) :
for each learning rate $\eta \in G$ do
    $w_\eta :=$ ClientUpdate($k, w_0, \eta$)
    $A_\eta := A(\mathcal{D}^k_{\text{valid}}, w_\eta)$
end
$\eta^*_k := \arg\max_{\eta \in G} A_\eta$
return $\eta^*_k$
```

In algorithm 5, we give the pseudocode of the GlobalOptimization method in GlobalHPO 3 based on the grid search algorithm with a fixed grid G. We iterate through the grid, perform the FedAvg algorithm 1 with the learning rate η as an additional argument, validate the performance of the model w_η on the validation data $\mathcal{D}^k_{\text{valid}}$ for all clients $k = 1, \dots, K$ and compute the average accuracy of all clients. Finally, the learning rate that yields the highest average accuracy A_η is selected.

In algorithm 6, we give the pseudocode of the LocalOptimization method in LocalHPO 2 based on Bayesian optimization. The objective function f takes the learning rate η as an argument, trains the model on the training data of client k based on the ClientUpdate method used in the FedAvg algorithm 1 with the learning rate η as an additional argument, validates the performance of the model w on the validation data $\mathcal{D}^k_{\text{valid}}$ of client k, and returns the resulting accuracy. We initialize a gaussian process GP for the objective function f with n_{init} sample points. Then, we find the next sample point $\eta_{n_{\text{init}}+i}$ by maximizing the acquisition function, evaluate $f(\eta_{n_{\text{init}}+i})$, and update the gaussian process GP. Finally, we select the learning rate η^* that yields the highest average accuracy. We repeat this for n_{iter} iterations.

Algorithm 5: Global Grid Search

```
GlobalOptimization() :
for each learning rate $\eta \in G$ do
    $w_\eta :=$ FederatedAveraging($\eta$)
    for each client $k = 1, \dots, K$ do
        $A^k_\eta := A(\mathcal{D}^k_{\text{valid}}, w_\eta)$
    end
    $A_\eta := \frac{1}{K} \sum_{k=1}^{K} A^k_\eta$
end
$\eta^* := \arg\max_{\eta \in G} A_\eta$
return $\eta^*$
```

Algorithm 6: Local Bayesian Optimization

```
LocalOptimization($k, w_0$) :
initialize a gaussian process $GP$ for $f$
evaluate $f$ at $n_{\text{init}}$ initial points
for $i = 1, \dots, n_{\text{iter}}$ do
    find sample point $\eta_{n_{\text{init}}+i}$ that maximizes acquisition function
    evaluate objective function $f$ at $\eta_{n_{\text{init}}+i}$
    update the gaussian process $GP$
end
$\eta^* := \arg\max_{i=1,\dots,n_{\text{init}}+n_{\text{iter}}} f(\eta_i)$
return $\eta^*$
objective function:
$f(\eta)$ :
$w :=$ ClientUpdate($k, w_0, \eta$)
$A := A(\mathcal{D}^k_{\text{valid}}, w)$
return $A$
```

In algorithm 7, we give the pseudocode of the GlobalOptimization method in GlobalHPO 3 based on Bayesian optimization. The objective function f takes the learning rate η as an argument, performs the FedAvg algorithm 1 with the learning rate η as an additional argument, validates the performance of the model w on the validation data $\mathcal{D}^k_{\text{valid}}$ for all clients $k = 1, \dots, K$, computes the average accuracy of all clients and returns the resulting accuracy. We initialize a gaussian process GP for the objective function f with n_{init} sample points. Then, we find the next sample point $\eta_{n_{\text{init}}+i}$ by maximizing the acquisition function, evaluate $f(\eta_{n_{\text{init}}+i})$, and update the gaussian process GP. Finally, we select the learning rate η^* that yields the highest average accuracy. We repeat this for n_{iter} iterations.

Algorithm 7: Global Bayesian Optimization

```
GlobalOptimization() :
initialize a gaussian process $GP$ for $f$
evaluate $f$ at $n_{\text{init}}$ initial points
**for** $i = 1, \dots, n_{\text{iter}}$ **do**
    find sample point $\eta_{n_{\text{init}}+i}$ that maximizes acquisition function
    evaluate objective function $f$ at $\eta_{n_{\text{init}}+i}$
    update the gaussian process $GP$
**end**
$\eta^* := \underset{i=1,\dots,n_{\text{init}}+n_{\text{iter}}}{\arg\max} f(\eta_i)$
return $\eta^*$
**objective function:**
$f(\eta)$ :
$w :=$ FederatedAveraging($\eta$)
**for** each client $k = 1, \dots, K$ **do**
    $A^k := A(\mathcal{D}^k_{\text{valid}}, w)$
**end**
$A := \frac{1}{K} \sum_{k=1}^{K} A^k$
return A
```

3.3 Experiments

First, we want to demonstrate the effectiveness of the IFL system for the industrial machine learning task. The cohort design of the IFL system aims at partitioning the available clients into cohorts and restricting knowledge exchange only to clients within a cohort, and thus to clients that have sufficiently similar data. Therefore, we partition the clients into cohorts using the elbow method and k-means clustering algorithm based on the feature distribution.

The cohort algorithm partitioned our 9 clients into 3 cohorts. Cohort 0 includes 4 clients with data generated by pump 1 and one client with data collaboratively generated by the pumps $1, 2, 3$. Cohort 1 includes 2 clients with data generated by pump 4 and one client with data collaboratively generated by the pumps $1, 2, 3$. Cohort 2 only consists of one client. We consider the data of this client highly non-identically distributed regarding its feature distribution due to its not-standardized measurement protocol. For the training, we set $R = 20$, $C = 1$, $E = 5$, and $B = 128$ in the IFL system. In order to evaluate the learning approaches in a direct comparison, we chose the number of epochs E in the individual and central learning approach as $E = E_{\text{fed}}R$ where E_{fed} is the number of epochs in the federated learning approach and R is the number of communication rounds. In figure 3.2, the colors indicate the according cohort. Figure 3.2 shows the test accuracy on the central cohort test data for each client, for i) a model trained on the individual training data of the client ii) a central model trained on the collected training data of all clients in the cohort iii) the federated model trained in the cohort. We observe that, in general, the IFL approach performs better than the individual learning approach and approximates the central learning approach. On client 6, the individual learning achieves a slightly better accuracy than the federated learning, and all models perform well on client 8.

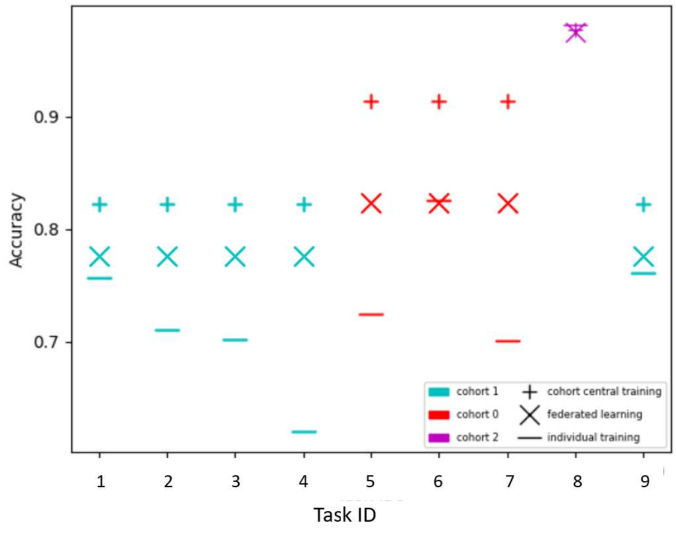

Figure 3.2: Comparison of individual learning, central learning and federated learning on the industrial dataset

Then, to systematically investigate the impact of global and local hyperparameter optimization, we want to compare the global and local hyperparameter optimization approach in an identically distributed FL setting, the MNIST machine learning task, as well as in a non-identically distributed FL setting, the industrial task. Therefore, we implemented the global and local optimization approach based on grid search with a grid $G := [0.0001, 0.001, 0.01, 0.1]$, and based on Bayesian optimization with the widely used squared exponential kernel and the upper confidence bound acquisition function. We searched for the learning rate η with fixed R, C, E and B.

In order to evaluate the global and local optimization approaches in a direct comparison, we chose the number of epochs E in the local optimization approach as $E = E_{\text{global}}R$ where E_{global} is the number of epochs in the global optimization approach and R is the number of communication rounds. In the global optimization task, we set $R = 10$, $C = 1$, $E = 1$ and $B = 128$ for the MNIST data, and $R = 10$, $C = 1$, $E = 5$ and $B = 128$ for the industrial data. In the local optimization task, we set $E = 10$ and $B = 128$ for the MNIST data, and $E = 50$ and $B = 128$ for the industrial data. For the evaluation of the global hyperparameter optimization approach, we optimized the learning rate using the global approach, trained the federated model with a global learning rate, and tested the resulting federated model on the cohort test data. Then, we optimized the learning rate using the local approach, trained the federated model with local individual learning rates for each client in the cohort, and tested the resulting federated model on the cohort test data.

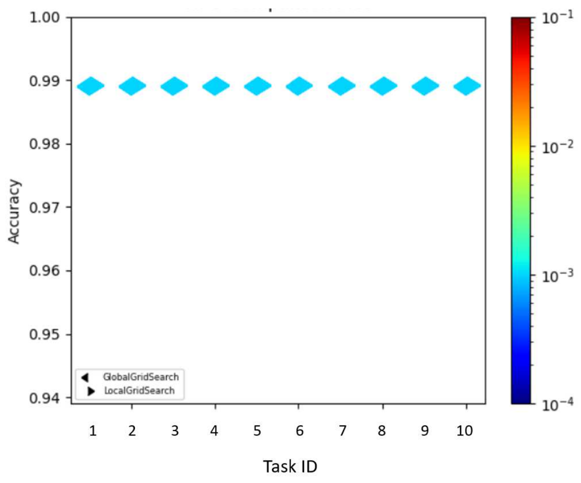

Figure 3.3: Comparison of the optimization approaches based on grid search for the MNIST task

Figure 3.3 shows the results for the MNIST data. The optimization approaches are based on the grid search algorithm. For the training posterior to the optimization, we set $R = 10$, $C = 1$, $E = 1$, and $B = 128$ in the IFL system. The colour indicates the optimized learning rate on the corresponding client. Since the MNIST data is identically distributed, there is only one cohort and all clients have the same federated model and thus the same test accuracy. Our results show that the grid search algorithm selected 10^{-3} in the local optimization of the learning rate on each client. According to our expectation, the global optimization approach yielded the same learning rate.

For the industrial task, we evaluated the global and local optimization approach based on grid search and Bayesian optimization. For the training posterior to the optimization, we set $R = 20$, $C = 1$, $E = 5$, and $B = 128$ in the IFL system. Figure 3.4 shows the results for the industrial data with the optimization approaches based on the grid search algorithm. The results show that, in all cohorts, the global approach yielded an equal or larger accuracy than the local approach.

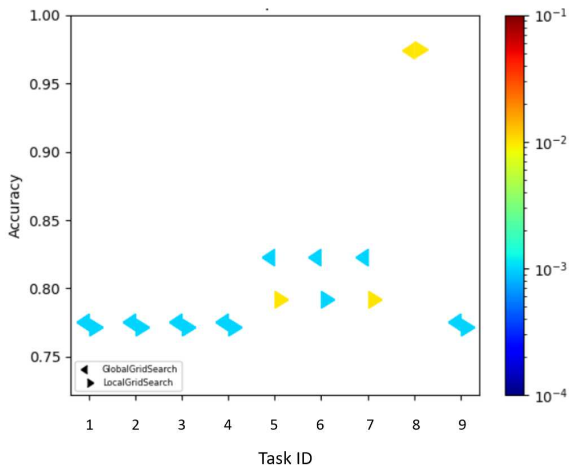

Figure 3.4: Comparison of the optimization approaches based on grid search for the industrial task

Figure 3.5 shows the results for the industrial data with the optimization approaches based on the Bayesian algorithm. Note that the search space of the learning rate was $[10^{-4}, 10^{-1}]$ in the optimization while the scale in the plot starts from 10^{-3}. The results show that the global approach yielded a larger accuracy than the local approach in cohort 0 and cohort 1. In cohort 2, however, the local optimization approach resulted in a learning rate of 0.0114 and yielded a test accuracy of 0.9736, and the global optimization approach resulted in a learning rate of 0.0976 and yielded a test accuracy of 0.3867.

In order to compare the optimization approaches for the industrial task, we performed a paired t-test regarding the test accuracy to determine the statistical significance, see table 3.1. We observe that the global optimization approach is significantly better than the local approach, both for the grid search approach ($p = 0.028$) and for the Bayesian

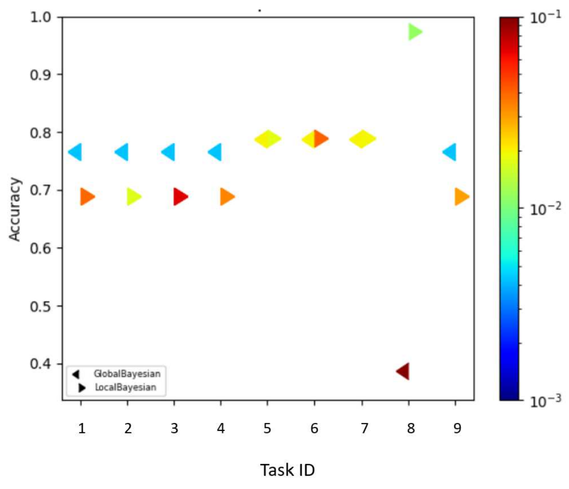

Figure 3.5: Comparison of the optimization approaches based on Bayesian optimization for the industrial task

approach ($p = 0.012$). Furthermore, the results show that the grid search approach is significantly better than the Bayesian approach, both for the global approach ($p = 0.004$) and for the local approach ($p = 0.008$). Note that we considered cohort 2 an outlier and excluded this cohort from our calculations. Cohort 2 solely consists of client 8, a client whose data was not generated according to the standard measurement protocol. Without outlier removal, the global grid search approach is still significantly better than the local grid search approach ($p = 0.032$), and the local grid search approach is significantly better than the local Bayesian approach ($p = 0.010$). However, there is no significant difference in the global Bayesian approach vs. the local Bayesian approach ($p = 0.755$) and in the global grid search approach vs. the global Bayesian approach ($p = 0.230$).

client	global grid	local grid	global Bayesian	local Bayesian
1	0.7756	0.7720	0.7659	0.6897
2	0.7756	0.7720	0.7659	0.6897
3	0.7756	0.7720	0.7659	0.6897
4	0.7756	0.7720	0.7659	0.6897
5	0.8230	0.7921	0.7882	0.7889
6	0.8230	0.7921	0.7882	0.7889
7	0.8230	0.7921	0.7882	0.7889
8	0.9740	0.9749	0.3867	0.9736
9	0.7756	0.7720	0.7659	0.6897

Table 3.1: Test accuracy of federated model on central cohort test data posterior to corresponding optimization approach and training

4 Conclusion

The field of machine learning is constantly evolving and spreading into a growing application sector driven by a wealth of available data. However, this data is often privacy sensitive and therefore unsuitable for a central machine learning approach. McMahan et al. [1] introduced FL that does not collect the data nor allow a server to access the data. However, Zhao et al. [12] proved that heterogeneity significantly reduces the accuracy of FL. Therefore, Hiessl et al. [8] introduced IFL that only orchestrates knowledge exchange between clients that have sufficiently similar data. We demonstrated the effectiveness of the IFL system for an IoT sensor based classification task on industrial assets. Based on the cohort algorithm, the IFL system created 3 cohorts for the 9 available clients. We observed that the cohort building prevents negative knowledge transfer and the results show that the federated learning approach approximates the central learning approach, while outperforming individual learning of the clients.

Hyperparameter selection is a crucial task in the optimization of knowledge-aggregation algorithms. In a FL setting, hyperparameter optimization poses new challenges and is a major open research area. Until now, there is no research about the impact of global and local hyperparameter optimization of a FL task with heterogeneous clients. In this work, we investigated the impact of global and local optimization approaches in an IFL System based on a proxy dataset and a real-world problem. In our experiments on the industrial data, local optimization yielded different learning rates on different clients in a cohort. However, the results show that a globally optimized learning rate, and thus, a global learning rate for all clients in a cohort improves the performance of the resulting federated model. Therefore, we conclude that the global optimization approach outperforms the local optimization approach. Consequently, we have to deal with a communication-performance trade-off in the hyperparameter optimization in FL. The local optimization approach allows us to reduce communication costs but is outperformed by the global optimization approach. In our experiments on the proxy dataset, however, the local approach achieved the same performance as the global approach. We implemented the optimization approaches based on the grid search algorithm and Bayesian optimization. The results show that the grid search approaches outperform the Bayesian approaches, both globally and locally. However, we only considered one hyperparameter in our optimization task. So, it would be interesting to explore whether we can confirm these observations for a hyperparameter configuration of more hyperparameters. Also, in Bayesian optimization, we only considered the upper confidence bound acquisition function and only run the algorithm for 8 iterations.

List of Figures

Bibliography

[1] H. Brendon McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. *Communication-Efficient Learning of Deep Networks from Decentralized Data.* 2016. arXiv: 1602.05629

[2] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. *Federated Learning.* 2019. Morgan & Claypool Publishers. pp. 1-6, 55-60

[3] Peter Kairouz, H. Brendon McMahan, Brendan Avent et al. *Advances and Open Problems in Federated Learning.* 2019. arXiv: 1912.04977

[4] Zhongxiang Dai, Bryan Kian Hsiang Low, and Patrick Jaillet. *Federated Bayesian Optimization via Thompson Sampling.* 2020. arXiv: 2010.10154

[5] Adrian Nilsson, Simon Smith, Gregor Ulm, Emil Gustavsson, and Mats Jirstrand. *A Performance Evaluation of Federated Learning Algorithms.* 2018. Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning.

[6] Li Yang, and Abdallah Shami. *On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice.* 2020. arXiv: 2007.15745

[7] James Bergstra, Yoshua Bengio. *Random Search for Hyper-Parameter Optimization.* 2012. The Journal of Machine Learning Research. pp. 281-284

[8] Thomas Hiessl, Daniel Schall, Jana Kemnitz, and Stefan Schulte. *Industrial Federated Learning – Requirements and System Design.* 2020. arXiv: 2005.06850

[9] Huy Phan, Martin Krawczyk-Becker, Timo Gerkmann, and Alfred Mertins. *DNN and CNN with Weighted and Multi-Task Loss Functions for Audio Event Detection.* 2017. arXiv: 1708.03211

[10] Yaoshiang Ho, Samuel Wookey. *The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling* 2020. arXiv: 2001.00570

[11] M.O. Ahmed, and S. Prince. *Bayesian Optimization.* 2020. Borealis AI. Available: https://www.borealisai.com/en/blog/tutorial-8-bayesian-optimization/

[12] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. *Federated Learning with Non-IID Data.* 2018. arXiv: 1806.00582

[13] C.E. Rasmussen, K.I. Williams. *Gaussian Processes for Machine Learning.* 2006. MIT Press. pp. 13-16

[14] Andre Ye. *The Beauty of Bayesian Optimization*. 2020. Towards Data Science. Available: https://towardsdatascience.com/the-beauty-of-bayesian-optimization-explained-in-simple-terms-81f3ee13b10f

[15] F.L. Markley. *Attitude Determination Using Vector Observation: A Fast Optimal Matrix Algorithm*. 1993. Journal of the Astronautical Sciences. pp. 261–280