Informatics

A systematic evaluation of federated learning algorithms in the context of industrial applications

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Data Science

eingereicht von

Viktorija Pruckovskaja, MSc
Matrikelnummer 11933811

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Mag.rer.soc.oec. Dr.rer.soc.oec. Ivona Brandić
Mitwirkung: M.Tech. PhD Shashikant Shankar Ilager

Wien, 16. Jänner 2023

Viktorija Pruckovskaja Ivona Brandić

Informatics

A systematic evaluation of federated learning algorithms in the context of industrial applications

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Data Science

by

Viktorija Pruckovskaja, MSc
Registration Number 11933811

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Mag.rer.soc.oec. Dr.rer.soc.oec. Ivona Brandić
Assistance: M.Tech. PhD Shashikant Shankar Ilager

Vienna, 16th January, 2023

Viktorija Pruckovskaja	Ivona Brandić

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Viktorija Pruckovskaja, MSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 16. Jänner 2023

Viktorija Pruckovskaja

Acknowledgements

I would like to express my sincere gratitude to everybody who made this thesis possible.

First, I would like to thank my thesis advisors, Ivona Brandić and Shashikant Shankar Ilager, for their guidance, constructive feedback, and also their high level of responsiveness, which was a great support in the completion of this thesis. Moreover, I would like to express my gratitude to numerous people at the Austrian Institute of Technology for the friendly collaboration and working atmosphere, which not only contributed a lot to the thesis but also was an incredible personal experience. Especially, many thanks to Bernhard Haslhofer and Clemens Heistracher for guiding me in the ideation of the thesis topic and creation of the proposal, but also for providing their valuable input throughout the whole time till the finalization of the thesis. Finally, I would like to thank my family, closest friends, and former working colleagues for their psychological support during my studies and for kindly tolerating my eventual absence.

Kurzfassung

Machine Learning und Predictive Analytics sind ein wichtiger Teil der Entwicklungen in Industrie 4.0, inklusive ihrer Anwendung in diversen intelligenten Entscheidungsprozessen. Große Datenmengen, die oft dezentral von verschiedenen Produktionseinheiten an unterschiedlichen Standorten oder sogar von unterschiedlichen Organisationen gesammelt werden, sind eine essentielle Basis für erfolgreiche Lösungen im Bereich der künstlichen Intelligenz.Das Sammeln und Verarbeiten solcher Daten in klassischer zentraler Weise stellt neue Herausforderungen dar oder ist unter Umständen gar nicht oder nur schwer möglich.Dies erfordert neue Ansätze für die Verarbeitung der Daten und für das Trainieren von maschinellen Algorithmen.Hier kommt Federated Learning (FL) in Betracht.Bei FL Ansätzen werden globale Modelle erstellt, ohne dass die Daten zentral gesammelt werden müssen.Die Daten der einzelnen Produktionseinheiten oder Organisationen werden mit anderen Einheiten oder mit dem zentralen Server nicht geteilt.Sondern es werden nur die lokal trainierten Modelle an den zentralen Server übermittelt, wo der zentrale Server sie zur Erstellung eines einzigen globalen Modell verwendet.Dieses globale Modell wird dann an die einzelnen Einheiten übermittelt and von deren weiter eingesetzt.Die Forschung im Bereich des FL nimmt seit seiner Einführung im Jahr 2017 stetig zu.Dennoch gibt es nur wenige Belege für die Nützlichkeit von FL Methoden für Predictive Maintenance Probleme, die oft mit tabellarischen Daten und sehr ungleicher Verteilung der Daten representiert werden.In unserer Arbeit geben wir einige Einblicke in die Leistung von FL Modelle für vier verschiedene Datensätze und Datenverteilungsszenarien. Wir bewerten ausgewählte FL Techniken hinsichtlich ihrer Effektivität, der zusätzlichen Kosten, die durch die Übertragung individueller und globaler Modelle entstehen, und der Fähigkeit, eine vergleichbare Effektivität zwischen den teilnehmenden Einheiten zu erreichen, also Fairness der Modelle.Darüber hinaus zeigen wir, dass für einige ausgewählte Szenarien FL eine geeignete Alternative zu klassischen Trainingsansätzen ist.In bestimmten Fällen müssen FL Methoden jedoch noch weiter erforscht und entwickelt werden, um mit individuell trainierten Modellen mithalten zu können.

Abstract

Machine learning and predictive analytics have become an important part of the intelligent decision-making process contributing to Industry 4.0 developments. The most important ingredient to successful artificial intelligence solutions is data, which are often produced in a decentralized manner by separate production units at different locations or even organizations. Collecting and processing such data in a classic centralized manner poses new challenges or may be intractable or even not possible. This requires novel approaches for processing the data and training machine learning algorithms. Here, Federated Learning may come in handy. Federated approaches train global models without the need to collect the data centrally. Individual data of each production unit or organization does not leave that entity; while trained local models are sent to the server, which aggregates them to produce a single global model. This single global model can then be used by individual entities. Currently, the research in the field of federated learning is growing since it was first introduced in 2017. Still, there is little evidence about the usefulness of federated approaches for predictive maintenance problems, represented by tabular and very imbalanced data. In our work, we provide some insights into the performance of federated models for different datasets and data distribution scenarios. We evaluate selected federated techniques in terms of their effectiveness, additional costs imposed by transferring individual and global models, and ability to achieve comparable effectiveness across participating entities, i.e. fairness. Moreover, we show that for some selected scenarios, federated learning is a suitable alternative to classic training approaches. Nevertheless, in certain cases, federated methods still require further research and development to be able to contend with models trained on an individual basis.

Contents

CHAPTER 1

Introduction

1.1 Motivation

Machine Learning is an important development in Industry 4.0 with applications in product recommender systems, dynamic product pricing, optimal supply chain scheduling, and many others. More and more production companies already employ artificial intelligence during the production process for automated equipment health monitoring and product fault detection to ensure their product quality and reliability, and to decrease maintenance costs. Tracking the data generated by machines, production units, sensors, and intelligent factories has become an essential part of manufacturing processes. However, the relevant data are often spread among different entities/locations within the same company or are even possessed by different organizations. Thus, the processing and employment of such distributed data requires novel machine learning approaches.

Collecting everything into a central resource to perform the training centrally might be a solution, yet this usually implies high communication costs for transferring the individual data. Also, finding optimal machine learning methods for predictive maintenance and defect detection problems requires large datasets due to the rare occurrence of failures in production processes or products. Such datasets are difficult and expensive to collect for companies individually [ZLM+21]. Therefore, the cooperation of several similar companies could be beneficial for developing joint solutions. However, this would imply the necessity to share the data among the companies, which may be an unacceptable obstacle due to the competitive nature of the market. Moreover, accumulating data into one source is, in general, associated with severe privacy and security threats [RLD+18], [XLD+19]. Finally, the large scale of some modern industrial infrastructures may make such centralized training approaches intractable [HLL+20a].

Federated learning approaches are used to avoid any central accumulation of data. These methods propose training a global model without directly accessing the data stored on different entities or edge devices, called clients. In particular, clients locally model their data, without having to share their data with anybody. The clients communicate their local models to a central server, and the central server aggregates them into a global model [GHP22]. Subsequently, the global model is shared with all participating entities. McMahan et al. [MMR+17] first introduced a secure training using client text-data on mobile devices to generate a global text-model. Since then, federated learning has found many applications in the industry, such as in healthcare or production companies, due to its obvious advantage of keeping the privacy of local data private and leveraging the data availability and optimal model performance.

Both the scientific community and business widely discuss the data privacy and other benefits of federated learning. Many studies concentrate on theoretical aspects and challenges of federated learning. For example, challenges associated with data and systems heterogeneity, privacy preservation, communication costs, etc. Still, there are relatively limited research focusing on industrial predictive maintenance tasks [CLH+21]. Many studies do implement federated learning in the industry, e.g. for machinery fault diagnosis [ZLM+21] or anomaly detection in industrial Internet of Things [LGN+21]. However, they usually propose certain federated learning frameworks and mostly use the basic federated learning algorithms to establish the global models, e.g. [ZZL+21]. Furthermore, many researchers consider data heterogeneity and imbalance across clients, but very few works examine the behavior of federated learning approaches on highly imbalanced industrial datasets. Typical predictive maintenance problems are solved using data with highly imbalanced distributions between faulty and non-faulty observations with only a fraction of data constituting the faulty samples. What is more, measurements are often represented by tabular data. Though, many existing works explore the performance of federated learning for computer vision or natural language processing applications. Finally, there are numerous adaptations of the initial federated learning algorithm, which tackle its different drawbacks. Nevertheless, there are no guidelines on how to select an optimal approach, making it even more challenging to identify the practical usefulness of this innovative approach in a concrete use case for industrial problems.

Therefore, our goal is to provide researchers and industrial experts some insights on the behavior of federated learning in the industrial context for problems with highly imbalanced available data, more precisely, in predictive maintenance and production quality control tasks. We aim to systematically compare different federated learning approaches for our selected problems. Finding a good machine learning model is often time- and resource-intensive, especially in the case of federated learning, where we have to consider not only the model architecture but also a variety of possibilities for aggregation of local models into the central model. Additionally, we contrast the federated setting against local training, where all clients base their models only on their data isolated from other clients, and against centralized training, where all data are centrally collected, thus violating the data privacy requirement.

1.2 Federated Learning

Federated Learning is a relatively new field of research, originating from the work of McMahan et al [MMR+17], who suggested a method to collaboratively learn a shared model involving hundreds to millions of mobile devices and the data stored on these devices. Since the local data on mobile devices are usually privacy-sensitive, the technique proposed a mechanism to train a shared model by aggregating only the local models trained on local data without collecting all the locally stored data centrally. Subsequent research showed that the described approach could not only be employed in cross-device settings (e.g., mobile phones, sensors) but also in cross-silo settings, involving data generated and/or stored by different organizations, like hospitals or factories, e.g. [HYF+20], [SER+20], [ZLM+21].

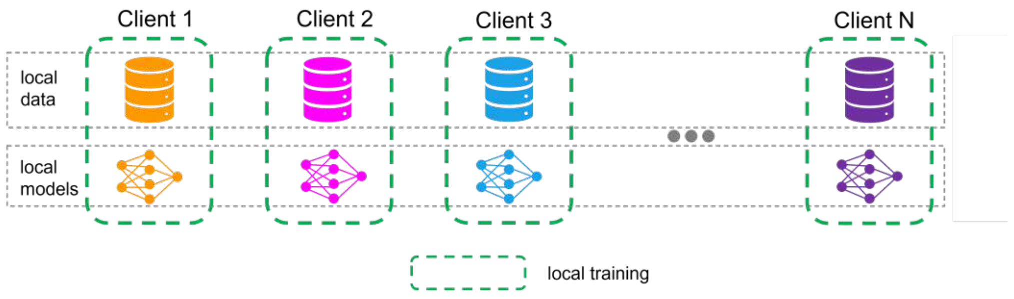
Federated learning is defined as a machine learning subcategory, where some edge devices or organizations, called clients, cooperate to achieve the global learning objective under the coordination of a central server [KMA+19]. Contrary to the centralized approach, where all data have to be collected on the central server, see Figure 1.1b, "in the case of federated learning, the clients do not transfer their data to the central server but build their local models and share model updates to the central server. The central server aggregates these updates into the global model and transfers the global model back to the corresponding clients for further training" [GHP22], see Figure 1.1c.

Formally, the learning objective of federated learning is usually to minimize the global objective function

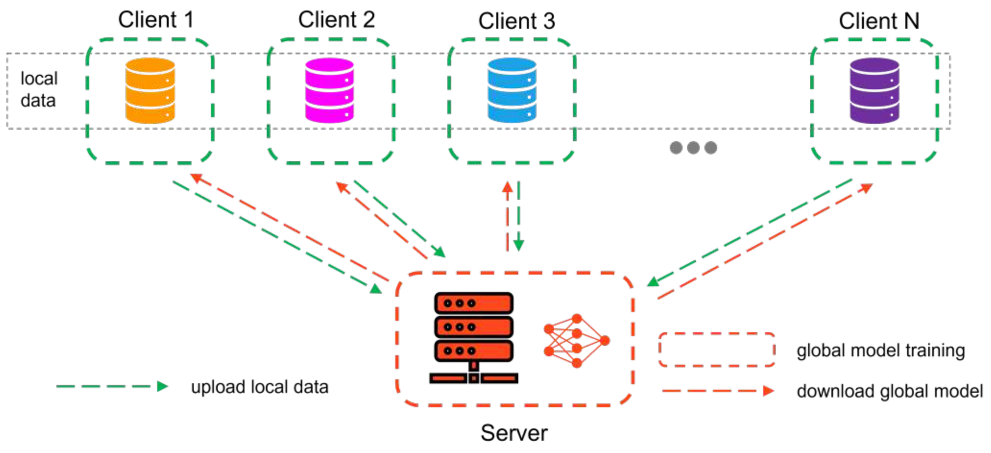
$$\min_{w} F(w), \text{ where } F(w) := \sum_{k=1}^{N} p_k F_k(w), \tag{1.1}$$

where N is the total number of clients (devices, organizations, etc.), $p_k \geq 0$ and $\sum_k p_k = 1$, $F_k(w)$ is the local objective function of k-th client [LSTS20]. The selection of weights p_k is subject to the user's choice. A common and quite natural strategy, known as Federated Average (FedAvg) [MMR+17], is defined by setting the weights p_k proportionally to the amount of data hold by each client, i.e. $p_k = \frac{n_k}{n}$, $n = \sum n_k$, where n_k is a number of data samples of k-th client.

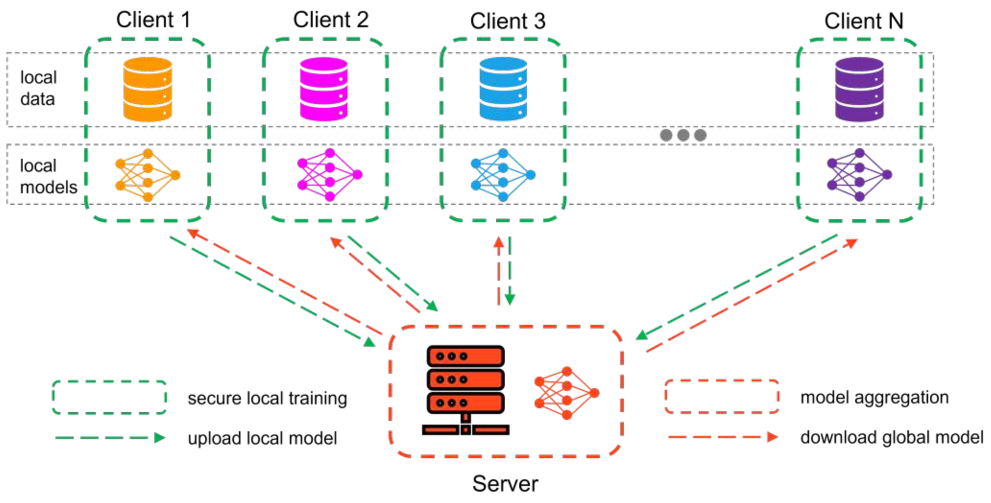
Finding an optimal solution to the problem defined in Equation 1.1 is not restricted to proper selection of weights or optimization of local objectives. There are numerous challenges present in the federated networks, which do not exist in the case of classical distributed training on the central server. First, we have to deal with heterogeneous data among the clients in most cases. Data on mobile devices generated by different users may be specific to these users' preferences and significantly vary in distribution. Similarly, data collected by various production companies or units may reflect specifics of their products or production processes. Thus, the "data originating from various clients usually cannot be considered as independent identically distributed (i.i.d.)" [GHP22], i.e., we cannot expect the local data to be a part of a single global distribution. Moreover,



(a) Local training



(b) Centralized training



(c) Federated Learning

Figure 1.1: An illustration of different learning architectures: **(a)** Local training, where clients use only their own data and no global model is trained, **(b)** Centralized training, where all local data are collected on the central server and is used there to train a single global model, **(c)** Federated learning, where the training is done locally and only model updates are aggregated centrally.

sizes of data on each client's side may vary as well. This aspect of federated networks is commonly known as the statistical heterogeneity. Heterogeneous data among clients may lead to a strong divergence of an aggregated global model resulting in lower or even much lower performance, at least for some clients [LSZ+18]. Second, in addition to varying the data distributions across the clients, clients' computation and communication capabilities are usually quite diverse. This issue is referred to as systems heterogeneity, because differences in hardware, network connectivity, etc., may significantly influence the ability of the clients to participate in the global training and their timely response. While this challenge is more actual in orchestrating a federated network with millions of devices, rather than in the case of tens to hundreds of organizations, still also in cross-silo settings it cannot be neglected entirely. Third, even though it is expected that federated learning strategies may be helpful in tackling communication costs since instead of uploading the entire data, only model updates are communicated to the server, "communication may become a critical bottleneck in federated approaches [KMA+19], especially when a great number of clients have to transfer local updates a multitudinous number of times" [GHP22]. Finally, although federated learning approaches suggest a strong advantage in privacy-sensitive applications, as the clients' data remain local, sharing the "model updates can pose a risk of revealing some sensitive information" [GHP22], while the approaches with enhanced privacy may have reduced effectiveness or efficiency [MRTZ18].

Our work focuses on exploring the performance of federated learning approaches in cross-silo settings with a reasonable number of participating clients within an order of magnitude of tens, and thus, systems heterogeneity and communication challenges are less relevant for our analysis. Therefore, we concentrate primarily on the optimization task from Equation 1.1, testing different statistical heterogeneity scenarios.

1.3 Research questions

McMahan et al. [MMR+17] introduced FedAvg, a "federated optimization method, where a global model is derived by taking a weighted average of the local models' parameters" [GHP22]. Empirical evaluations indicate a good performance of FedAvg; however, this approach "does not provide any convergence guarantees and can diverge in practical settings when data are heterogeneous" [LSZ+18]. In other words, simple averaging of the local models may not be optimal for constructing a global model, especially when clients' data are non-i.i.d. Therefore, consideration of other aggregation methods becomes inevitable.

Ji et al. [JSP+21] researched numerous aggregation strategies and suggest a taxonomy of federated model fusion algorithms by designating them to one of the following groups: adaptive/attentive aggregation, regularization, clustering, Bayesian methods, and fairness, which are based on specifics of every algorithm. In our research we focus on three aggregation algorithms in addition to FedAvg: FedProx, FedYogi, and qFedAvg, as they have different designs and suppose to address different disadvantages of the classical FedAvg approach. Since achieving the acceptable performance is vital for every machine learning task, we formulate our first research question as follows:

- ***RQ1:*** How effective are selected aggregation algorithms for our predictive maintenance/defect detection task?

Typical predictive maintenance/defect detection problems could be represented by binary classification tasks with two values - failure / no failure. Also, the data is usually highly imbalanced - failures have much lower occurrence in the data than healthy states. Thus, we consider two metrics for effectiveness, which are meaningful performance indicators in case of imbalance data. We calculate F-beta score [Ste] based on the resulting confusion matrix and subsequently the standard metrics precision and recall:

$$F_\beta = \frac{(1+\beta^2)\cdot \text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}.$$

By setting the parameter β we will be able to control the relative importance of the different types of prediction mistakes - false positives vs false negatives. Also, since one of the main tasks of predictive maintenance/defect detection is to optimize the economic cost of maintenance, the F-beta score alone may not be enough to incorporate various business constraints and requirements [SMWB18]. Therefore, we additionally model the maintenance cost as the total cost, required for unnecessary checks of false positives and replacement of missed false negatives:

$$\text{Maintenance cost} = c_1 \cdot \#\text{ false positives} + c_2 \cdot \#\text{ false negatives},$$

where c_1 is a unit cost of checking for a possible failure during the production process and c_2 is a unit cost for replacement of a faulty product; often $c_1 \ll c_2$.

However, model effectiveness is not the only aspect for optimization in federated networks. Numerous uploads and downloads of the model updates may significantly influence communication costs and become a critical issue [KMY$^+$16]. Thus, it is vital to consider decreasing communication by reducing the amount of communicated information or by reducing the number of communication rounds; a communication round represents the full cycle between local downloads of the global model and uploads as well as subsequent aggregation of local updates on the central server. On the other hand, it may be also important to look both at the total effectiveness of the whole network and examine the effectiveness metrics for each client locally. One drawback is that after the aggregation process, some clients may be advantaged or disadvantaged if, for example, the model gets biased towards the clients with more data or towards the clients having more similar distributions. Hence, we formulate the second research question:

- ***RQ2:*** What is the relationship between the effectiveness of the aggregation algorithm and the communication cost and fairness of the model?

In our selected aggregation methods, the amount of information communicated in each round is the same, i.e., if the same model architecture is selected, the number of communicated model weights is the same. Thus, as a proxy for communication cost, we can straightforwardly use the number of communication rounds between the server and the clients. To estimate the fairness, there are different definitions and metrics in machine learning. Li et al. [LSBS19] studied the fairness issue in federated learning and suggested a fairness definition for a federated learning framework:

Definition 1.3.1 (Fairness of performance distribution) *"For trained models w and w', we informally say that model w provides a more fair solution to the federated learning objective 1.1 than model w' if the performance of model w on the n devices, $a_1, ..., a_n$, is more uniform than the performance of model w' on the n devices." [LSBS19]*

To measure the fairness as defined above, i.e., to assess the uniformity of the effectiveness measures, we consider one of the fairness metrics suggested by Divi et al. [DLFB21] - Entropy of clients' performance metrics; in our case entropy of clients' individual F-beta scores:

$$\text{Entropy} = -\sum_{i=1}^{n} \frac{F_i}{\sum_{j=1}^{n} F_j} \mathbf{log} \frac{F_i}{\sum_{j=1}^{n} F_j}.$$

Finally, we compare the performance of the federated strategies against two baseline methods - pure local training and centralized training. In the case of local training, the clients train individual models utilizing exclusively their local data, without any further communication, see Figure 1.1a. In the case of centralized training, all the data are collected centrally, consequently, the data privacy is sacrificed and a single central model is trained, see Figure 1.1b. Thus, we formulate the third research question:

- ***RQ3:*** To what extent can the federated learning approach replace the centralized and /or local training without major losses in model effectiveness?

To answer the research questions, we select four industrial datasets with a high class imbalance - three publicly available datasets and one private dataset from the industry. We then choose a base model for each dataset and use the same architecture for all individual clients, and the global models. Due to convenience for executing calculations with weights, and since most of federated approaches are designed for and used with neural networks, see e.g. [ZZL+21], [LWW+21], we select Feed-Forward Neural Networks as a base model. We use the same initial neural network, i.e. same architecture with the same initial weights, to train 4 federated learning approaches. Also, we train the same network in local and centralized settings, thus making the results of different methods comparable to each other.

CHAPTER 2

Related Work

2.1 Machine Learning in Manufacturing

The fourth industrial revolution evolved as a consequence of recent technological advances in Internet of Things, computational and connectivity resources, cloud and edge computing, digitization of processes, and others. Data analytics and artificial intelligence have become an essential part of smart factories - self-aware, self-predictive, and self-maintaining organisms - and have found broad applicability to different areas ranging from production operations and supply chain management to customer service [RTID21]. By applying machine learning techniques to different segments of manufacturing pipelines, companies can increase the level of automation and efficiency, saving both money and time.

One of the fundamental functions of every manufacturing process is maintenance, which has a significant contribution to manufacturing costs and may constitute from 15% up to 60% of total operational costs [Mob02]. Thus, there is a lot of potential for various optimizations by utilizing machine learning approaches.

The increased importance of intelligent data-driven engineering solutions has also been reflected in academia. Zonta et al. [ZdCdRR+20] in their systematic literature review indicate that the amount of research on predictive maintenance has been growing continuously and more than half of the works deal with data-driven AI-related techniques. Cinar et al. summarize numerous examples, how these techniques were applied successfully in several predictive maintenance fields, like systems and equipment health management, product state of health identification, remaining useful life prediction, and other. The most relevant for us is the diagnosis of products or their parts.

In component health diagnostics, two approaches serve as a common practice - visual inspection of the images of completed parts or product surfaces or model-based approaches taking the advantage of the metrics measured and collected during different stages of the production process. The appropriateness of one method or another is determined by the data available and problem specifics.

The latest visual inspection approaches cover recent advances in deep learning methods for image processing, like convolutional neural networks (CNN). For example, using CNN, Weimer et al. [WSRS16] achieved over 99% classification accuracy for industrial optical inspection tasks with different industrial texture patterns. Their suggested method showed also superior results in terms of true negative rate, i.e. it succeeded in correctly identifying defected cases for the most texture types. Other works employing different variants of CNN, e.g. steel defect detection [MMC+12], wood veneer quality control [SLZ+20], casting products inspection [NCP+21], demonstrate similar good performance. Thus, CNNs are typically preferred over classical methods, e.g. feature extraction, like SIFT/HOG/etc., combined with SVM/PCA/etc., fuzzy clustering, Hidden Markov Models.

Different methods are used with the data representing measurements gathered during the production, such as temperature, pressure, vibration, speed, oscillation, corrosion levels, and others. Typical techniques for both classification/regression problems include regression models, support vector machines (SVM), decision trees and random forest, and artificial neural networks. The most common and best-performing supervised learning approaches for early fault detection or component health assessment, as determined by Cinar et al. [ÇANZ+20], are artificial neural networks, random forest, and SVM. Still, the effectiveness of the models differs substantially depending on the problem specifics and underlying data. For some tasks, neural networks can be used as a primary solution showing satisfactory results and/or outperforming other approaches. The examples include wind turbine testing [BS15] with almost 93% prediction accuracy, tool wear monitoring of a Computer Numerical Control (CNC) milling machine [HM19], where the neural network outperformed SVM and KNN, prediction of motor failure [SSVFSdSAdS19], where ANN showed equal or higher performance than SVM, decision tree, random forest. At the same time, many works prove usefulness of other algorithms, like random forest, SVM, etc., e.g. [FMNC19], [WY07], [XHL18], [BDP19], [QMM+18]. Overall, according to the research performed by Zhang et al. [ZYW19], neural networks indicate a decent performance in predictive maintenance tasks, however, they are primarily suitable in case of complex large-scale systems.

Nonetheless, the applicability of machine learning to a concrete company's case is strongly correlated with the availability of the data and heavily ranges throughout the industry sectors. For example, many studies exist for the semiconductor industry, the earliest of them going back to 1993 (e.g. [AWG93]), due to the possibility to gather lots of data in a short time. Still, typical challenges, like missing or imbalanced data, remain applicable up till now [CB17]. However, in most cases, it is extremely difficult to gather enough high-quality data, which may be adopted for comprehensive analyses and building models, as the behavior of machines and causes for failures are too diverse and complex [ZHA19]. For example, the data used in the previously mentioned work of Weimar et al. [WSRS16] was created artificially, still incorporating real-world observations. In practice, only a small part of companies have already reached the state of continuous real-time monitoring of manufacturing by incorporating automated processes employing predictive models [HMV17]. Most manufacturers still perform manual periodic inspections or rely on predefined rules or critical levels, which are mostly based on expert knowledge and best practices.

2.2 Federated Learning approaches

Federated Learning is a machine learning method designed to deal with decentralized data without collecting the data centrally. In practice, there are three main scenarios of how federated learning could be utilized for training models on partitioned data - horizontal federated learning, vertical federated learning, and federated transfer learning.

Horizontal federated learning (HFL) is applicable when clients share a similar feature space, but the users/objects are different, see Figure 2.1.(a). For example, there are some manufacturing units, which are spread geographically and produce similar products, thus, measuring the same features during the production. In such a case, HFL is useful to increase the size of the data, leading to potential improvements in the global model's effectiveness. In general, HFL settings require that different clients work on a similar machine learning problem, but due to some reasons cannot share their data. The clients would then calculate and upload local gradients to a central server, and the server would incorporate local gradient updates into the global model.

Contrary to HFL, "vertical federated learning (VFL) is employed, when the feature spaces among the clients are rather disjoint, but these clients store the features or characteristics of the same users/objects" [GHP22], see Figure 2.1.(b). In other words, the same users/objects are processed by different clients for different tasks/purposes. Primarily, VFL methods are applicable to cross-silo settings, i.e. when clients are organizations or their units rather than individual devices. For example, the same product is tracked over a sequence of stages of the production process and different metrics of this product are recorded at each stage. Another typical example of vertically partitioned data covers the patients, whose drug files are stored by pharmacies and whose clinical history is collected by hospitals. Obviously, VFL is more complicated than HFL due to entity resolution challenges, since the matching the individual entities across the different parties may not always be known a priori or be easily achieved [GSB+17]. Thus, the aim of VFL is to collect distinct features in an encrypted way to obtain a comprehensive model. Still, available secure encryption techniques may only be used with simple models like logistic regression and thus need to be further improved [LFTL20].

In the case of federated transfer learning (FTL), the clients observe neither common features nor common users/objects, see Figure 2.1. Typically, FTL scenarios exhibit a lack of data labels and poor data quality. Liu et al. [LKX+20] first suggested an FTL framework to ensure privacy-preserving transfer learning with a similar level of accuracy. Some works employ FTL approaches, e.g. [CLL+20], still, the research state of FTL is not yet mature [LFTL20].

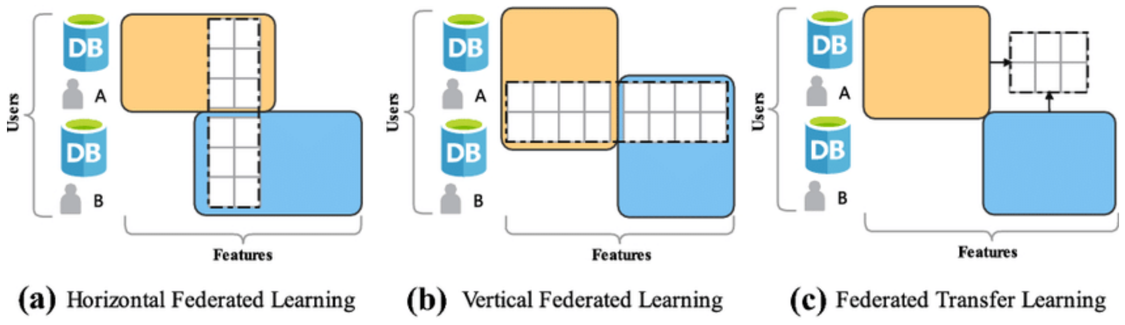

Users
DB
A
DB
B
Features
(a) Horizontal Federated Learning
(b) Vertical Federated Learning
(c) Federated Transfer Learning

Figure 2.1: Different data partitions in federated learning
Figure source: [LHW+21]

In practice, HFL is most commonly adopted [ZZL+21]; in our analysis, we also model HFL settings. Thus, we restrict further discussion to HFL approaches only. There are many directions for optimization in federated learning, such as decreasing communication overhead, leveraging fault tolerance and resource allocation, and dealing with privacy risks. In our experiments, we focus on model optimization and data heterogeneity. Model optimization task essentially encompasses two aspects - "selection of underlying machine learning algorithm for local training and a choice for model aggregation to obtain the global model" [GHP22].

Zhang et al. [ZXB+21] in their survey on federated learning outlined three main machine learning model groups, which are used in federated schemes - linear models, tree models, and neural networks. There are some works showing successful implementation of privacy-preserving linear and tree models. Nikolaenko et al. [NWI+13] proposed a ridge regression system, which achieved the best performance. Cheng et al. [CFJ+21] constructed SecureBoost, a lossless gradient tree-boosting system, which before constructing boosting trees "conducts entity alignment under a privacy-preserving protocol" and was indicated to be as accurate as its non-federated counterparts. Li et al. [LWH20] designed a similarity-based federated learning framework for gradient boosting decision trees and obtained similar, yet slightly worse, accuracy as the centralized approach. Still, different types of neural networks prevail in federated learning applications due to their broad applicability to numerous problems - from modeling tabular data to solving computer vision and text processing tasks.

While some researchers suggested federated versions of random forest and SVM, [LLL+20], [GLZL21], [VYJ08], these approaches still lack proof of concept. Also, only neural networks and linear and simple tree models are realized in federated learning frameworks, as described in Section 2.4.

Another area for optimization in HFL is related to the production of a global model based on the local ones. FedAvg [MMR+17] is a common and straightforward method for combining the local models. Clients train distinct models employing only their local data, communicate their models to the server and the server produces the global model by taking a simple or weighted average of the local models. However, FedAvg has some weaknesses, especially when clients' local data are heterogeneous. When training their local models, clients carry out the optimization of their local objectives only, which in the case of heterogeneous data may differ significantly among each other as well as from the global optimum, i.e. the optimum of the global model.

There are numerous methods in the literature, which deal with different perspectives in the management of the global training process. In our research, additionally to FedAvg, we explore FedProx, FedYogi, and qFedAvg. We provide detailed information on these algorithms in Section 3.3, thus we do not discuss them here. Still, there are many other approaches, which are more advanced than FedAvg and often use FedAvg as a base. SCAFFOLD [KKM$^+$20] uses variance reduction techniques to restrict the client drift from the global model. Similar to FedProx, which limits deviations from the global model by adding a proximal term, SCAFFOLD may help to decrease communication costs and the global effectiveness in heterogeneous data. While FedProx and SCAFFOLD control the local training process to favor global optimization, FedNova [WLL$^+$20] seeks to improve the aggregation stage by normalizing local model updates when averaging. Attentive methods, such as FedAtt [JPL$^+$19] and FedMed [WLW20], are used to manipulate the global model at the aggregation stage as well. While FedAvg usually weights the client models proportionally to the clients' underlying data, attentive methods operate with the attention scores for client model parameters. FedMA[WYS$^+$20] is another method, used for leveraging global model effectiveness and total communication burden. It is based on an observation that different permutations of neural network architectures yield the same outputs. Thus, this technique suggests composing the global model layer-wise, by identifying similar elements in the network (e.g., neurons in MLPs, channels in CNNs, etc.) and averaging them.

All previously described techniques aim for optimization of the total performance of the system, i.e. improving the effectiveness and/or communication costs with respect to the whole data available in the system. An alternative solution is to design personalized federated approaches. On the one hand, these solutions should prioritize local optimization. On the other hand, they should still profit from the knowledge exchange by participating in federated training. Per-FedAvg [FMO20] and pFedMe [TDTN20] are two examples of personalized federated learning algorithms.

To be noted, FedAvg often serves as a baseline in evaluations of alternative approaches.

For the sake of completeness, we point out that there are also federated learning mechanisms designed for many other tasks than supervised machine learning. Such approaches include meta-learning, reinforcement learning, collaborative filtering, matrix factorization, generative adversarial learning, and others. Some examples in these areas are mentioned in the studies [LWW$^+$21] and [JSP$^+$21].

2.3 Industrial Federated Learning Applications

Originally, federated learning was suggested for cross-device settings, i.e. for mobile or edge devices, due to the increasing computational power and storage capacity of these devices. Examples of successful applications include, for example, next word or emoji prediction on keyboard [CMOB19] and human activity recognition [SVG18].

Inspired by the promising results in cross-device applications, scientists found also plentiful use cases in the industry. Due to its privacy-preserving mechanisms, FL gained decent attention in health care. Normally, medical institutions process loads of patients' data, however, the data of a single institution still may not be enough to build a reliable model. Pfohl et al. [PDH19] used electronic health records from intensive care units to model the length of stay and hospital mortality and showed that FL surpasses local learning achieving efficacy similar to that of centralized training, i.e. cooperation of medical institutions would be beneficial over non-cooperation. Huang et al. [HYF+20] adopted the FL approach to train a model for patient mortality prediction using drug utilization data, which outperformed the baseline. Silva et al. [SGR+19] used brain magnetic resonance images for feature extraction in a federated manner for further determination of brain diseases.

Driven by the achievements in privacy preservation, but also by promising FL performance, FL found wide-ranging applications in other industrial branches as well. Zhang et al. [ZLY+20] proposed to combine FL and blockchain technologies to detect device failure in the Industrial Internet of Things. They designed a centroid distance weighted federated averaging algorithm, which builds up weights depending on the separation of the classes of each client's data set, which showed similar or better results than classic federated averaging. While the blockchain was used to ensure the data integrity by hashing and periodically storing the clients' data on the blockchain. However, in many tested cases, centralized and local training showed more favorable results. Liu et al. [LGN+20] performed anomaly detection using sensor data from different fields, e.g. power demand space shuttle, etc. Due to the exchange of lots of information, i.e. high number of gradients, which implies high communication costs till a model is sufficiently trained, they suggested a "Top-k selection-based gradient compression scheme" to tackle the communication challenges. The proposed technique proved to be effective and efficient, also better than some centralized schemes. Still, these results were shown for time series data only. Li et al. [LWS+20] introduced the FL-based intrusion detection model in industrial cyber-physical systems. They enhanced their model with a secure communication scheme, which is aimed to retain the security and privacy of the local data but also of the model weights. Their suggested method demonstrated close performance to the baseline. However, it was only tested with a single dataset. Hu et al. [HGLM18] applied FL in urban environment monitoring. Insufficient environment sensing sites, sparse sensory data, and incomplete records showed to make a detrimental impact on the central model performance. The suggested solution to cluster the sensors into regions and subsequently apply federated averaging improved computational efficiency and increased model accuracy compared to the centralized approach. Yet, this approach was again explored with only one dataset.

Other industrial FL applications include FL-based recommender systems, e.g. video recommendations [DZW+19], federated collaborative filtering [AUDIK+19], traffic flow prediction [LJK+20], energy demand prediction [SHN+19], credit card fraud detection [YZY+19], and many other. To be noted that almost all of the described solutions used different kinds of neural networks.

Even though FL utilization in the industry is manifold, the number of works on predictive maintenance remains limited. Even less research exists for tabular imbalanced data. Zhang et al. [ZGC+21] suggested privacy-preserving federated learning approach AdaPFL for fault diagnosis in Internet-of-Ships with adaptive changes to the model aggregation interval for reducing cryptography computation and communication costs. Empirical experiments prove the high effectiveness of the AdaPFL close to the centralized training approach. Yet, the data used in the experiments was perfectly balanced with 500 samples per class. Ge et al. [GLZL21] constructed an FL SVM and random forest approaches for predicting the failures in a production line, compared them to centralized SVM and random forest respectively, and achieved similar effectiveness between corresponding federated and central learning models, e.g. SVM centralized vs SVM federated. However, this study can be generalized to a very restricted extent and the suggested algorithms may show different performances in other experimental settings. Zhang et al. [ZLM+21] analyzed the effects of different types of data heterogeneity in machinery fault diagnostics. They introduced an FL method with dynamic validation and self-supervision with promising performance on non-i.i.d. data. At the same time, they focused on modeling non-i.i.d.-Class and non-i.i.d.-Domain imbalance, e.g. by assigning different types of faulty states to different clients, and did not explore the data imbalance issue within the same client. Mowla et al. [MTDC19] discuss the class-imbalance problem for detection of jamming attacks in Flying Ad-Hoc Network and propose a client group prioritization mechanism to identify better client groups for global model construction. Still, this work is isolated to the Unmanned Aircraft Vehicle data, which is pre-processed in a way to artificially create an unbalanced class distribution. Also, only federated averaging is considered for the aggregation of local models. Zhang et al [ZWZ+21] adopted a combination of autoencoders and Siamese networks to detect faults in permanent magnet synchronous motors. The proposed method was tested with both centralized and federated learning approaches using sparse data and achieved increases in accuracy against other common deep learning architectures. But here again, only one dataset was included in the analysis, making the study hardly generalizable.

Thus, to our best knowledge, there are no studies so far on imbalanced tabular predictive maintenance data that evaluate the performance of several federated learning algorithms simultaneously considering the effectiveness, communication cost, and fairness of the models.

2.4 Existing Federated Learning Frameworks

With the increasing interest in federated learning methods, several federated frameworks have been developed recently, which enable a more convenient practical usability of this novel approach. Common frameworks cover TensorFlow Federated (TFF) from Google [ten] and PySyft from OpenMined community [pys].

TensorFlow Federated and PySyft are both designed to be used together with deep learning approaches - TFF is built on top of TensorFlow, and PySyft should support PyTorch and TensorFlow libraries. Even though they provide powerful solutions to many tasks, which require deep learning solutions, still they do not support any classic machine learning approaches, which may be disadvantageous in some cases. Moreover, the choice of federated algorithms is quite limited despite many existing strategies for aggregation of the local models. TFF aggregation techniques are limited to FedSGD, FedAvg, FedProx and MIME_lite [KJK+20] in weighted and unweighted versions. Similarly, PySyft covers also only FedAvg, FedSGD, FedProx, and FedDANE as ready techniques. As opposed to TFF, PySyft supports both vertical and horizontal FL, while TFF can be used only for horizontal data partitioning in the FL network. Still, both libraries are composed of different classes for computation of estimates, aggregation of local models, performance metrics calculation, etc., so users may further extend and develop the existing resources based on their own needs. Another general issue with these frameworks, which would not have any effect for our analysis though, is that both are suited well for experiments and simulations, however, practical applicability in industrial products is still limited [KYF+20].

Other open-source frameworks include Federated AI Technology Enabler Framework (FATE) [fat], Paddle Federated Framework [pad], Federated Learning and Differential Privacy (FL&DP) Framework. FATE was designed for the industrial-level application, i.e. is well-suited in cross-silo settings, ensuring a high degree of privacy by using homomorphic encryption and secure multiparty computing. It can be used with neural networks but also implements some regression models and decision trees. Paddle Federated Framework uses a deep learning platform PaddlePaddle and has support not only for centralized but also for decentralized FL, where no central server is involved and clients may communicate directly with each other. Contrary to TFF and PySyft, FATE and Paddle have "all the necessary features to be used in production" [KYF+20]. FL&DP is a rather simple framework, yet it works together with neural networks as well as with linear and clustering models from the scikit-learn package. Unfortunately, this framework has a very low number of contributors (6).

Most open-source frameworks are designed for simulations on a single machine or may be available for production in cross-silo settings only, while the solutions supporting large-scale industrial applications are commercial and thus closed. What is more, they primarily rely on client simulations and usually perform the necessary calculations in nested loops rather than replicate real FL environments, where the calculations are performed on edge devices simultaneously. Therefore, Beutel et al. suggested another

FL framework FLOWER [BTM+20], which should also model systems heterogeneity in computing capabilities, network bandwidth, model, etc. Furthermore, the authors claim that their solution offers a language and ML framework-agnostic implementation, e.g. compatible with PyTorch, TensorFlow, scikit-learn, or even raw NumPy. Still, by looking at the source code of this framework we conclude that mainly models, which are defined by their weights, can be used in the framework. On the other hand, it includes the most model aggregation strategies in comparison to the frameworks described before.

Commercial institutions like NVIDIA and IBM offer their own frameworks as well, which again confirms the relevance of federated learning to the industry.

We considered different available federated framework networks and decided to implement our own to have the flexibility necessary for our explorations with industrial imbalanced data regarding different simulations of client data, resampling and loss re-weighing techniques, aggregation algorithms, performance metrics, and others.

CHAPTER 3

Background

3.1 Neural networks

Neural networks / deep learning has become a powerful tool for solving different machine learning problems in a supervised manner due to recent developments of hardware and computational resources. Numerous deep learning approaches exist in modern practice, like convolutional neural networks designed for image processing or recurrent neural networks for sequence modeling [GBC16], but also more sophisticated methods with e.g. attention mechanisms for multiple natural language processing tasks [VSP+17], and other. Despite specifics in model architectures, all these approaches share a rather simple base originating from a perceptron method inspired by the neuroscience [Ros58] and its extension to a more flexible multi-layer perceptron (MLP).

Looking at the example in Figure 3.1, MLP can be considered as a mathematical function, which itself consists of a set of simple functions connected in a chain, mapping input values $\mathbf{x}$ to output values $\mathbf{y}$, i.e. $\mathbf{y} = f_n(...(f_2(f_1(\mathbf{x}))))$. The inputs, intermediary values, and outputs, all called neurons, are organized in layers, where the calculations are processed simultaneously on all neurons of a layer and where the outputs of the same layer are propagated forward as inputs for the next layer. The first layer of a network is called an input layer, the final layer is an output layer and all layers in between are referred to as hidden layers. To generate the outputs, a set of calculations on input values are performed. The input values, which are themselves the outputs of the preceding layer (with an exception of the input layer), are multiplied by the parameters called weights and are shifted by the bias parameters. The resulting values are then summed up and the resulting sum is transformed via an activation function. The activation function is used to distinguish to what extent the output of the neuron is relevant for the whole model. Common activation functions include non-linear functions like rectified linear unit (ReLu), sigmoid, hyperbolic tangent, and others, thus allowing to model non-linear relationships between input $\mathbf{x}$ and output $\mathbf{y}$.

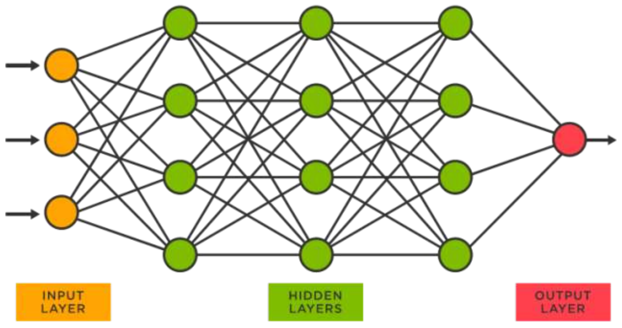


Figure 3.1: An example of MLP with three input units, three hidden layers and a single output unit

Neural networks have high flexibility due to the possibility to tune their architecture by selecting the number of layers and calibrating the sizes of each layer by adjusting the number of neurons, as well as by setting the activation functions. However, training the neural network usually means optimization task for millions of weight and bias parameters. This can be done by applying small changes to the parameters to step-wise decrease the model error defined by the loss function. Different loss functions exist; one commonly used for classification task is cross-entropy loss, which in binary case is defined as follows:

$$L = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \cdot \log(f_\theta(x_i)) + (1 - y_i) \cdot \log(1 - f_\theta(x_i)\right],$$

where n is the number of training samples, y_i is the target label for training example i, x_i input for training sample i f_θ is the neural network with weights θ. To minimize the loss function Stochastic Gradient Descent (SGD) algorithm [GBC16] or its adaptations like ADAM [KB14], etc. can be used. All are based on the idea of calculating gradient, i.e. vector of partial derivatives, in order to determine the direction of the sharpest increment of the given function. For minimization of the function, a step in the opposite direction is taken, and this procedure is repeated, iteratively adjusting the parameters of the function till certain criteria are reached, for example, after a certain number of steps or till the error is decreased to a given level. The size of these, usually small, steps is controlled by the learning rate, which is selected by the user in advance. The higher this parameter is set, the bigger steps can be done at once, potentially decreasing the number of optimization rounds. Still, the learning rate is usually selected rather moderate in order not to miss the targeted minimum, by "jumping" over or around it.

3.2 Handling data imbalance

A key feature of industrial predictive maintenance or defect detection data is a very high class imbalance, reflecting the real world that failures happen rather rare in comparison to the total production, however, the cost of missing such failure is quite high. Classifying unbalanced data generally leads to poor prediction accuracy and low generalization ability of the model [Lee00]. Moreover, since miss-classification of a faulty object results in a much higher relative cost, we aim to find an appropriate model to optimize not only for total accuracy but, more important, to correctly detect the minority class still keeping the number of false positives low. Hence, approaches for dealing with the class imbalance problem were broadly studied over the last decades. Existing solutions usually address this issue from two perspectives - on the data level by applying class re-balancing approaches like data resampling or on the algorithm level by employing cost-sensitive re-weighting techniques [JK19], for example by adjusting the weights of classes in the cost function.

3.2.1 Data Resampling

Data resampling techniques are frequently used to alter the class distribution. The simplest approaches include random undersampling and oversampling. In random undersampling, a selected percentage of majority class samples are discarded randomly, while in random oversampling random instances of the minority class are duplicated to achieve a selected class distribution. However, random oversampling was shown to have a tendency for overfitting [CJK04] and thus alternative oversampling methods shall come into consideration.

One of the more sophisticated oversampling methods, proposed by Chawla et al. [CBHK02] is Synthetic Minority Oversampling Technique (SMOTE). Instead of simply duplicating the existing minority samples SMOTE is used to generate new synthetic samples from minority distribution. For each minority instance, a neighborhood of the k closest minority instances is considered and new instances are created by moving in one or several directions towards the nearest neighbors.

More resampling techniques exist, for example, cluster-based oversampling, one-sided selection, Wilson's editing. Still, even though they may seem more "intelligent" than random duplication or discarding of instances, Van Hulse et al. [VHKN07] showed these simple techniques may be preferable in case of severe class imbalance when positive instances constitute less than 5% of total data.

3.2.2 Cost-sensitive techniques

In comparison to data resampling, cost-sensitive techniques do not introduce any perturbations in the data, but rather reflect the variability of costs by introducing the penalties of different magnitude for different classes. In case of deep learning, a common approach is to include weight factors into a loss function, i.e. previously defined cross-entropy loss is adapted to weighted cross-entropy loss given by:

$$L_{weighted} = -\frac{1}{n}\sum_{i=1}^{n}\left[w \cdot y_i \cdot \log(f_\theta(x_i)) + (1-y_i)\cdot \log(1-f_\theta(x_i)\right],$$

where w represents the weight parameter(s). Selecting the weights may be quite straightforward by setting them to inverse class frequency, or more advanced like assigning them with Effective Number of Object Class [CJL+19], defined by formula $w = \frac{1-\beta^{n_j}}{1-\beta}$, where $\beta \in [0, 1)$ is a hyper-parameter and n_j is a number of samples in j-th class.

Alternatively, other options like Focal Loss [LGG+17] with automatic adjustments to weights based on the confidence of the classifier may be applied. However, the experiments with different weighting strategies or other loss functions will be considered out of the scope of this work, and the simple inverse class frequency weights will be used.

3.3 Federated Learning aggregation algorithms

FThe ederated Learning paradigm covers machine learning approaches, where the central server orchestrates the training performed on the clients' side, collects the local updates centrally, and aggregates the updates into a global model. In our work, we explore 4 aggregation strategies - FedAvg, FedProx, qFedAvg, and FedYogi, as they put a different focus on optimization methods, robustness, and fairness.

3.3.1 Federated Averaging (FedAvg)

One of the leading techniques to find a global optimum in a federated network is Federated Averaging [MMR+17]. Let us consider a federated infrastructure with one central server and N clients. At any time t the central server hosts a global model represented by its weights w^t (e.g. weights defining a neural network). Within one communication round, the central server selects K clients, which will participate in this training round and communicates to them the global model w^t. Each client $k \in K$ on their side carries out E epochs of local optimization, i.e. each client performs E steps of SGD with a learning rate η on the local objective function F_k. The obtained local models w_k^{t+1} are transferred back to the central server, where an updated global model w^{t+1} is constructed by taking the average of local models. Thus, starting with an initial global model w^0, which, for example, could be initialized randomly, after executing T communication or in other words global training rounds, we obtain a trained global model, which possibly better solves the global optimization problem defined in Equation 1.1.

In Algorithm 3.1 we provide the pseudo-code for the original version of FedAvg suggested by McMahan et al., where a simple average of the local models is calculated. Later works suggest utilizing a weighted average, for example, with weights proportional to the size of data held by each client to improve the performance of the global model and avoid a big influence of outlying local distributions.

3.3.2 FedProx

Although FedAvg was widely explored empirically and indicates a reasonable performance, especially in convex settings, simply averaging the local models may not always lead to the optimal predictions, when local client distributions vary significantly. Indeed, in

Algorithm 3.1: Federated Averaging (FedAvg), [MMR+17]

```
  Input: $K, T, \eta, E, w^0, N, p_k, k = 1; ...; N$
1 for $t$ = 0;...; T–1 do
2     Server selects a subset $S_t$ of $K$ clients at random (each device $k$ is chosen with
        probability $p_k$)
3     Server sends $w^t$ to all chosen clients
4     Each device $k \in S_t$ updates $w^t$ for $E$ epochs of SGD on $F_k$ with step-size $\eta$ to
        obtain $w_k^{t+1}$
5     Each device $k \in S_t$ sends $w_k^{t+1}$ back to the server
6     Server aggregates the $w$'s as $w^{t+1} = \frac{1}{K} \sum_{k \in S_t} w_k^{t+1}$
7 end
```

case of heterogeneous data, i.e. when data are not identically distributed, but also in heterogeneous networks, FedAvg may have severe losses in performance, e.g. [KKM+20], [HQB19]). Moreover, FedAvg comes without convergence guarantees and can diverge in practice. [LSZ+18] One more concern associated with FedAvg is that it does not consider a realistic situation, when different clients may be able to complete different amounts of work, i.e. to achieve different progress or even drop out of the network.

To improve the stability of the training process in diverse settings in terms of both data and systems heterogeneity, Li et al. [LSZ+18] introduced the FedProx algorithm, an adaption of FedAvg. This approach suggests adding a proximal term μ, which shall "restrict strong drifts of the global model towards any of the local ones" [GHP22] by limiting the contribution size of the local updates. Additionally, it allows to incorporate incomplete work of the clients or to ignore non-responding clients without posing any risk of high perturbations to the global model. In comparison to FedAvg, FedProx does not derive a global model from the local models directly, however, it iteratively updates the global model based on the local updates limiting their impact by using the proximal term μ, see pseudo-code lines 4,6 in Algorithm 3.2.

Note that the algorithm does not require having exact solutions to the local objectives F_k at any step, but demands the local solution to be γ_k^t-inexact, see Definition 3.3.1. This requirement allows deriving convergence guarantees for FedProx for both convex and nonconvex functions; authors also empirically demonstrated the model's superiority in heterogeneous settings. Moreover, by setting the proximal term to 0, the model is reduced to FedAvg, however in that case the converge guarantees cannot be maintained.

Definition 3.3.1 (γ_k^t-inexact solution) *"For a function $h_k(w; w_t) = F_k(w) + \frac{\mu}{2}\|w - w_t\|^2$, and $\gamma \in [0, 1]$, we say w^* is a γ_k^t-inexact solution of $\min_w h_k(w, w_t)$, if $\|\nabla h_k(w^*, w_t)\| \leq \gamma_k^t \|\nabla h_k(w^*; w_t)\|$, where $\nabla h_k(w; w_t) = \nabla F_k(w) + \mu(w - w_t)$." [LSZ+18]*

Note that a smaller γ_k^t corresponds to higher accuracy.

Algorithm 3.2: FedProx, [LSZ+18]

```
  Input: $K, T, \mu, \gamma, w^0, N, p_k, k = 1; ...; N$
1 for $t = 0; ...; T-1$ do
2     Server selects a subset $S_t$ of $K$ devices at random (each device $k$ is chosen
        with probability $p_k$)
3     Server sends $w^t$ to all chosen devices
4     Each chosen device $k \in S_t$ finds a $w_k^{t+1}$ which is a $\gamma_k^t$-inexact minimizer of:
        $w_k^{t+1} \approx_w h_k(w; w^t) = F_k(w) + \frac{\mu}{2}\|w - w_t\|^2$
5     Each device $k \in S_t$ sends $w_k^{t+1}$ back to the server
6     Server aggregates the $w$'s as $w^{t+1} = \frac{1}{K}\sum_{k \in S_t} w_k^{t+1}$
7 end
```

3.3.3 FedYogi

Some works analyzing FedAvg have already highlighted its convergence challenges in heterogeneous networks. Aggregating local models with non-identically distributed data may result in s global model shift if the individual optima move too much away from the optimum of the whole network. Another problem lies in lack of adaptivity since in case of FedAvg, distributed SGD does not incorporate the information about past training rounds, which may result in slower training and thus in increased communication costs.

Reddi et al. [RCZ+21] suggest a technique to integrate the knowledge from previous communication rounds by extending FedAvg with adaptivity options similarly to momentum in SGD optimization. The suggested approach FedYogi benefits from optimization on both the client and the server side - similarly to FedAvg clients perform several local epochs E of SGD updates, but contrary to FedAvg, the server also applies additional gradient-based updates. In other words, FedYogi extends FedAvg by adding adaptive optimization also on the server size, and by setting respective parameters FedYogi can be reduced to FedAvg. This adaptive server's behavior may help reduce the number of global training rounds and thus improve significantly expensive communication.

The detailed training steps are described in the pseudo-code of Algorithm 3.3. Here, in comparison to FedAvg, not only a local learning rate η has to be set for local training on the client side, but also a global learning rate η_g for adaptivity on the server side has to be selected. Additionally, the parameter τ is used to manage the level of adaptivity - smaller τ values resulting in higher adaptivity.

To be noted that the authors suggest different adaptive approaches, i.e. in addition to FedYogi, also FedAdagrad and FedAdam are proposed, which differentiate from FedYogi by v_t calculation in line 11 of Algorithm 3.3.

Algorithm 3.3: FedYogi, [RCZ+21]

```
   Input: $K, T, \eta, \eta_g, E, w^0, v_{-1} \geq \tau^2, N, p_k, k = 1; ...; N$,
   decay parameters $\beta_1, \beta_2 \in [0, 1)$
1  for $t$ = 0;...; T–1 do
2      Server selects a subset $S_t$ of $K$ clients at random (each client $k$ is chosen with
         probability $p_k$)
3      Server sends $w^t$ to all chosen clients
4      for each client $k \in S_t$ in parallel do
5          Client $k \in S_t$ updates $w^t$ for $E$ epochs of SGD on $F_k$ with step-size $\eta$ to
             obtain $w_k^{t+1}$
6          $\Delta_k^t = w_k^{t+1} - w^t$
7      end
8      Each client $k \in S_t$ sends $\Delta_k^t$ back to the server
9      $\Delta_t = \frac{1}{|S_t|} \sum_{k \in S_t} \Delta_k^t$
10     $m_t = \beta_1 m_{t-1} + (1 - \beta_1)\Delta_t$
11     $v_t = v_{t-1} - (1 - \beta_2)\Delta_t^2 \text{sign}(v_{t-1} - \Delta_t^2)$
12     Server updates the $w$'s as $w_{t+1} = w_t + \eta_g \frac{m_t}{\sqrt{v_t} + \tau}$
13 end
```

3.3.4 qFedAvg

Heterogeneous client distributions pose challenges not only for convergence of the global model. A global model may perform well on the whole infrastructure in general, e.g. by achieving high average accuracy, however, it may deliver poor prediction results for a subset of clients In other words, there is no possibility to ensure a similar accuracy level across all individual clients in the network. Averaging local models, including calculating the weighted average, may result in a biased model favoring clients with more data or clients with "popular" distributions. Nevertheless, in some situations, optimization of the global model from the overall perspective may be insufficient, if similar prediction quality has to be ensured for all clients without exceptions. For example, in cross-silo settings, organizations would have the interest to participate in collaborative training, if they can profit from such training themselves. Also, in case of predictive maintenance, even if collaborative training is performed within the same organization by consolidating different units, the organization may be interested to have similar model performance across all units for better diversification and control of costs and resources. Thus, certain problems may require an increased degree of performance uniformity across the clients, which indicates the necessity for considering the fairness of the model.

To tackle this problem, Li et al. [LSBS19] proposed a fairness-driven federated optimization qFedAvg, where the clients with poor performance at a given time point are dynamically assigned with higher relative weights. Higher weights increase the impact of low-performing clients on the global model and so the global model is encouraged to seek less variation in performance across the clients. This approach was inspired by the α-fairness in Network Resource Allocation [LKCS10], where the orchestrator of the network can manage the level of fairness by setting different levels of α.

We will leave the theoretical background for the reader to study from the original paper [LSBS19] on their own. In Algorithm 3.4 we provide the detailed pseudo-code. We note that similarly to the *alpha*-fairness metric, the algorithm includes the parameter q, which allows to control the degree of fairness by emphasizing more the clients with higher local losses F_k. Higher q values imply superior fairness levels. Still, since an overall performance remains an essential goal of the global optimization, setting the parameter q forces the user to balance between the fairness of the model and the total model effectiveness.

The algorithm involves one more hyper-parameter L - Lipschitz constant, which in our experiments we set to $1/\eta$, similarly to the paper author.

Algorithm 3.4: qFedAvg, [LSBS19]

```
  Input: $K, E, T, q, 1/L, \eta, w^0, N, p_k, k = 1; ...; N$
1 for $t$ = 0;...; T−1 do
2     Server selects a subset $S_t$ of $K$ clients at random (each client $k$ is chosen with
        probability $p_k$)
3     Server sends $w^t$ to all chosen clients
4     Each client $k \in S_t$ updates $w^t$ for $E$ epochs of SGD on $F_k$ with step-size $\eta$ to
        obtain $w_k^{t+1}$
5     Each client computes:
          $\Delta w_k^t = L(w^t - w_k^{t+1})$
          $\Delta_k^t = F_k^q(w^t)\Delta w_k^t$
          $h_k^t = qF_k^{q-1}(w^t)\|\Delta w_k^t\|^2 + LF_k^q(w^t)$
6     Each client $k \in S_t$ sends $\Delta_k^t$ and $h_k^t$ back to the server
7     Server updates $w^{t+1}$'s as:
          $w^{t+1} = w^t - \frac{\sum_{k\in S_t} \Delta_k^t}{\sum_{k\in S_t} h_k^t}$
8 end
```

CHAPTER 4

Experiments

4.1 Data

Meaningful real federated datasets are problematic to collect due to data privacy and legal concerns [HLL+20b]. Moreover, predictive maintenance data are mostly considered private and thus are rarely disclosed for open access. We select three publicly available datasets, which are suitable for a classification task with a very high class imbalance. Our selected datasets are single collections of data. Thus, we simulate the clients by splitting existing data. Additionally, we include an evaluation on a real dataset received as a part of project *Interactive* carried out by the Austrian Institute of Technology[1]. This dataset shall represent a realistic federated scenario since we can allocate the data to individual clients semantically.

4.1.1 AI4I 2020 Predictive Maintenance Dataset (Synthetic)

The AI4I 2020 Predictive Maintenance Dataset[2] provided by Matzka [Mat20] is a synthetic dataset, which aims to represent real-world predictive maintenance data appearing in the industry. Hereinafter, we will address this dataset as **Synthetic**.

The dataset contains 6 features, which include one categorical variable for product type with three possible values and numerical variables for process temperature, air temperature, rotational speed, torque, and tool wear, as well as data for different types of failures. We restrict our analysis to a binary problem - the presence or absence of a failure, as the production process would fail in case of any type of failure.

The dataset consists of 10 000 data points, out of which 339 contain some kind of failure, resulting in a high class imbalance with 3.39% of samples having a positive class. We

[1]`https://www.interactive-project.info/`

[2]`https://archive.ics.uci.edu/ml/datasets/AI4I+2020+Predictive+Maintenance+Dataset`

use 80% of the data for training, 10% for validation, and 10% for testing. To tackle the data imbalance issue, we employ SMOTE oversampling technique for training data. We generate minority samples till the total number of failures account is 20% of the majority class size.

Recalling that different types of classification errors have significantly different contributions to the total maintenance cost, we set $c_1 = 1$ and $c_2 = 30$, as the author suggests 30 times higher cost for false negatives than for false positives.

We point out that the preprocessing, i.e. oversampling, is performed for each client separately in the case of local and federated training, without taking into account the data from other clients. The same methodology is applied to all other datasets to preserve data privacy. Obviously, in the case of central training, as we collect local data centrally, we execute all preprocessing steps also centrally utilizing the full collected dataset.

4.1.2 Air pressure system failures in Scania trucks

The Failures in Scania trucks dataset[3], provided by the company Scania CV AB for the Industrial Challenge 2016 on The 15th International Symposium on Intelligent Data Analysis (IDA), is a collection of different component failures in Scania trucks in everyday usage. The positive class, in this case, represents the failures, which are related to the air pressure system, while the negative class covers the failures not associated with the air pressure system. The dataset contains 60 000 data points for training, thereof only 1 000 records or 1.7% belong to the positive class. We use 20% of the training data for validation. For testing, an additional dataset is provided. It encompasses 16 000 records, out of which 2.3% of the positive class. Each data point is assigned 170 features, which represent operational measurements. The feature names are anonymized.

The dataset contains a lot of missing values. Therefore, we delete the features, where more than 70% of values are missing. For the rest, we impute the missing values using the median. We randomly remove 70% of the negative class observations and use SMOTE to upsample the positive class records to comprehend 20% of the data. We also standardize all values before training.

To evaluate maintenance cost, we again consider different types of false predictions, where a false positive would represent an unnecessary check by a mechanic and a false negative would mean missing a faulty truck and potentially resulting in a breakdown. As provided by the dataset author, we set the cost parameters $c_1 = 10$ and $c_2 = 500$.

This dataset was widely studied in the literature in terms of different machine learning approaches. However, only centralized training approaches were considered. The best-performing models resulted in a total cost of approximately 10 000 - 11 500 for the test set ([CN16]). This should give us an indication, of whether our model has a decent performance. However, we also accept models with a moderately higher total cost.

[3]https://archive.ics.uci.edu/ml/datasets/APS+Failure+at+Scania+Trucks

4.1.3 Hard Drive Reliability Sample

Since 2013, Backblaze provides daily data and summary statistics about the performance of hard drives in their data centers[4]. The detailed data includes serial number and model of the drive, drive capacity, 45 S.M.A.R.T. statistics [Wik] in raw and normalized form, and a failure indicator. These data are recorded daily and can be used for health monitoring of the hard drives by taking into consideration the timely development for predicting the remaining useful life. In order to have a binary classification problem, we will consider only the short-term health of hard drives and aim to identify the faulty records at the moment the measurements are taken.

The dataset includes different hard drive models, whose annualized failure rate may vary between 0.5% till 2%. Also, S.M.A.R.T. statistics may vary in meaning based on the manufacturer and the model, making this dataset a good candidate for modeling heterogeneity in a federated network.

Although almost 10 years of data are available, we will use only a snapshot of the data, collected in 2019. This snapshot contains over 30 million records, representing the data for more than 100 000 unique hard drives. In our experiments, we use a portion of the data, as for many models, e.g. Hitachi, Seagate, DELLBOSS, no failing hard drives were recorded in 2019. Moreover, some important S.M.A.R.T. characteristics, as suggested by Backblaze, such as 5, 187, 188, 197, and 198 are not available for many of these drives. Thus, we include in our analysis only ST drives, for which at least some failure instances are present in the data. Furthermore, we consider only a subset of features, which should be the most useful for predicting failures. In particular, we use these S.M.A.R.T. metrics: 3, 5, 7, 187, 188, 190, 194 197, 198, 199, either raw or normalized, as suggested by Amram et al. [ADTZ21].

During preprocessing, we remove approximately 200 000 records, which contain missing values (most of these records have no failure). Still, we are left with more than 30 million records, over 90 000 unique drives, thereof 1967 with a failure. We assign the drives to train/validation/test sets in 80%/10%/10% proportions. By splitting the data based on the drives we ensure that no drive appears in more than one set and so there is no data leakage. There are only a few failed hard drives, as the failure is indicated on the last day, while all preceding days represent no failure. To increase the percentage of the positive class in the training data, we modify the labels of the failed drives so that the last two days of a drive's life represent failure. By doing so we can double the number of positive class observations. On the other hand, we do not adjust more observations, as this may lead to an increased false alarm rate, i.e., false positive rate. Especially, since the deviations in S.M.A.R.T statistics that differentiate working drivers from failed ones are very small [AW]. At the same time, instead of simply downsampling the negative class, we select 4 random samples per good drive, which should be enough to maintain necessary information [ZWL+13].

[4] https://www.backblaze.com/b2/hard-drive-test-data.html

As a benchmark, we consider other works employing the Backblaze data. e.g. [ADTZ21], [SH18]. For example, Amram et al. achieved a 10% false alarm rate while identifying around half of the failing drives (50% failure detection rate). Shen et al detect almost all failures while ensuring only a 1.764% false alarm rate, however they analyzed hard drive models not present in our dataset. Overall, various investigations are often based on different time frames, isolate the experiments for a single drive submodel, or use different S.M.A.R.T. features, which make the surveys less comparable. Moreover, many studies use decision trees or random forest. Still, they provide some insights about meaningful effectiveness, which we could also expect and target.

We observed that most of the works dealing with hard drive data analyze both metrics: false alarm rate and failure detection rate without prioritizing one of them. Also, most investigations use prediction accuracy or f1-score and do not weigh the impact of type I or type II errors. Therefore, also in our analysis, we weigh the classes equally and set a cost for both false positives and false negatives to 1.

4.1.4 Automotive dataset from industry

The automotive industry presents an ideal real-world use case for the application of Federated Learning. First, components are standardized, and their production is distributed all around the globe, while the segmentation into multiple suppliers prohibits the exchange of data. Therefore, federated learning could provide a standard model while respecting the privacy concerns of the involved parties. Second, suppliers store production data for individual articles for up to five years due to quality assurance requirements. Thus, machine learning practitioners can access high-quality historical data.

The dataset originates from the production of gearbox components, in which prefabricated parts are first deformed and then combined using laser welding. During production, quality inspection engineers extract single articles for destructive testing. These detailed quality inspection results represent the target variables for this use case. The original dataset consists of 6230 samples with 178 features representing measurements such as position, force, and moment of a single part, and 237 quality metrics, considered as target variables.

For our experiments, we selected a single target variable that indicates whether an article passed the quality inspection. Thus we have again a binary classification task. The relevant data for this task consists of 4090 samples, which can be grouped into four variants of the same product. These variants, more precisely product types, represent the individual clients in our distributed scenario. This simulates the production of comparable products in different production facilities.

Based on the discussions with the business people, we set $c_1 = 5$ and $c_2 = 15$ for estimation of the total maintenance cost. Here the difference between type 1 and type 2 errors is lower than in the previous cases, as it shall consider that stopping the production line to perform unnecessary checks is a costly process as well. Also, such parameters reflect the specifics of the data collection process, as not every article in the production line is being inspected.

4.2 Client Simulation

While some manufacturing companies, their subsidiaries, or production units may produce standardized products, the manufacturing and maintenance processes may have different specifics across the companies. Also, companies may range in size, capabilities, production volumes, etc. Thus, we cannot always expect that the data gathered by individual entities are i.i.d. Therefore, in our analysis, we consider different scenarios, based on data distribution across the clients. On the one hand, we are interested in the behavior of i.i.d. data, which may simulate different production sites of a single enterprise. On the other hand, we explore non-i.i.d. settings, as they may often represent the situations described before.

Kairouz et al. [KMA+19] summarized non-IID scenarios from a statistical distribution perspective. For our investigations, feature distribution skew and quantity skew schemes are most meaningful. Feature distribution skew appears when the statistical distributions of clients are non-identical given the same feature space. This skew type could simulate variation in manufacturing processes, even though the same items are produced. Quantity skew is observed when different units possess different amounts of observations. Other types include Label distribution skew (e.g., different clients record different types of failures), Same label, different features (e.g., same failures, but caused by different reasons), and opposite, Same features, different label. Real-world data often incorporates a composite of these factors.

Thus, we consider three scenarios in our investigations:

- **I.I.D.:** In such case, we split the data across clients randomly in equal proportions;
- **NON-I.I.D. with feature distribution skew:** To simulate clients with different feature distributions, we employ principal components analysis. Principal components are known to capture the summarized information stored in different features. Thus, we calculate the principal components of our data and assign the observations to clients according to the value of their first principal component. To ensure that we simulate isolated feature distribution skew without incorporating also data quantity skew, we allocate each client a similar portion of data, defined by the quantiles of the principal component. For example, in the case of 10 clients, we allocate 10% of existing data to the first client, but we allocate those observations, whose principal component values lie within the range of the first 10-quantile. Analogically, the second client is allocated with the data having their principal component within the second 10-quantile, and so on. In Figure 4.1 we see t-SNE representation of (a) Synthetic dataset and (b) Scania trucks data split into 10 clients employing PCA. Even though we do not see pure disjoint clusters, we can

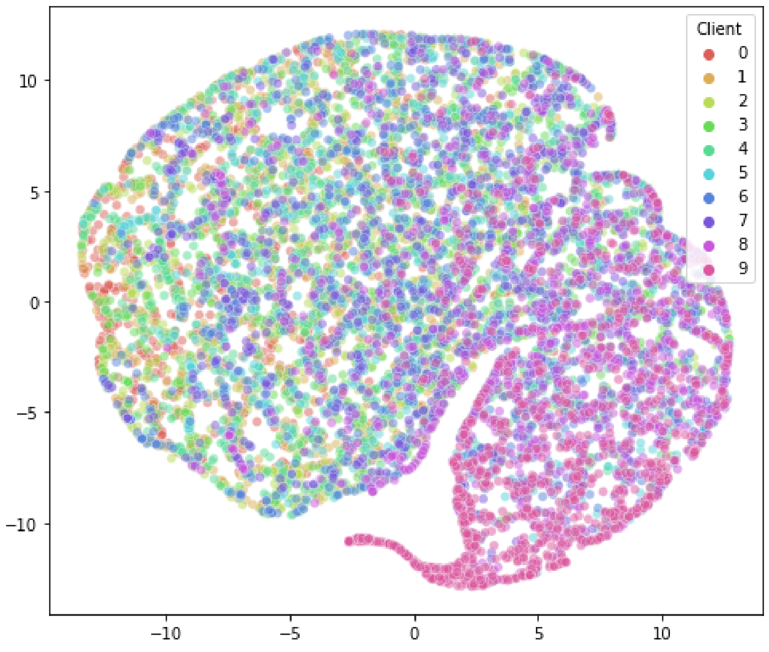
still capture some differences in the distributions of the clients. In the case of the Synthetic dataset, Figure 4.1a, we observe that most data of clients 8-9 are displayed in the lower right quadrant, while clients 0-4 tend to be represented on the left half of the graph. However, clearer differences can be recognized for Scania trucks data, Figure 4.1b. Here, clients 0-2 seem to form some clusters at the sides, while clients 7-9 lean towards the center of the plot. We note that for both datasets, the first principal component accounts for 30% of the variance in the data only, thus it is not surprising that we do not see clear separations between the clients. Similar to our approach, Verma et al. [VWJ+19] in their experiments, split the data based on the first two principal components by assigning the quadrants of the PCA 2D plot to the respective clients.

- **NON-I.I.D. with quantity skew:** Yurochkin et al. [YAG+19] suggested using Dirichlet distribution for simulations of skew in federated learning and this method is now widely used in research for experiments with different skew types. In such case, clients are allocated with proportions of data according to Dirichlet distribution, i.e. the proportions are sampled $p_k \sim Dir(\beta)$ and the clients $k \in K$ are assigned with $n_k = n \cdot p_k$ random samples. The parameter $\beta > 0$ controls the imbalance of the distribution - smaller β values impose a higher concentration of data at a few clients. Still, if β is chosen too small, some clients would end with only some records. In Figures 4.2 and 4.3 we see examples of such data split for Synthetic and Scania trucks datasets respectively. Like previously, 10 clients are simulated for each dataset. In part (a) we see equally-sized clients, which would occur in the case of I.I.D. or NON-I.I.D. with feature distribution skew data splits. While in part (b) we see clients with quantity skew, i.e. with varying dataset sizes, obtained according to Dirichlet distribution. For Synthetic dataset we used $\beta = 1.3$ and for Scania trucks $\beta = 1.1$. Even though in the literature the parameter β is often set to 0.5, we use higher β values due to high global data imbalance. In other words, we aim to have at least some observations from both negative and, more important, positive classes.

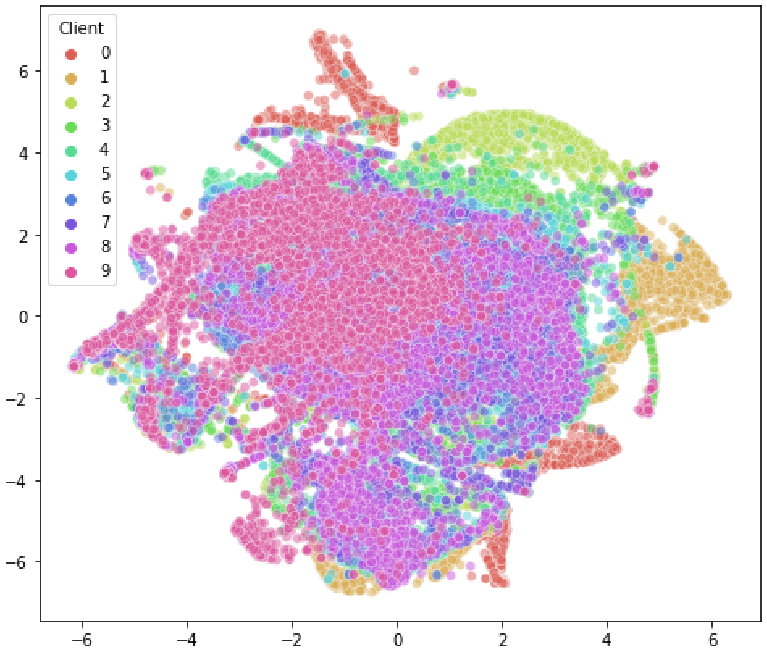
4.3 Infrastructure Simulation

Typically, federated learning is considered in cross-device (e.g. mobile devices) or cross-silo (organizations) settings. According to Kairouz et al. [KMA+19], the federated learning domain for organizations has the following characteristics. Typically, 2 to 100 organizations, i.e. clients, are involved in the system. The clients are almost always available and each client may participate in each training round. The clients are reliable, i.e. they are not expected to fail or drop out during the training round (no stragglers). Each client has an identity and can be specifically addressed by the system. The primary bottleneck in the cross-silo settings might be either computation or communication.

Thus, for our experiments, we consider a rather small number of clients. We analyze different settings based on the available data:

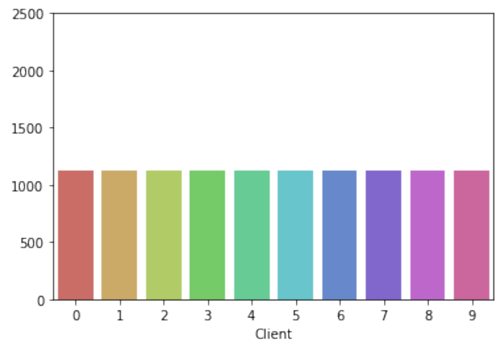


(a) Synthetic dataset

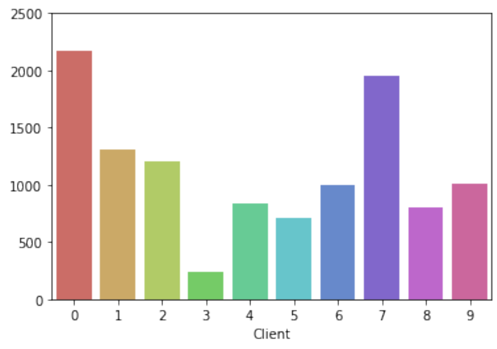


(b) Scania trucks

Figure 4.1: t-SNE representation of clients' data

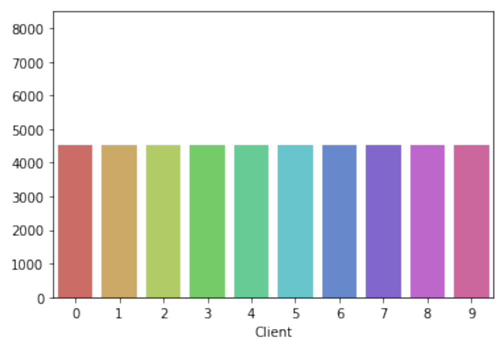


(a) Equally sized clients

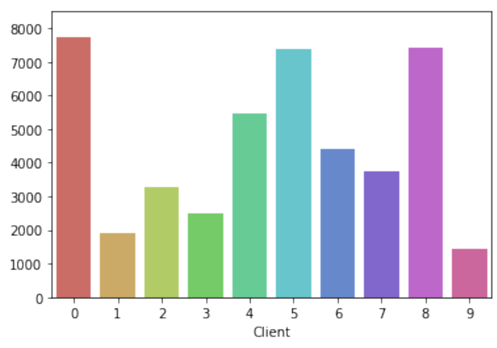


(b) Clients with quantity skew

Figure 4.2: Client data sizes, Synthetic dataset



(a) Equally sized clients



(b) Clients with quantity skew

Figure 4.3: Client data sizes, Scania trucks

- *Synthetic dataset*: we divide the data in 5 / 10 / 15 clients. We assign data to individual clients based on different data splitting strategies, as described in Section 4.2 - client distributions are I.I.D. / NON-I.I.D. with feature skew / NON-I.I.D. with quantity skew.

- *Scania trucks dataset*: we split the data in 10 / 20 / 30 clients. Again, we use the same 3 strategies for data assignment to the clients.

- *Hard drive dataset*: this dataset contains information about the hard drive model for each observation. We have the data for 11 models: ST4000DM000, ST500LM030, ST6000DX000, ST8000DM002, ST8000NM0055, ST8000DM005, ST8000DM004, ST10000NM0086, ST12000NM0007, ST12000NM0117, ST12000NM0008. Using this information, we can simulate clients with heterogenous distributions with feature distribution skew, but also to some extent with quantity skew, as various models have an unequal number of observations. Based on the amounts of available data, we group similar models and simulate 3 clients: ST10000NM0086, ST12000NM0007, ST12000NM0117, ST12000NM0008, ST8000DM002, ST8000NM0055, ST8000DM005, ST8000DM004 and ST4000DM000, ST500LM030, ST6000DX000. Additionally, we consider again i.i.d. clients and non-i.i.d. with quantity skew, simulated according to the same methodology as before.

- *Automotive dataset*: 4 predefined clients are considered, where every client represents a certain type of product.

For each of the described settings, we perform local training, centralized training, and federated training. In the federated scenario, we train 4 algorithms FedAvg, FedProx, qFedAvg, and FedYogi. Every client shall participate in every training round, i.e. we simulate no stragglers.

We note that we execute data preprocessing steps for different training scenarios on the appropriate level. In the cases of local and federated training, we preprocess the data on the client level for each client separately. We do not operate with any global data, as we should also not have any access to local data. In the case of the centralized training, we first collect all data centrally and then do the preprocessing for the full dataset at once. Hence, we incorporate the knowledge of all local data.

4.4 Parameter tuning

Tuning models and their parameters is an inevitable part of solving most machine learning problems. In the case of federated learning, tuning is an even more comprehensive process than in the case of centralized training. In addition to variations in local models, in federated networks, we have to adjust a number of new parameters specific to federated algorithms.

In further subsections, we discuss the features, which we have to consider before training and comparing all models. We tune all parameters for every dataset separately. However, we examine a single setting for each of the datasets:

- *Synthetic dataset*: 10 clients with feature distribution skew (same clients, as in Figure 4.1a).
- *Scania trucks dataset*: 10 clients with feature distribution skew (same clients, as in Figure 4.1b).
- *Hard drive dataset*: 3 clients with feature distribution skew (data split according to the hard drive model).
- *Automotive dataset*: 4 defined clients.

In all other scenarios, we use the best parameters of the respective datasets.

4.4.1 Base model and local learning rate

All our considered approaches require finding an underlying model that would solve the respective optimization objective. For example, in a federated setting, the objective is defined in Equation 1.1. Our main goal is to carry out comparisons between selected training approaches in terms of convergence, communication, and fairness. Still, we aim for an appropriate underlying model to be trained further. To make the results comparable among different training paradigms, we select a single underlying model architecture, which shall be trained within each scenario individually from scratch.

Finding a proper model within federated training is rather complicated due to numerous parameters, which have to be tuned. Therefore, we make a quick evaluation of different model architectures in the centralized framework. Even though in real federated infrastructures this approach would not be achievable, we consider it sufficient for our investigations.

After experiments with different MLP architectures and learning rates, we find these models satisfactory, as their performance is similar to the state of the art found in the literature:

- *Synthetic dataset*: MLP with 3 linear layers - 50, 20 and 10 neurons respectively; ReLU activation function; an output layer with 2 neurons; Learning rate = 1e-5.
- *Scania trucks dataset*: MLP with 3 linear layers - 200, 100 and 50 neurons respectively; ReLU activation function; an output layer with 2 neurons; Learning rate = 1e-4.
- *Hard drive dataset*: MLP with 4 linear layers - 400, 200, 100 and 50 neurons respectively; ReLU activation function; an output layer with 2 neurons; Learning rate = 1e-5.
- *Automotive dataset*: MLP with 3 linear layers - 35, 15, 15 neurons respectively; ReLU activation function; an output layer with 2 neurons; Learning rate = 1e-4.[5]

4.4.2 Number of local epochs in federated training

A number of local epochs E per single training round can have a significant impact on the performance of federated learning algorithms and in some cases may lead to non-improving plateau behavior or even divergence of the model [MMR$^+$17]. Thus, we explore the impact of the amount of local work per round on the overall effectiveness of the models.

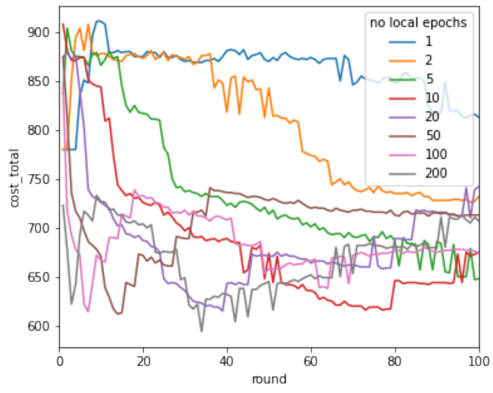
[5]Obviously, for this dataset we do not have any indication of the state-of-the-art performance. Therefore, we additionally trained random forest and SVM in a central manner to check that our model is acceptable

For each dataset, we train a separate model employing the Federated Averaging method, as FedAvg should be more sensitive to a local epochs choice than, for example, FedProx [WYS+20].

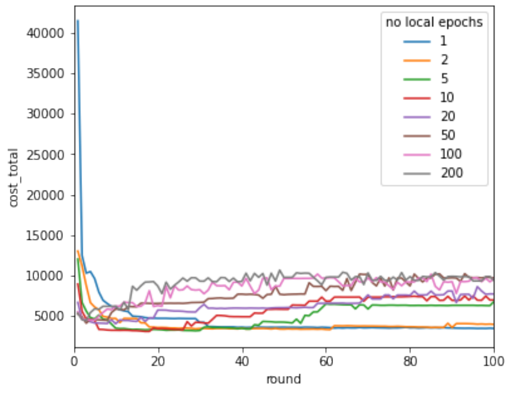
For Synthetic and Scania trucks datasets, we consider a candidate set $E \in \{1, 2, 5, 10, 20, 50, 100, 200\}$, while for Hard drives data we tune the number of local epochs over the grid $E \in \{1, 2, 5, 10, 20\}$. In Figure 4.4 we can see the performance of different algorithms for our three cases in terms of the total maintenance cost, summed over all clients. For Scania trucks, Figure 4.4b, we select $E = 10$ as an optimal parameter, as it shows faster improvements and reaches the best total cost value of all considered options. Its effectiveness starts to decrease after 30 global rounds, while shorter local training (1-2 local epochs per round) seems to have more consistent performance for more global rounds. Still, very few local epochs per round would imply more costly training in terms of communication. Due to similar reasons, we select $E = 5$ for Hard drives data, as training longer than for 5 epochs locally results in a soon overfitting of the model, see Figure 4.4c. On the other hand, there is almost no difference in the initial convergence speed between 5, 10, and 20 local epochs. For the Synthetic dataset, we cannot decide about the optimal number of local epochs so easily, see Figure 4.4a. It seems like 200 local epochs per round would deliver the best total cost value in round 33. However, the training process is not stable in this case and after reaching the optimum peak the model diverges rapidly. In general, we prefer a lower amount of local epochs to avoid such spikes, as in cases of 50, 100, and 200 local epochs. Therefore, we additionally inspect the effect of local epochs on FedProx. In Figure 4.5, we can see the behavior of different FedProx models, i.e., with different values of the parameter μ, for Synthetic data. Here, we prefer slower local training with 10 epochs per round rather than quickly reaching the optimum with an instant decrease in performance. The lower values of the number of local epochs result in much slower training.

4.4.3 Proximal term μ in FedProx

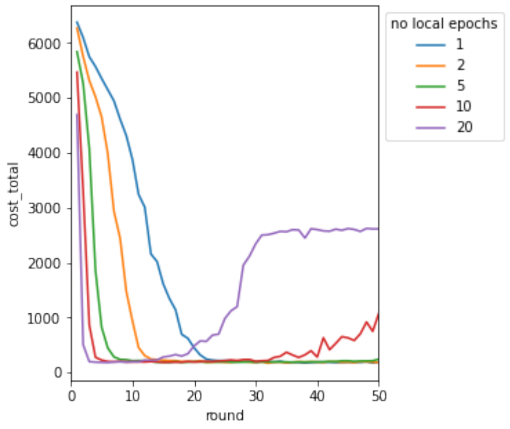
FedProx has two main parameters: the number of local epochs and the proximal term μ. A high number of local epochs E, as discussed in the previous section, may lead to unstable training. The proximal term μ may be seen as a re-parameterization of E, which may help to avoid drift from a global optimum and increase the stability of the model [LSZ+18]. On the other hand, high μ values may slow down the training progress. Such behavior can be clearly seen for Synthetic and Hard drives data, Figures 4.6a and 4.6c respectively. In the case of the Synthetic dataset, small μ values result in model divergence after the 80th epoch. We note that the smaller the size of μ, the closer FedProx gets to FedAvg - FedAvg is a single case of FedProx with $\mu = 0$. Thus, we select $\mu = 0.1$. In the case of the Hard drives dataset, there is very little difference for μ values between 0.001 and 0.1. Therefore, we select the highest μ of the available meaningful options to allow more difference from FedAvg. We set μ to 0.1 for Hard drives data as well. For Scania trucks data, we select $\mu = 0.001$, as it is the only parameter delivering the total cost much below 4000.



(a) Synthetic dataset



(b) Scania trucks

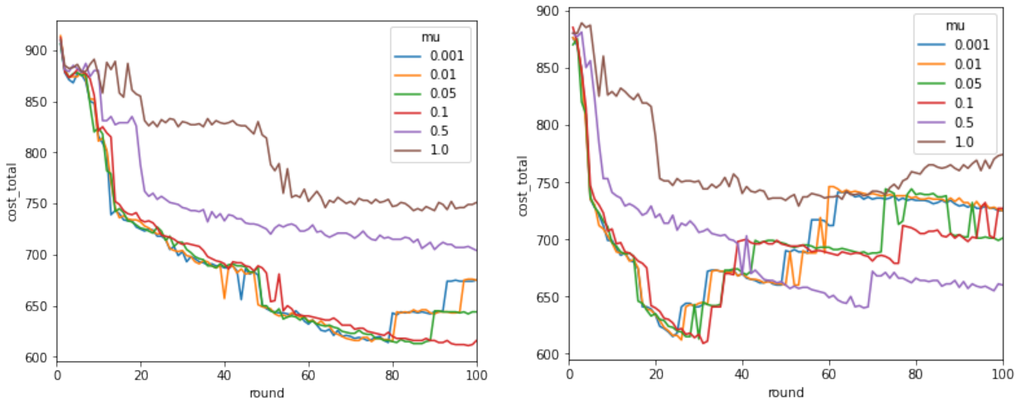


(c) Hard drives

Figure 4.4: Parameter tuning: number of local epochs

4.4.4 Fairness parameter q in qFedAvg

The fairness parameter q shall control the degree of fairness in qFedAvg. This parameter influences the training process by assigning higher relative weights to the clients with higher loss. Still, increasing the fairness requirements not only may have detrimental effects on the model effectiveness, but also slow down the speed of convergence [LSBS19]. We observe this phenomenon for Scania trucks data, Figure 4.7b. Higher q values do indeed result in much higher total cost. However, lower q parameters do not indicate almost any differences in terms of fairness, expressed here as the entropy of the clients' F-scores. We select here a medium fairness parameter $q = 1e\text{-}5$. For Hard drives data, Figure 4.7c, we also observe very few differences with respect to the both effectiveness and fairness. Thus, we choose here a highest fairness parameter value $q = 1.0$. In the case of Synthetic data, we give priority to the model effectiveness over the fairness and opt for $q = 5e\text{-}12$, as the models with only slight improvements in the fairness result in much higher total cost, see Figure 4.7a.



(a) 10 local epochs (b) 20 local epochs

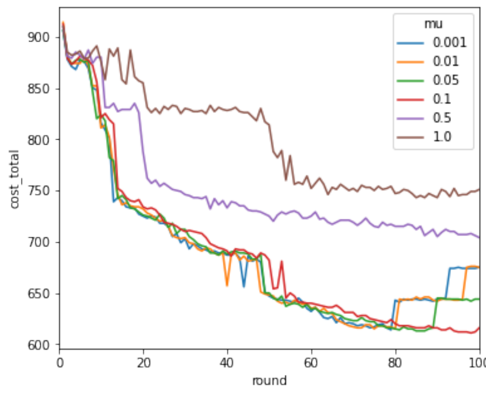
Figure 4.5: Influence of number of local epochs on FedProx performance, Synthetic dataset

4.4.5 Global learning rate η_g and degree of adaptivity τ in FedYogi

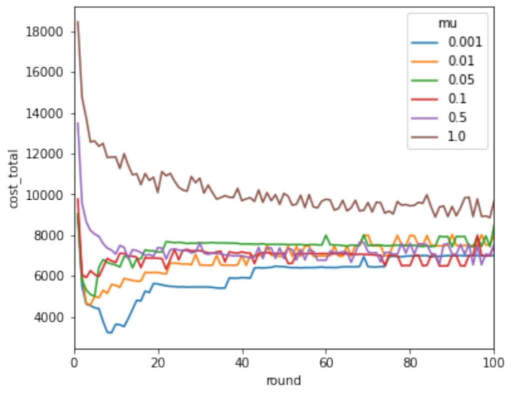
For FedYogi we carry out a grid search for a global server learning rate and degree of adaptivity over the grid $\eta_g, \tau \in \{1e\text{-}05, 0.0001, 0.001, 0.01, 0.1\}$. A higher global learning rate may increase the convergence speed. On the other hand, it may also contribute to a divergence of the model. Smaller τ values imply a higher degree of adaptivity. The highest disparity among different global learning rates and degrees of adaptivity can be observed for Scania trucks data, Figure 4.8b. Lower $\eta \in \{1e\text{-}05, 0.0001\}$ significantly slow down the training, while higher $\eta \in \{0.01, 0.1\}$ lead to instability of the models. Also, a proper selection of τ drives an increase in the convergence speed. Thus, for this dataset we use a parameter pair $\eta = 0.001$ and $\tau = 1e\text{-}05$. For other datasets, we barely observe any differences, see Figures 4.8a and 4.8c. We make the following choice: for Synthetic dataset $(\eta, \tau) = (0.1, 1e\text{-}05)$, for Hard drives data $(\eta, \tau) = (0.001, 0.001)$.

4.5 Final Setup

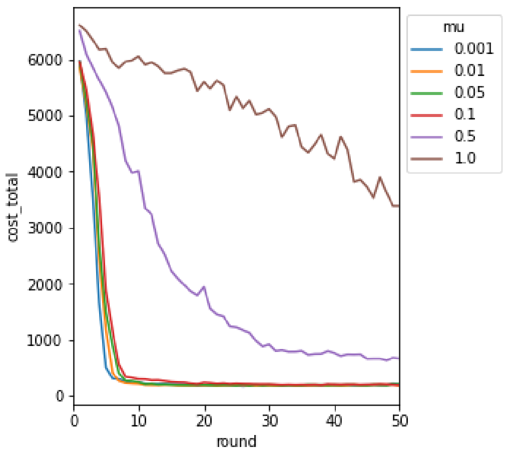
Based on all the above steps, we define a set of scenarios for our experiments as described in Table 4.1. As we used the validation sets for parameter tuning, we evaluate the performance of the trained models on the test sets and present the results in the next chapter.



(a) Synthetic dataset

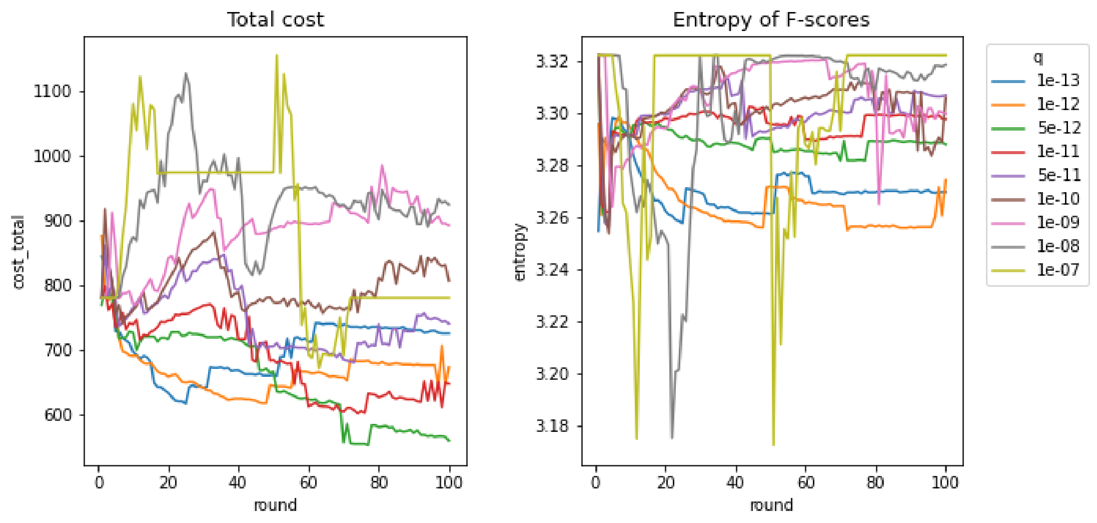


(b) Scania trucks

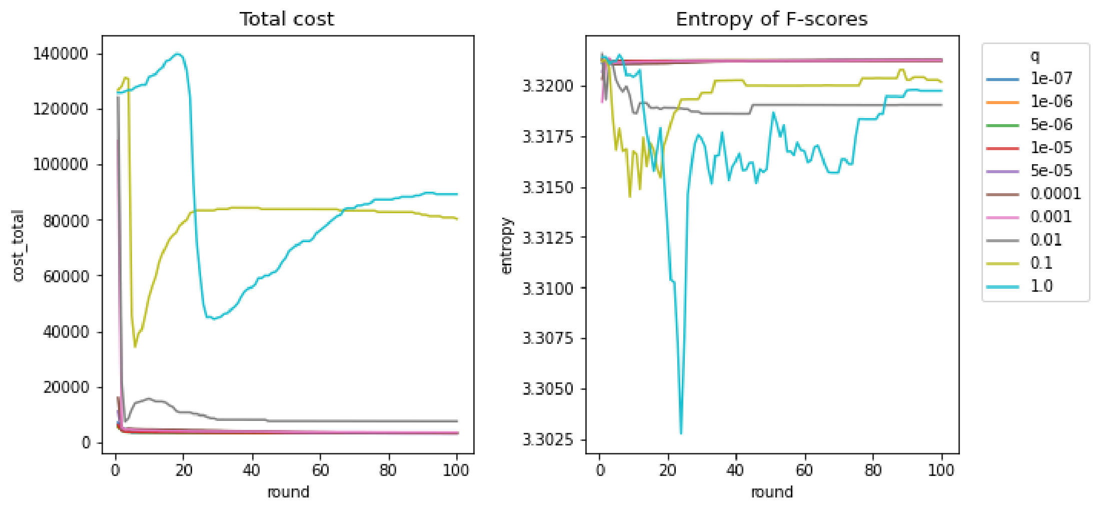


(c) Hard drives

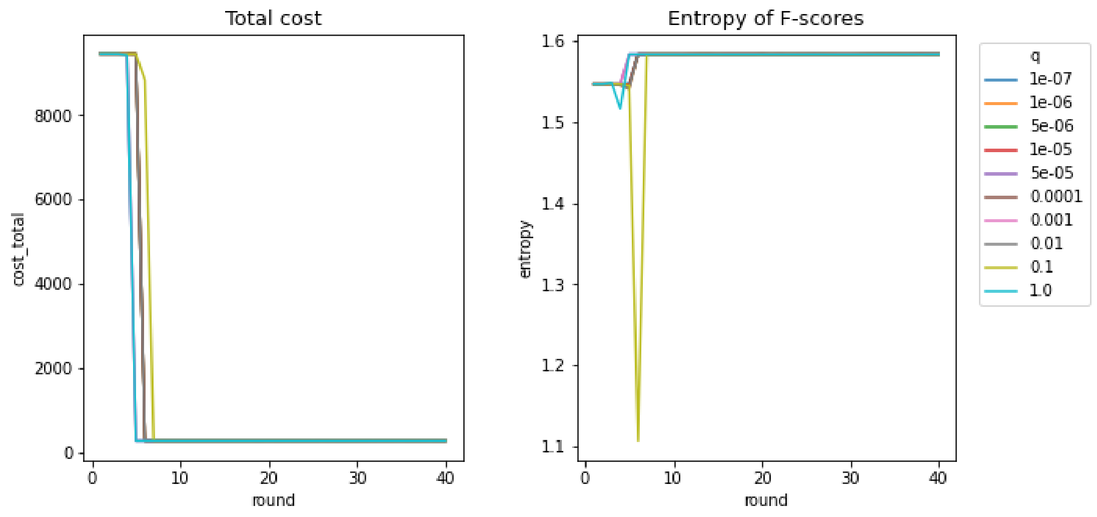
Figure 4.6: Parameter tuning: proximal term μ



(a) Synthetic dataset

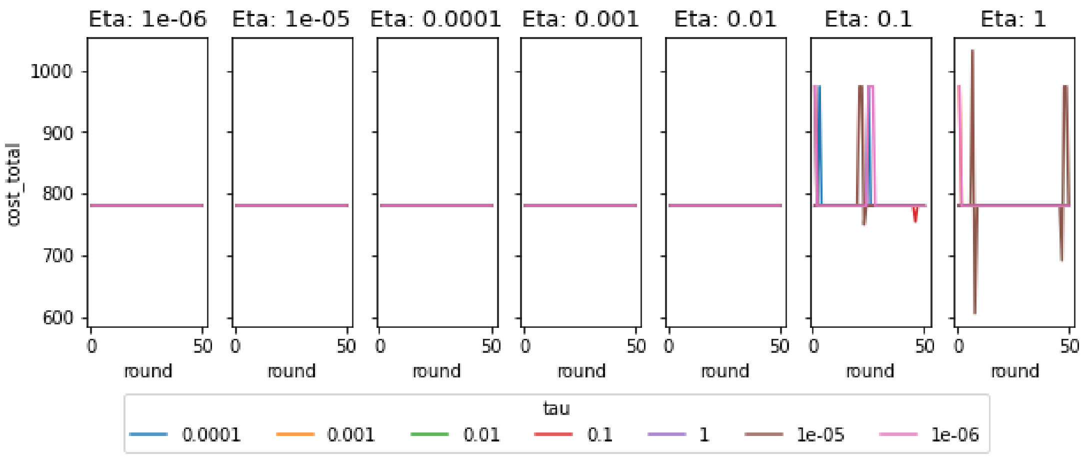


(b) Scania trucks

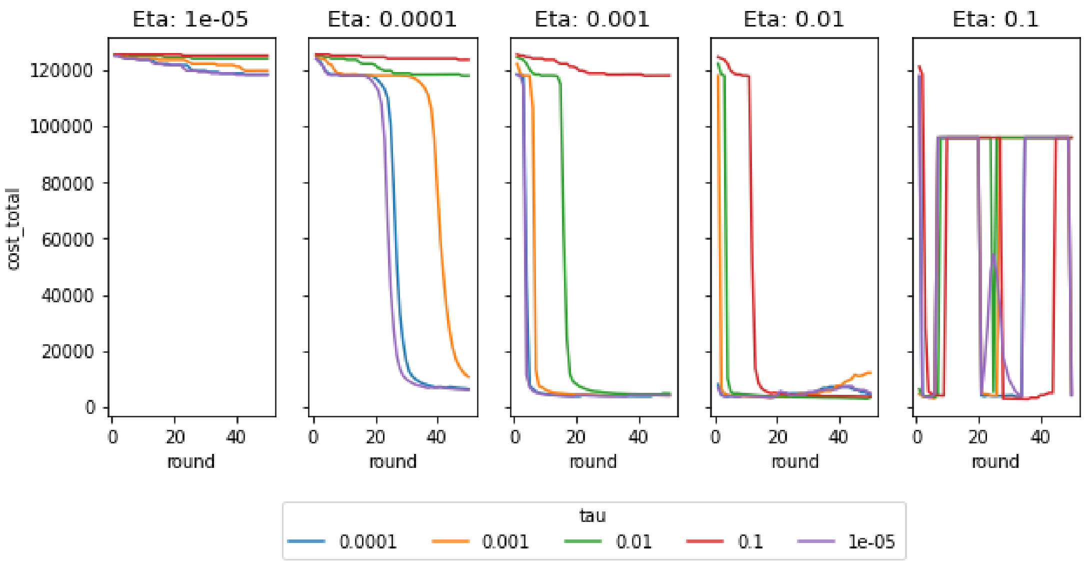


(c) Hard drives

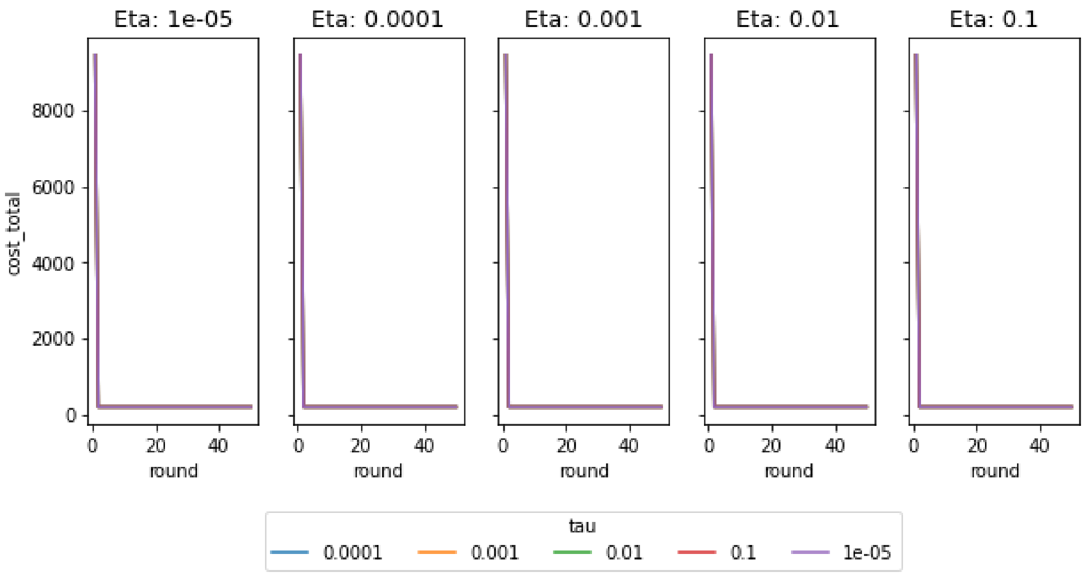
Figure 4.7: Parameter tuning: fairness degree q



(a) Synthetic dataset



(b) Scania trucks



(c) Hard drives

Figure 4.8: Parameter tuning: global learning rate η_g and degree of adaptivity τ

Table 4.1: Experiment settings

Parameter	**Synthetic dataset**	**Scania trucks**	**Hard drives**	**Automotive dataset**
β for F-beta score	30	50	1	3
Number of clients K	5 / 10 / 15	10 / 20 / 30	3 / 10 / 20	4
Client data simulation	i.i.d. / non-i.i.d.	i.i.d. / non-i.i.d.	i.i.d. / non-i.i.d. / by product	by product
Concentration parameter β (quantity skew)	1.1	1.3	1	-
Base model MLP architecture	[50, 20, 10]	[200, 100, 50]	[400, 200, 100, 50]	[35, 15, 15]
Local learning rate η	1e-05	1e-04	1e-05	1e-04
Batch size	128	128	1024	75
Number of local epochs E	10	10	5	20
Number of global rounds T	100	100	50	500
Proximal term μ (FedProx)	0.1	0.001	0.1	0.05
Fairness parameter q (qFedAvg)	5e-12	5e-05	1	5e-06
Global learning rate η_g (FedYogi)	0.1	0.001	0.001	0.1
Degree of adaptivity τ (FedYogi)	1e-05	1e-05	0.001	1e-05

CHAPTER 5

Results

In order to make inferences about the performance of different training approaches (central training, local training, federated learning), we tested each approach for 4 datasets and evaluated the results of 28 scenarios using the parameters as described in Table 4.1. We calculated 4 defined metrics for a test set of each scenario - total cost and F-beta score of the infrastructure, the number of communication rounds, and entropy of individual clients' F-scores[1]. Based on modeled scenarios and resulting metrics, we answer the research questions.

5.1 Effectiveness of federated learning algorithms

In figures 5.1 (a)-(d) we see calculated average f-score among clients for each data splitting approach, i.e. data across clients is i.i.d., has feature distribution skew or quantity skew. The strokes in the boxplots represent different numbers of clients in the infrastructure, e.g. for Scania trucks 10/20/30 clients. We note that we display only the best-achieved scores (at any moment of the training).

We see that in all cases the simplest federated algorithm FedAvg, which averages the individual clients' models, achieves a good and in some cases, the best performance in comparison to other algorithms - for best performance see Synthetic dataset, Figure 5.1a, and Hard drives dataset, Figure 5.1c, even when the data across the clients are skewed (feature distribution or quantity skew).

FedProx reaches very similar prediction accuracy to FedAvg in terms of average F-score, still performing worse in the case of Hard drives and Automotive datasets, see Figures 5.1c and 5.1d respectively. Looking back at the definition and purpose of FedProx, see Algorithm 3.2, we remind that FedProx was introduced to improve the stability of a

[1]For Hard drives data we analyze mainly F-score since the cost parameters are both equal to 1, i.e. $c_1 = c_2 = 1$

training process and leverage data, but, more importantly, systems heterogeneity. In our case, we did not analyze systems heterogeneity. This is consistent with our modeled cross-silo setting (limited number of participating organizations), where the clients are mostly reliable and there should be no stragglers. Thus, in our cases, the gains of employing FedProx are rather limited.

More complex aggregation algorithms, qFedAvg and FedYogi, performed much poorer on Synthetic and Hard drives data, than FedAvg and FedProx. Still, for the Scania trucks dataset, FedYogi obtained the highest average F-score in almost all modeled scenarios (except for 10 clients in the non-i.i.d. setting with feature distribution skew). Similarly, qFedAvg achieved the best performance for the Automotive dataset. Here we have to note that the underlying data, which come from the industry, represent a realistic federated scenario with a limited amount of data and having differing distribution across clients. In particular, client number 4 has a significantly different distribution than the other three clients. With the help of qFedAvg, we were able to slightly improve the total performance by imposing the requirement for fairness.

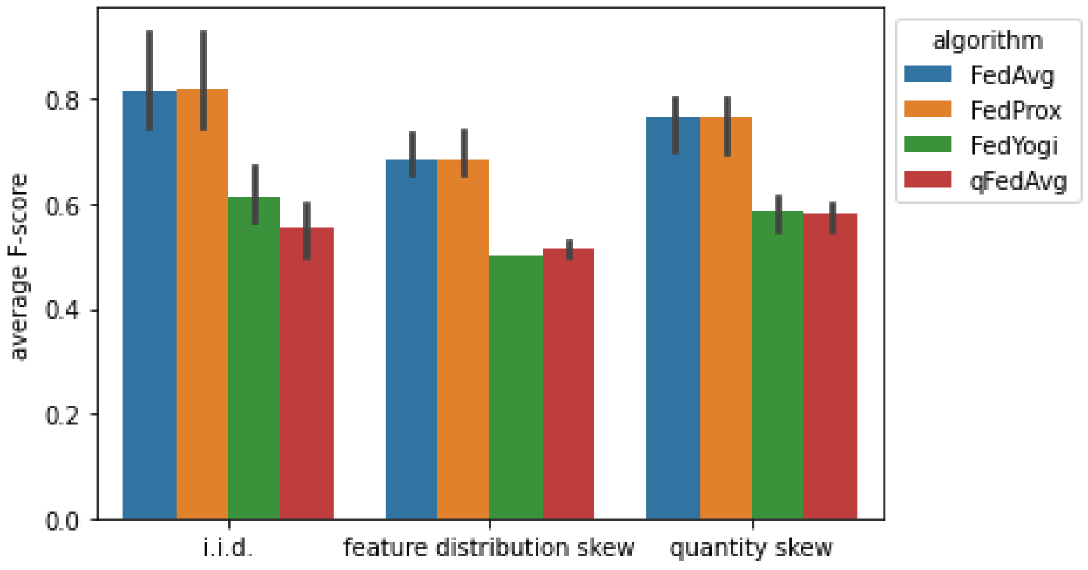
5.2 Communication cost and fairness of federated learning algorithms

Now we analyze the communication cost and the fairness of the algorithms at the moment, when they achieved their best effectiveness, as in the section 5.1.

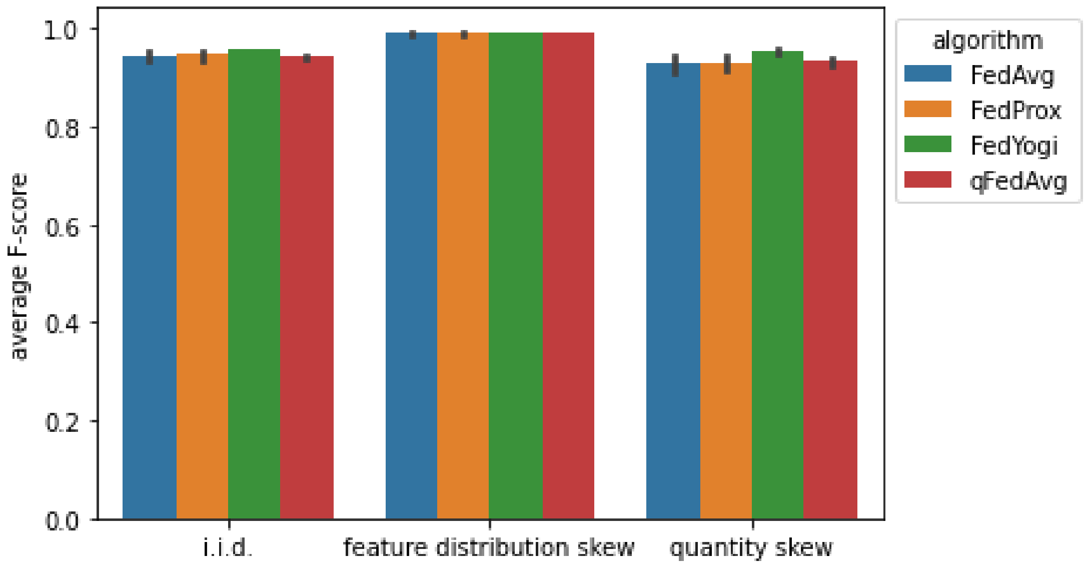
We observe that FedAvg and FedProx in the case of the Synthetic dataset and Scania trucks require a very similar number of global rounds to achieve their local optima, see Figure 5.2 (a), (b). In Appendix, we can also see that for these two datasets, the training progress of FedAvg and FedProx is almost the same. Recalling the definition of FedProx, see Algorithm 3.2, FedProx's main difference compared to FedAvg is the proximal term μ. The closer this term gets to 0, the more similar FedProx is to FedAvg. Thus, we may suspect that we selected μ too small. However, during the parameter tuning process, we aimed for the highest μ without having much impact on the performance and training speed, therefore we consider parameters as suitable.

For the Synthetic dataset, qFedAvg and FedYogi achieved their results much faster, however, their effectiveness is much lower than FedAvg and FedProx. For Scania trucks, in the case of i.i.d. and non-i.i.d. with quantity skew FedYogi required slightly more communication rounds to achieve its optimum, but it also outperformed other algorithms in terms of average F-score. Similarly in the non-i.i.d. with feature distribution skew scenario, where FedYogi outperformed other algorithms. However, in this case, it took many more communication rounds to achieve this performance. qFedAvg required a similar number of global rounds as FedYogi but reached slightly lower effectiveness.

In the case of Hard drive data, FedYogi and qFedAvg have a much lower number of global rounds, but they show poor performance in terms of F-scores. FedAvg and FedProx

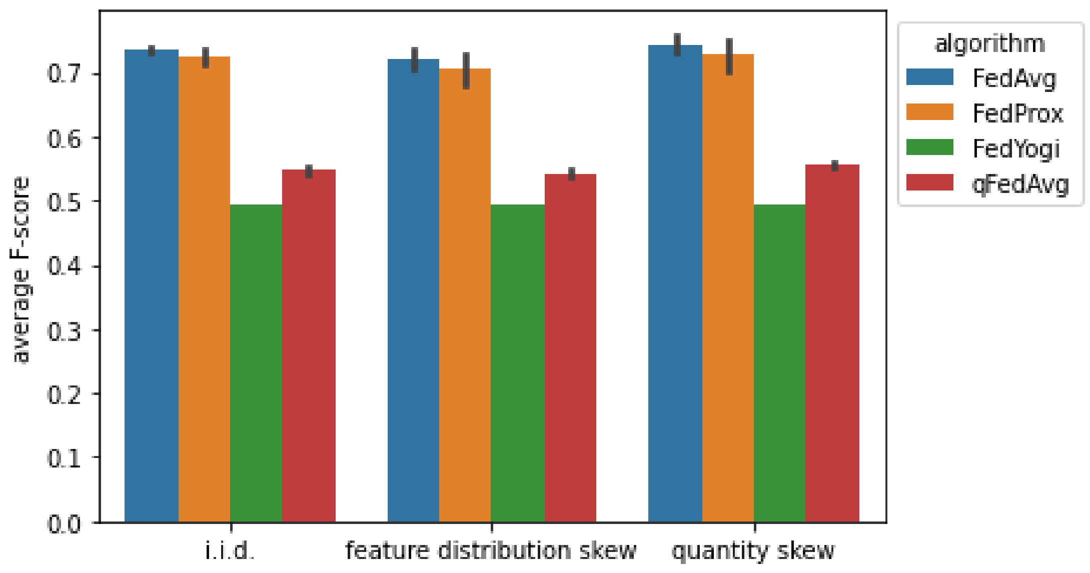


(a) Synthetic dataset

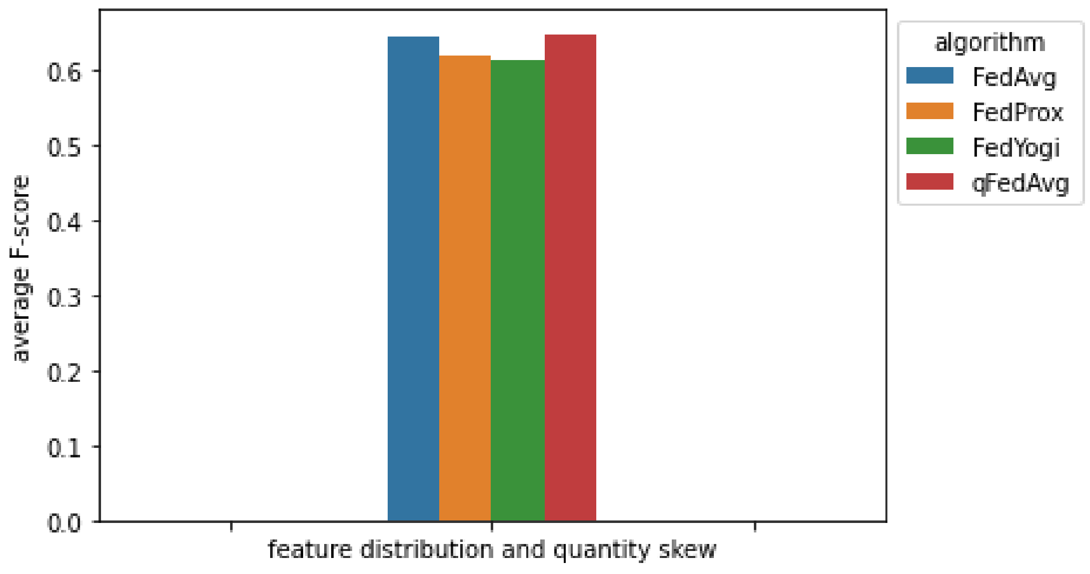


(b) Scania trucks

Figure 5.1: Average F-score by dataset and client simulation method



(c) Hard drives



(d) Automotive dataset

Figure 5.1: Average F-score by dataset and client simulation method (cont.)

accomplish similar performance, but FedAvg reaches its best values slightly faster. This can also be seen in Appendix, Figures A.8, A.9, A.10.

Finally, for the Automotive dataset, FedAvg and qFedAvg have similar training speeds, i.e. number of communication rounds, achieving top performance. While FedProx and FedYogi lag in terms of both effectiveness and communication cost.

Analogically to the communication cost, we mainly analyze the fairness of the models, which have satisfactory effectiveness. That means that we do not consider the fairness of qFedAvg and FedYogi for Synthetic and Hard drives datasets, as they have much lower effectiveness.

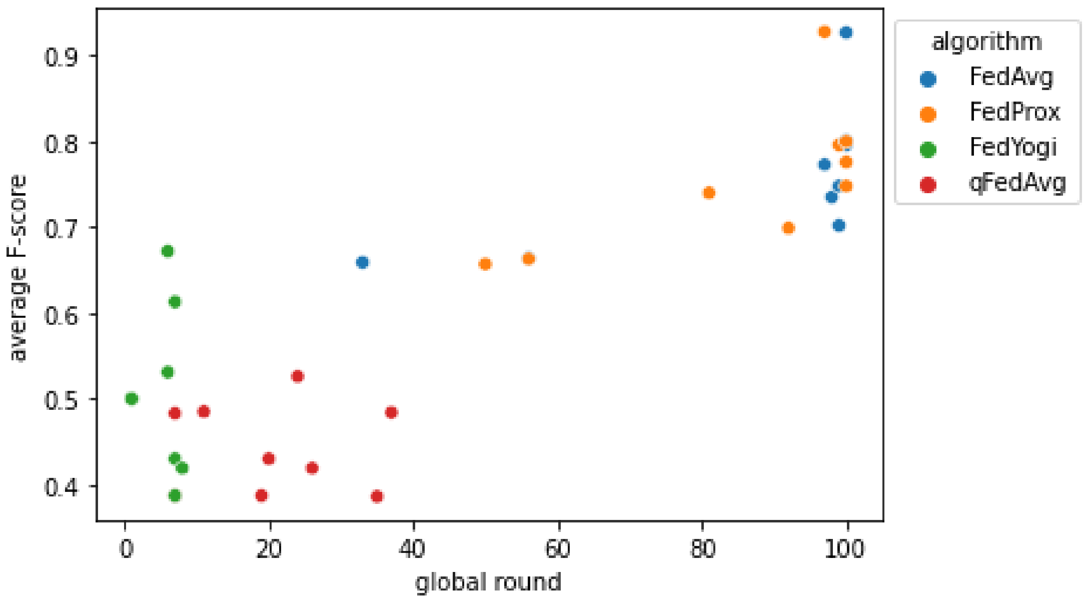
In the case of the Synthetic dataset, FedAvg and FedProx have very similar effectiveness, but also entropy of the individual clients' F-score is almost identical. With exception of the non-i.i.d. case with feature distribution skew with 5 and 10 participating clients - here FedAvg is slightly fairer (see also Figure B.3 in Appendix).

In the case of Scania trucks, the fairness level is again very similar or in some cases identical across different training methods. Here we also point out that our chosen fairness metric turns up to be suboptimal, as the differences in the entropy values are considerably low (e.g. starting at the 3rd or even 5th decimal place) and we have to examine the distributions of F-scores in detail. In Figure 5.3, we can see that:

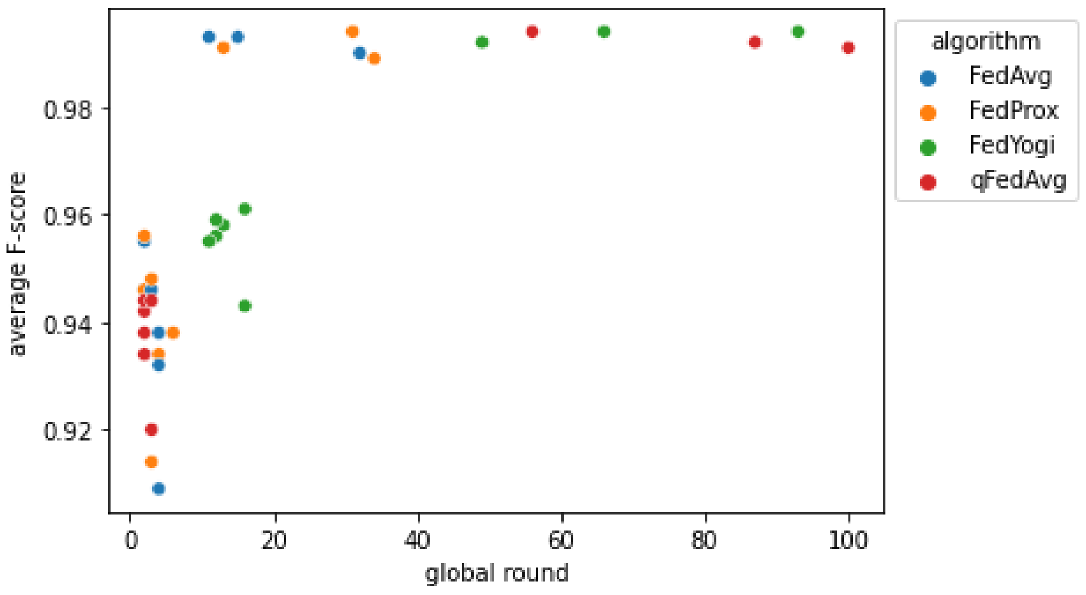
- case (a): FedYogi is fairer (distribution more skewed) than FedAvg or FedProx, while respective entropy values are 3.32178 vs 3.32176 or 3.32175. qFedAvg is the least fair, even though it should ensure more fairness by the definition of the algorithm.
- case (b): here qFedAvg looks the fairest, however, its effectiveness is the lowest among all approaches (highest total cost).
- case (c): in this scenario, qFedAvg and FedYogi have quite similar fairness levels, but FedYogi has superior effectiveness.
- case (d): here again, qFedAvg seems to be the fairest. Also, in this case, qFedAvg delivers the best effectiveness.
- case (e): here the higher fairness of qFedAvg and FedYogi is obvious, also fairness values differ more - 4.32168 (qFedAvg) vs 4.31686 (FedAvg). In this case, FedYogi achieves the best effectiveness, being also the fairest model, while qFedAvg performs the worst in terms of the total cost.

In the case of Hard drives, similarly to the Synthetic dataset, FedAvg and FedProx have similar fairness and prediction accuracy levels.

For the Automotive dataset, qFedAvg has slightly higher fairness, i.e. entropy value. It also outperforms other models. Here we remind again that client 4 has dissimilar

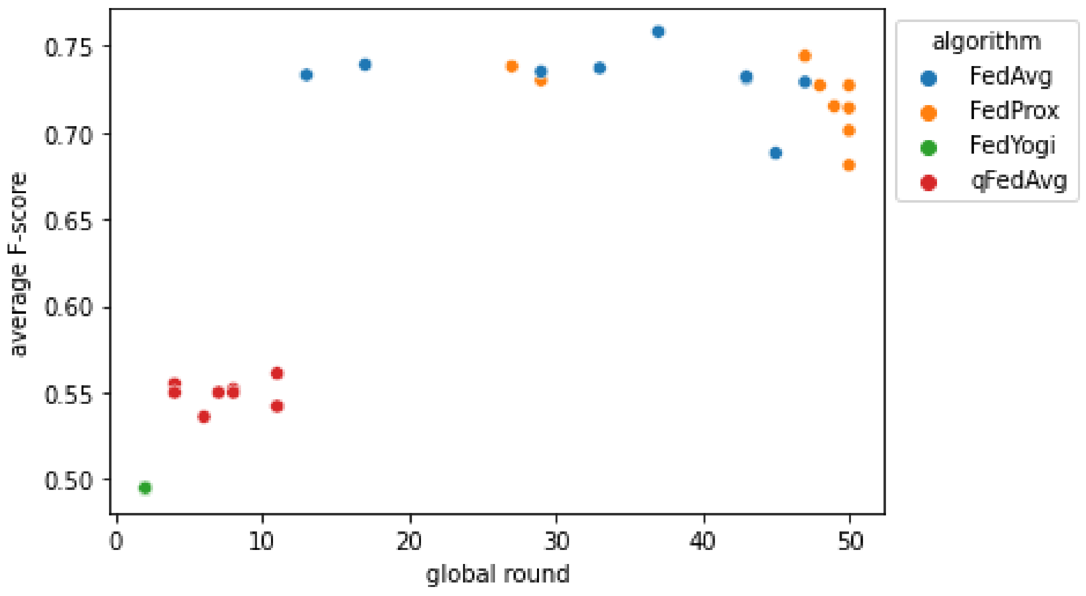


(a) Synthetic dataset

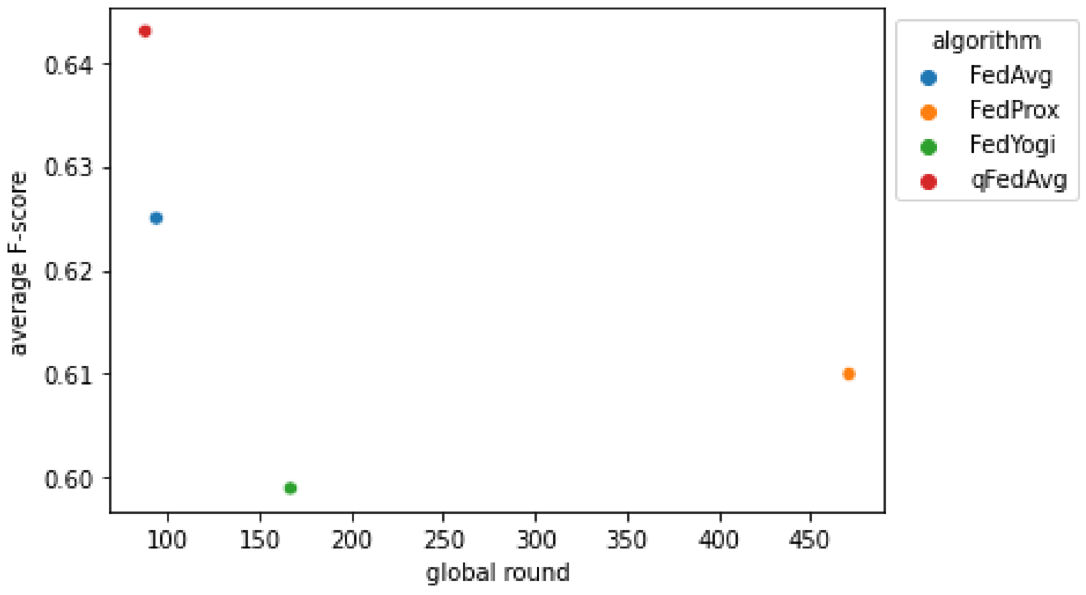


(b) Scania trucks

Figure 5.2: Number of communication rounds till optimum



(c) Hard drives



(d) Automotive dataset

Figure 5.2: Number of communication rounds till optimum (cont.)

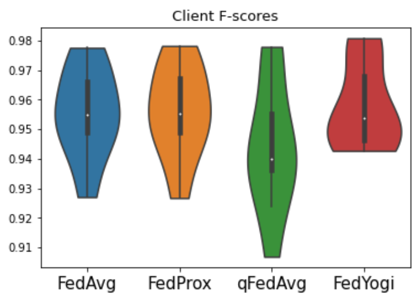
distribution than the other 3 clients and thus the fairness criterion may help to improve the total performance by controlling for fairness level.

Table 5.1: Synthetic dataset, entropy of the F-scores (fairness)

Data distribution	**Number of clients**	**Training method**			
		FedAvg	FedProx	qFedAvg	FedYogi
i.i.d.	5	2.3203	2.3203	2.3218	2.3148
	10	3.2755	3.2757	2.9998	2.9998
	15	3.8626	3.8626	3.7002	3.6806
non-i.i.d. with feature distribution skew	5	2.2925	2.2903	2.3219	2.3219
	10	3.2893	3.2663	3.3219	3.3219
	15	3.8657	3.8661	3.8898	3.9069
non-i.i.d. with quantity skew	5	2.2761	2.2760	1.9998	2.2465
	10	3.2961	3.2961	3.1681	3.1681
	15	3.8262	3.8288	3.5848	3.8072

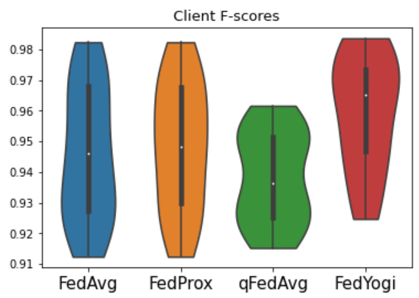
Table 5.2: Automotive datasets, entropy of the F-scores (fairness)

Data distribution	**Number of clients**	**Training method**			
		FedAvg	FedProx	qFedAvg	FedYogi
non-i.i.d. with feature distribution and quantity skew	4	1.9819	1.9857	1.9901	1.9838



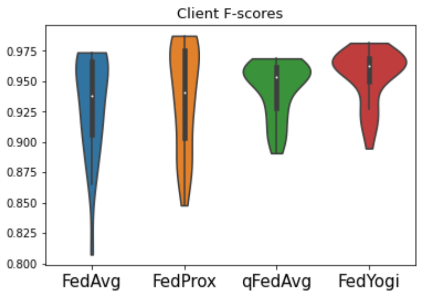
	FedAvg	FedProx	qFedAvg	FedYogi
f-scores entropy	3.32176	3.32175	3.32158	3.32178
round	2	2	2	13
total cost	14660	14510	19420	14300

(a) 10 clients, i.i.d.



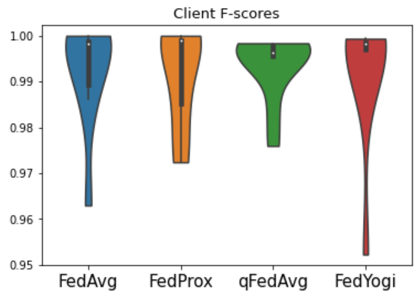
	FedAvg	FedProx	qFedAvg	FedYogi
f-scores entropy	4.32156	4.32159	4.32175	4.32167
round	3	3	2	12
total cost	19340	18350	19800	14180

(b) 20 clients, i.i.d.



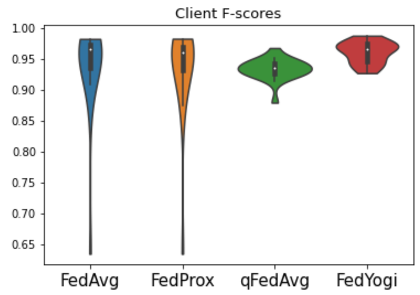
	FedAvg	FedProx	qFedAvg	FedYogi
f-scores entropy	4.90553	4.90545	4.90648	4.90651
round	4	6	3	11
total cost	24140	23940	18650	14840

(c) 30 clients, i.i.d.



	FedAvg	FedProx	qFedAvg	FedYogi
f-scores entropy	3.32184	3.32185	3.32189	3.32178
round	11	13	56	49
total cost	2450	3060	2060	2420

(d) 10 clients, feature distribution skew



	FedAvg	FedProx	qFedAvg	FedYogi
f-scores entropy	4.31686	4.31676	4.32168	4.32169
round	4	4	2	16
total cost	16700	17940	19140	12150

(e) 20 clients, quantity skew

Figure 5.3: Fairness and communication cost of FL algorithms, Scania trucks, selected scenarios

Table 5.3: Scania trucks, entropy of the F-scores (fairness)

Data distribution	**Number of clients**	**Training method**			
		FedAvg	FedProx	qFedAvg	FedYogi
i.i.d.	10	3.3218	3.3218	3.3216	3.3218
	20	4.3216	4.3216	4.3218	4.3217
	30	4.9055	4.9055	4.9065	4.9065
non-i.i.d. with feature distribution skew	10	3.3218	3.3218	2.3219	3.3218
	20	4.3218	4.3218	4.3218	4.3218
	30	4.9067	4.9067	4.9068	4.9068
non-i.i.d. with quantity skew	10	3.3212	3.3211	3.3216	3.3215
	20	4.3169	4.3168	4.3217	4.3217
	30	4.8983	4.9018	4.8982	4.8988

Table 5.4: Hard drives dataset, entropy of the F-scores (fairness)

Data distribution	**Number of clients**	**Training method**			
		FedAvg	FedProx	qFedAvg	FedYogi
i.i.d.	3	1.5849	1.5849	1.5840	1.5850
	10	3.3187	3.3190	3.3187	3.3219
	20	4.3118	4.3114	4.3165	4.3219
non-i.i.d. with feature distribution skew	3	1.5820	1.5833	1.5821	1.5850
	10	3.3188	3.3188	3.3187	3.3219
	20	4.3116	4.3112	4.3165	4.3219
non-i.i.d. with quantity skew	3	1.5841	1.5839	1.5840	1.5850
	10	3.3149	3.3164	3.3165	3.3219
	20	4.3085	4.3108	4.3047	4.3219

5.3 Effectiveness of various training methods

One of the main tasks of our analysis was to find out, whether federated learning approaches can be a good alternative to classic central or local training. Thus, we examine the total cost for each of our scenarios to answer this question. We present the total cost for each of the modeled settings (for Hard drives data we analyze F1-score instead) in the Tables: Synthetic dataset 5.5, Scania trucks 5.6, Hard drives 5.7 and Automotive dataset 5.8.

For the Synthetic dataset, federated learning algorithms showed in most scenarios a poorer performance than central and local training approaches. Thus, in this case, in the absence of the possibility to collect the data centrally, training the individual models locally would be preferred. For Scania trucks data, federated learning approaches outperformed both central and local training methods with quite significant differences. For Hard drives data, federated learning algorithms outperformed both central and local training approaches in most cases. However, in this case, the difference between the best federated models and the models trained locally or on centrally collected data are smaller, thus it would be worth evaluating more infrastructure characteristics, like costs and others, before introducing additional complexity by employing federated learning. For the Automotive dataset, federated learning showed lower effectiveness than central and local training techniques, however, the differences are not high. Recalling that client 4 has a very different distribution, this may introduce some noise into the global model trained in a federated manner. Thus, a possible approach would be to try a mixed procedure combining local and federated training.

Table 5.5: Synthetic dataset, total maintenance cost

Data distribution	**Number of clients**	**Training method**					
		Central training	Local training	FedAvg	FedProx	qFedAvg	FedYogi
i.i.d.	5	<u>**101**</u>	247	159	158	966	609
	10	<u>**101**</u>	228	300	296	966	966
	15	<u>**101**</u>	349	399	399	957	910
non-i.i.d. with feature distribution skew	5	<u>**101**</u>	165	526	514	966	1020
	10	<u>**101**</u>	222	662	665	966	1020
	15	<u>**101**</u>	218	653	654	974	1020
non-i.i.d. with quantity skew	5	<u>**101**</u>	240	241	241	965	761
	10	<u>**101**</u>	273	320	321	962	962
	15	<u>**101**</u>	301	358	368	960	607

Table 5.6: Scania trucks, total maintenance cost

Data distribution	Number of clients	Training method					
		Central training	Local training	FedAvg	FedProx	qFedAvg	FedYogi
i.i.d.	10	16550	17420	14660	14510	19420	**<u>14300</u>**
	20	16550	25060	19340	18350	19800	**<u>14180</u>**
	30	16550	28140	24140	23940	18650	**<u>14840</u>**
non-i.i.d. with feature distribution skew	10	16550	4960	2450	3060	**<u>2060</u>**	2420
	20	16550	3080	2130	2000	2790	**<u>1960</u>**
	30	16550	5160	3480	3680	2570	**<u>1920</u>**
non-i.i.d. with quantity skew	10	16550	21240	15670	15650	18610	**<u>13850</u>**
	20	16550	21740	16700	17940	19140	**<u>12150</u>**
	30	16550	34140	25970	24790	19030	**<u>12640</u>**

Table 5.7: Hard drive dataset, average F1-score

Data distribution	Number of clients	Training method					
		Central training	Local training	FedAvg	FedProx	qFedAvg	FedYogi
i.i.d.	3	0.730	0.729	**<u>0.739</u>**	0.730	0.550	0.495
	10	0.730	0.690	**<u>0.735</u>**	0.727	0.550	0.495
	20	**<u>0.730</u>**	0.646	0.729	0.714	0.542	0.495
non-i.i.d. with feature distribution skew	3	**<u>0.730</u>**	0.697	0.688	0.681	0.536	0.495
	10	0.730	0.688	**<u>0.737</u>**	0.727	0.550	0.495
	20	0.730	0.646	**<u>0.732</u>**	0.715	0.542	0.495
non-i.i.d. with quantity skew	3	0.730	0.735	0.733	**<u>0.738</u>**	0.555	0.495
	10	0.730	0.698	**<u>0.758</u>**	0.744	0.552	0.494
	20	0.730	0.656	**<u>0.731</u>**	0.701	0.561	0.495

Table 5.8: Automotive dataset, total maintenance cost

Data distribution	**Number of clients**	**Training method**					
		Central training	Local training	FedAvg	FedProx	qFedAvg	FedYogi
non-i.i.d. with feature distribution and quantity skew	4	<u>**2465**</u>	2485	2500	2570	2515	2575

CHAPTER 6

Conclusion

In our work, we analyzed the applicability of federated learning approaches to typical problems in the industry - predictive maintenance and defect detection. We evaluated the performance of various federated learning algorithms in terms of their effectiveness, communication costs, and fairness to represent different aspects of business problems, like maintenance costs, training complexity / IT infrastructure costs, and allocation of costs to different units.

We selected 4 datasets, which depict industrial data with a typical pattern where failures or defects appear on a rare basis. Obviously, this is very expected from the business side. Still, from a machine learning perspective, this poses numerous challenges for modeling very imbalanced data, where missing an erroneous item results in a high cost (e.g. high cost to replace a faulty part), yet checking each and every item makes the maintenance process itself very costly.

Furthermore, as there is a lack of real federated datasets from the industry accessible to the scientific community, we had to split available data artificially to model federated infrastructures. When modeling the individual clients, we aimed to approximate realistic federated networks by considering three scenarios: a) data across clients are i.i.d., b) data across clients are non-i.i.d. with feature distribution skew, and c) data across clients are non-i.i.d. with quantity skew.

Utilizing this background, we focused on answering three research questions:

- ***RQ1:*** How effective are selected aggregation algorithms for our predictive maintenance/defect detection task?
- ***RQ2:*** What is the relationship between the effectiveness of the aggregation algorithm and the communication cost and fairness of the model?
- ***RQ3:*** To what extent can the federated learning approach replace the centralized and /or local training without major losses in model effectiveness?

We demonstrated that each of our selected federated algorithms (FedAvg, FedProx, qFedAvg, and FedYogi) may be a good choice, depending on a concrete use case. Moreover, in many of our tested cases, FedAvg achieved decent effectiveness, and thus we can add our work to the list of empirical evaluations, showing the good performance of FedAvg. Still, we should not discard other methods as unsuitable for predictive maintenance tasks, as in some cases they could be beneficial in terms of effectiveness, communication costs, or fairness. Furthermore, we showed that even though FedProx is designed to be handy when dealing with different kinds of heterogeneity, it indicated to be less practical in our scenarios, as we modeled cross-silo settings with all clients participating in every round. We also did not consider systems heterogeneity.

Secondly, we estimated communication costs and fairness of different aggregation algorithms. In most of our analyzed cases, we did not encounter any major differences, and the models showing similar effectiveness would often have comparable communication costs and fairness. In order to have a deep dive into the differences, one would need to extend our research and model the impact of communication cost and fairness to assess the results objectively, whether these minor differences would be impactful for the business or not.

Finally, we evaluated the applicability of federated learning to training models on imbalanced data by comparing the performance of federated to classic central and local training approaches. In the case of central training, we collect all data centrally and train a single central model that shall be used by all clients. In the case of local training, clients train individual models using only their data, without profiting from the experience of others. While in the case of federated learning, one global model is trained, which incorporates the knowledge of each participating client, yet without accessing clients' individual data. We showed that in some situations, federated learning may outperform both central and local training approaches (e.g. Scania trucks and Hard drives datasets). Federated learning takes advantage of each client's individual situation but still allows for indirect cooperation between the clients, in this way helping them to leverage their disadvantages (e.g. low amounts of data in case of local training) and not generalizing too much (e.g. in comparison to training central model on merged data and considering each client being part of a single distribution). On the other hand, in other cases (e.g. Synthetic and Automotive datasets), especially when the clients' individual distributions are too diverse or clients possess too little data, it is preferable to train a single model, when a central collection of data is possible, or to let the clients train their models independently.

6.1 Future Work

We analyzed the suitability of federated learning to solve distributed predictive maintenance and defect detection problems employing very imbalanced data. We provided some insights about the performance of federated learning, still, there are some limitations in our study. Our investigation focuses primarily on the evaluation of federated learning in different data heterogeneity settings. However, a typical federated infrastructure is represented not only by data heterogeneity but also by systems heterogeneity. In other words, the performance of federated algorithms is also dependent on the individual systems and the network, and these aspects should be explored as well.

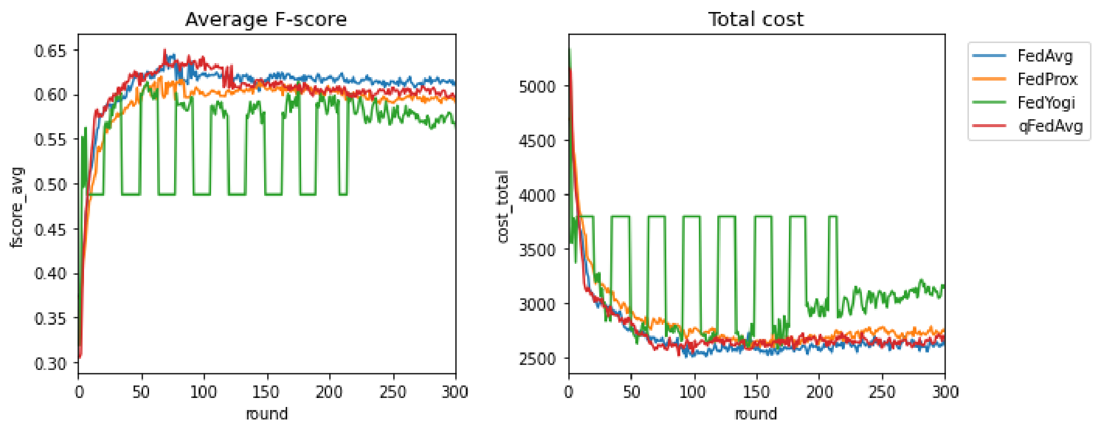
Another limitation in our work is originating from the lack of available federated methods employing other models than neural networks. Much research is focused on solving computer vision or natural language processing tasks, thus investigating different approaches to be used with deep neural networks. Though, there are much fewer works exploring and developing federated learning algorithms suitable for classic machine learning algorithms, like the random forest, SVM, etc, which could be useful, when modeling tabular data. We mention several existing approaches in Section 2.2, however as already mentioned before they miss proof of concept by the community. Also, in our study, we omitted the investigation of personalized approaches, see Section 2.2 for examples. As discussed for the Automotive dataset, one client had a very distinct distribution from the other three, thus this client may have profited from a personalized approach.

Finally, we executed our research on non-federated data, which we had to split manually according to our defined scenarios. Still, these methods are only proxies for real-world problems involving a mixture of different skew types and more complex data patterns. So explorations with real federated datasets would be advised.

APPENDIX A

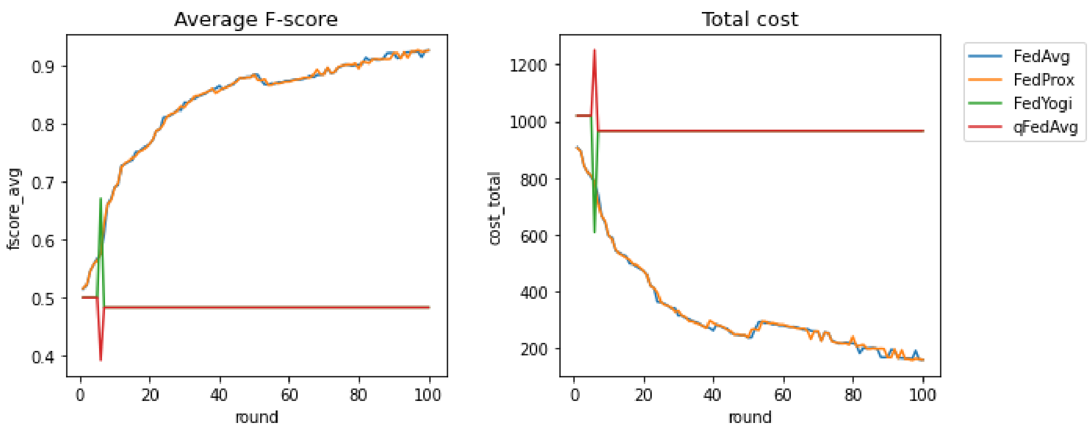
Training progress of federated learning algorithms

In Figures A.2, A.3, A.4, A.5, A.6, A.7, A.8, A.9, A.10, A.1 we display the development of average F-score and total cost of the respective test sets through the training for all scenarios.

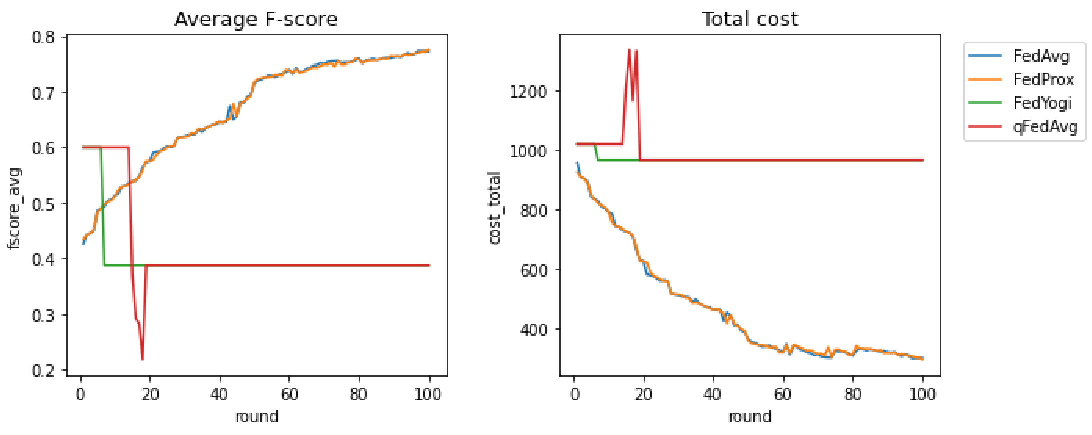


(a) 4 clients

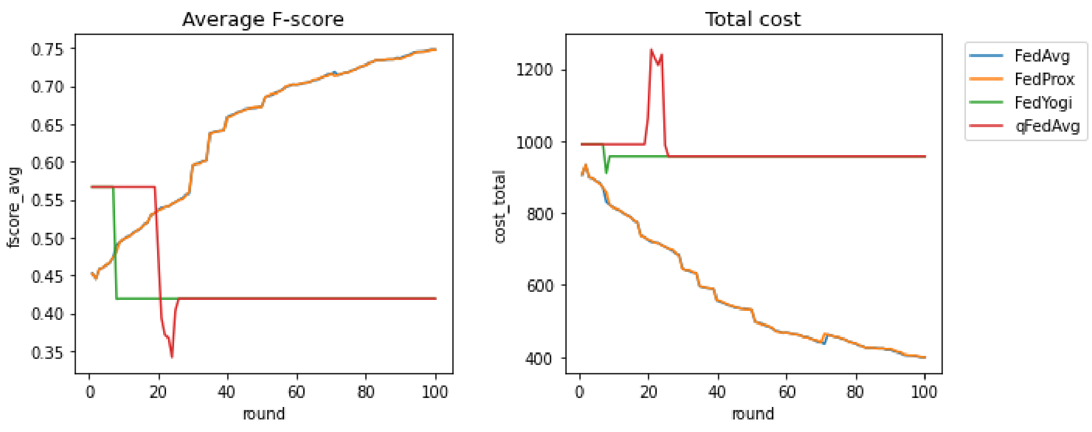
Figure A.1: Training of FL algorithms, Automotive dataset, feature distribution and quantity skew case



(a) 5 clients

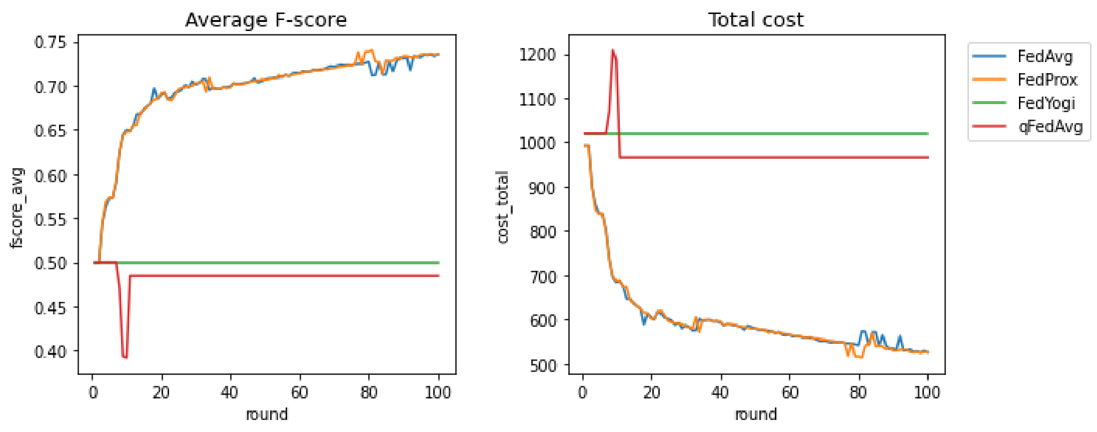


(b) 10 clients

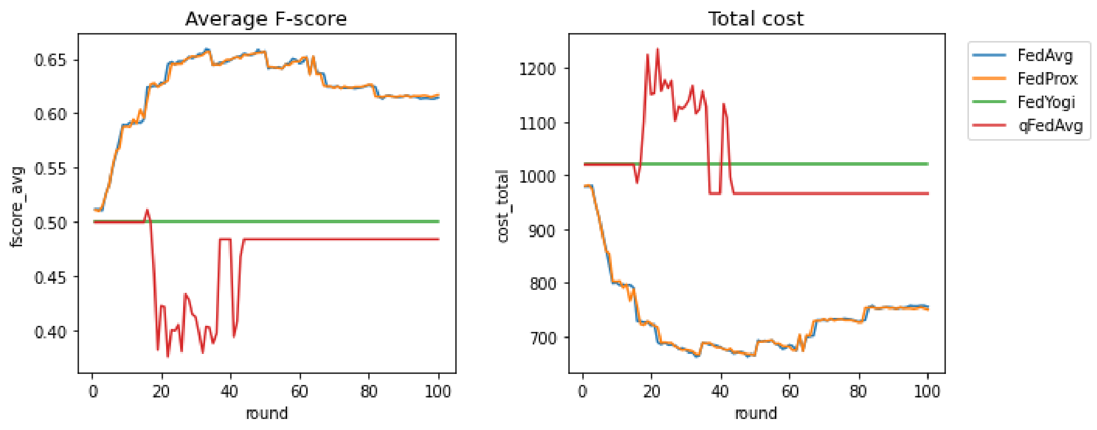


(c) 15 clients

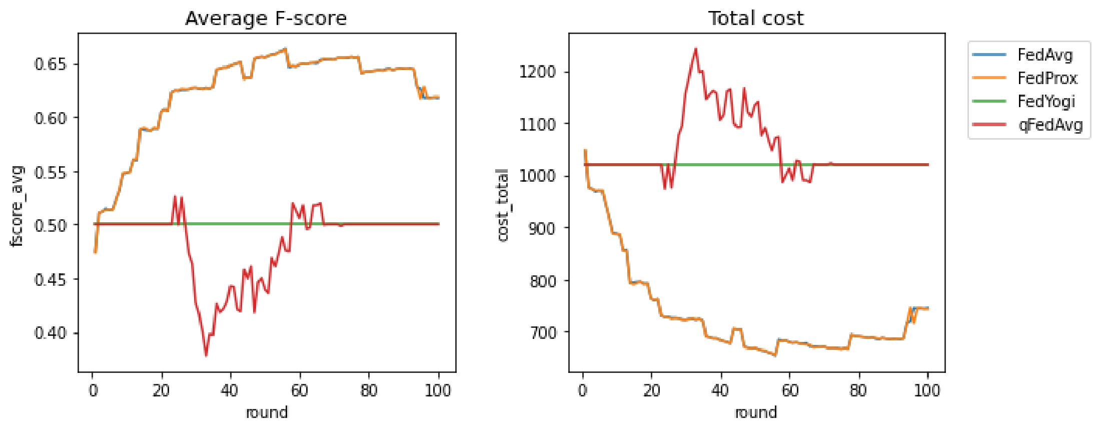
Figure A.2: Training of FL algorithms, Synthetic dataset, i.i.d. case



(a) 5 clients

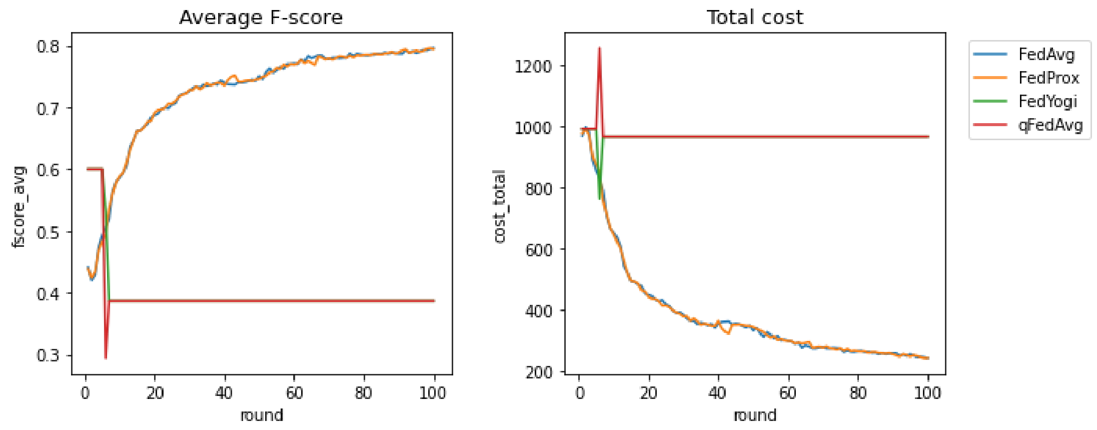


(b) 10 clients

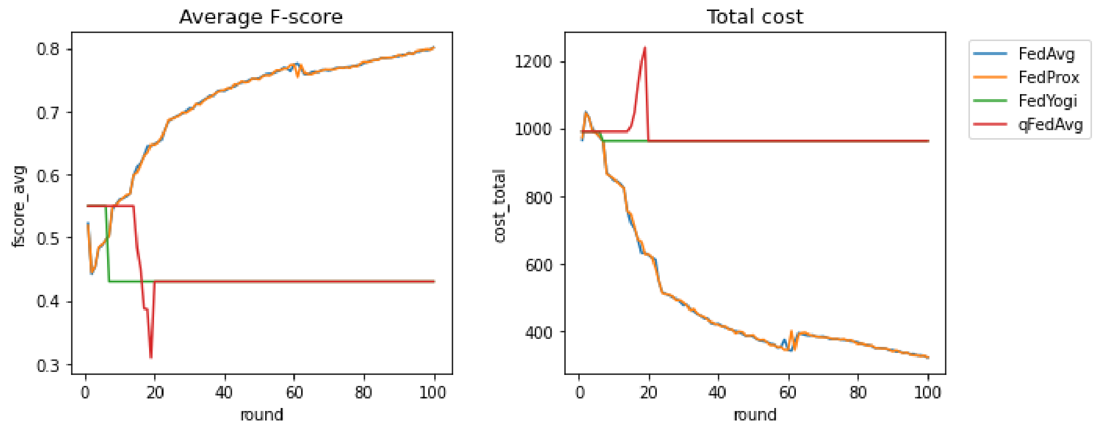


(c) 15 clients

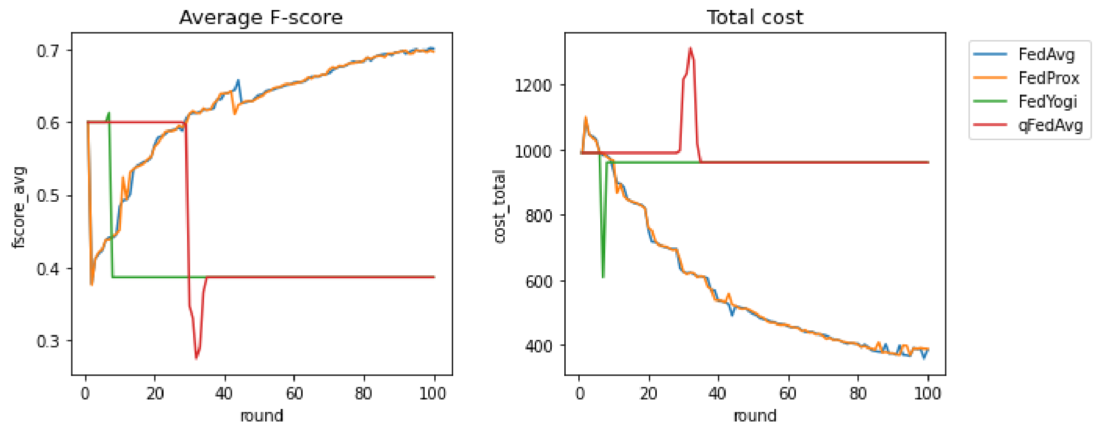
Figure A.3: Training of FL algorithms, Synthetic dataset, feature distribution skew case



(a) 5 clients

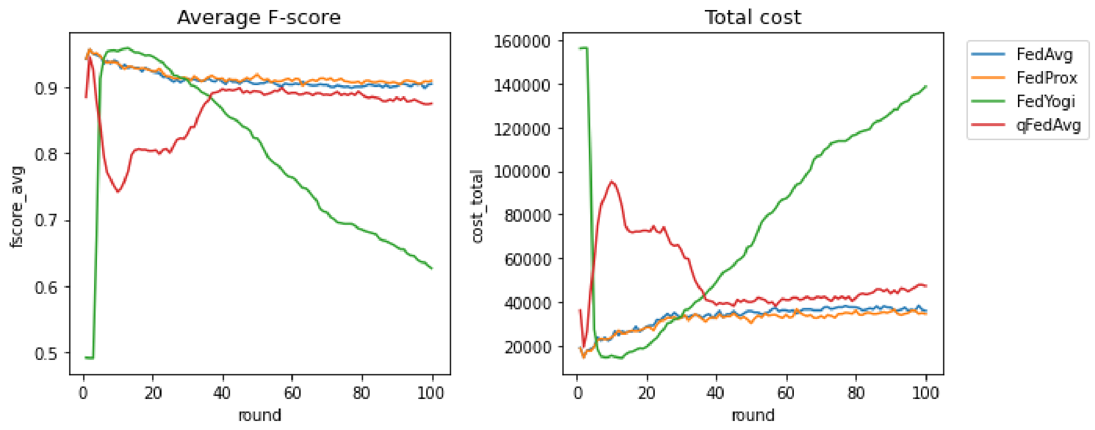


(b) 10 clients

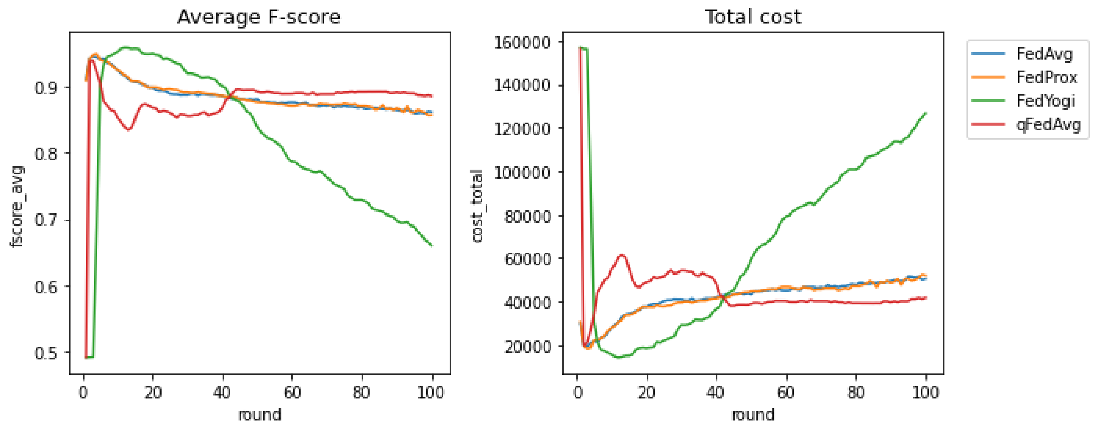


(c) 15 clients

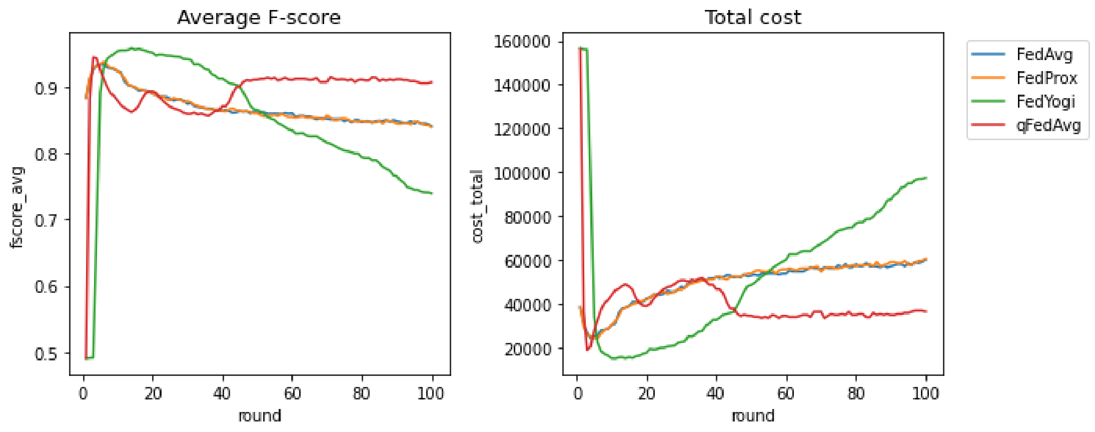
Figure A.4: Training of FL algorithms, Synthetic dataset, quantity skew case



(a) 10 clients

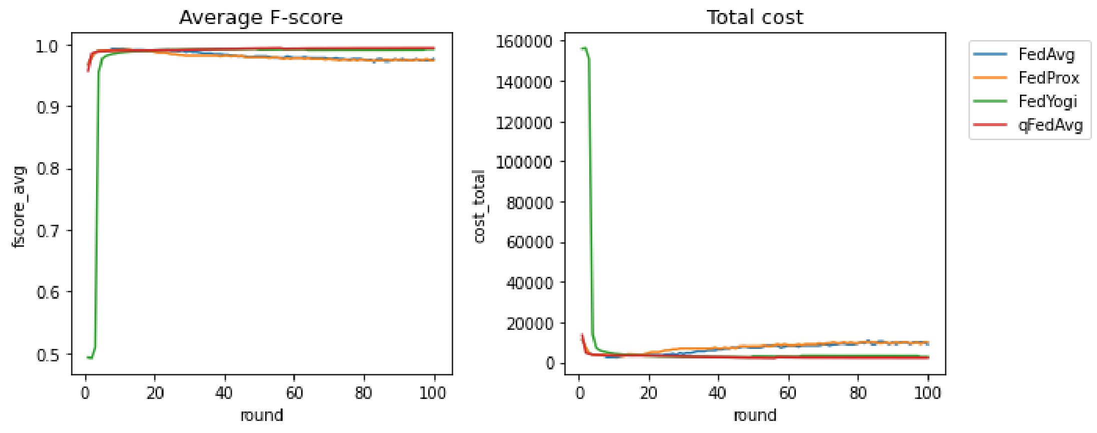


(b) 20 clients

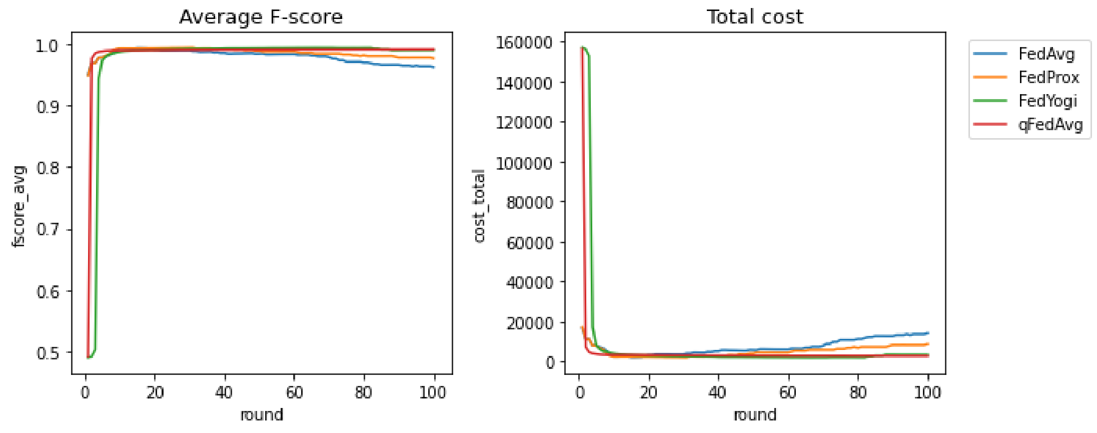


(c) 30 clients

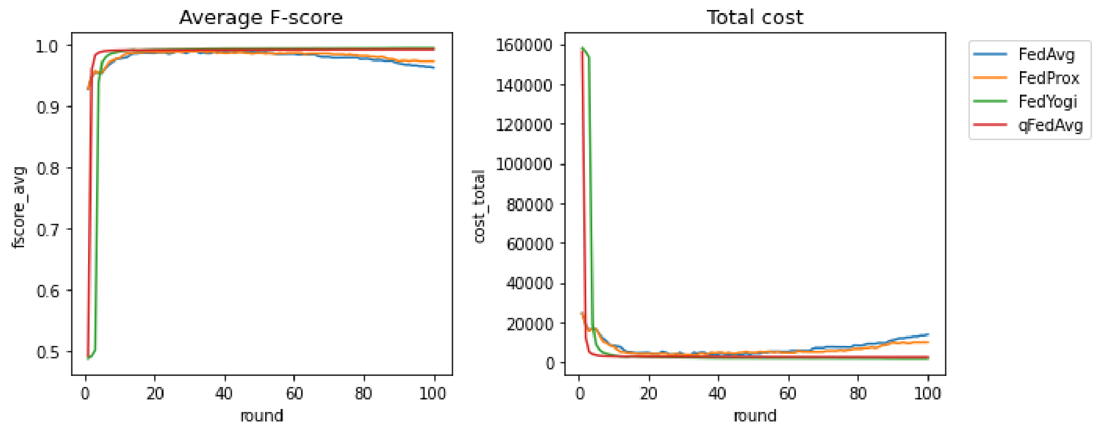
Figure A.5: Training of FL algorithms, Scania trucks, i.i.d. case



(a) 10 clients

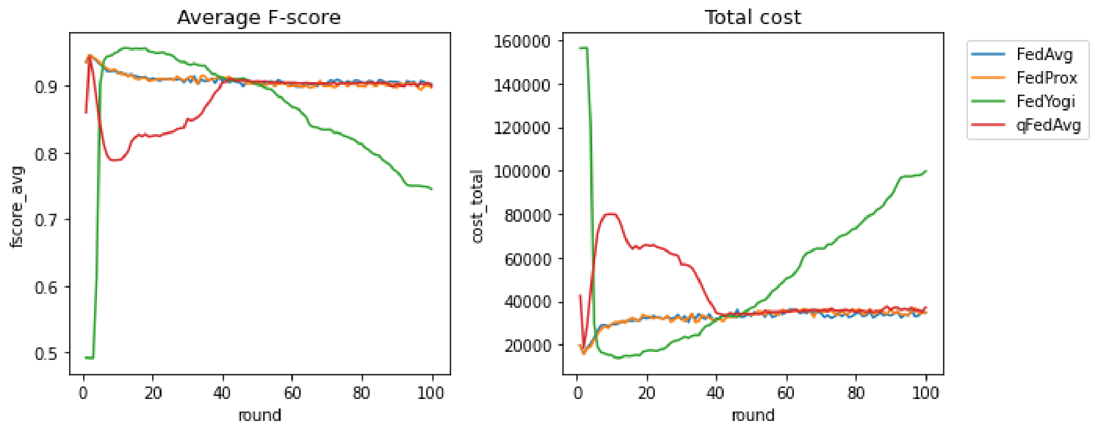


(b) 20 clients

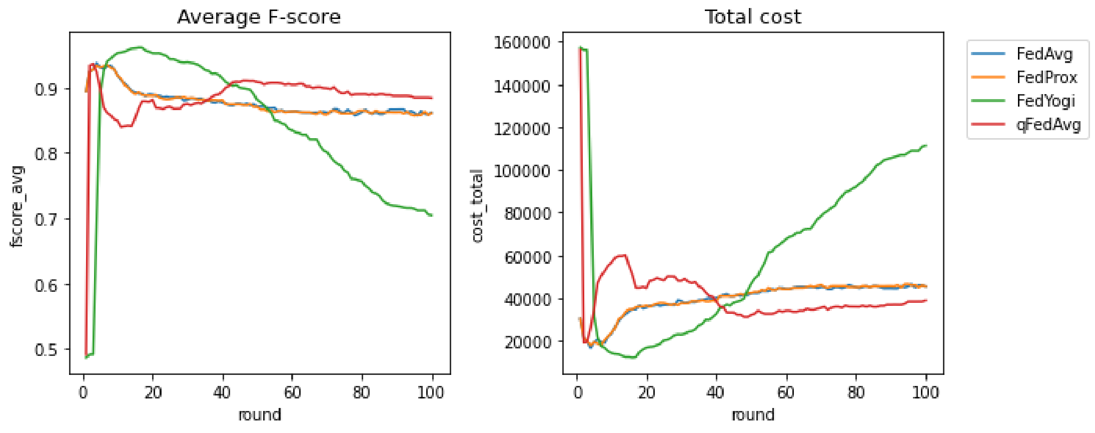


(c) 30 clients

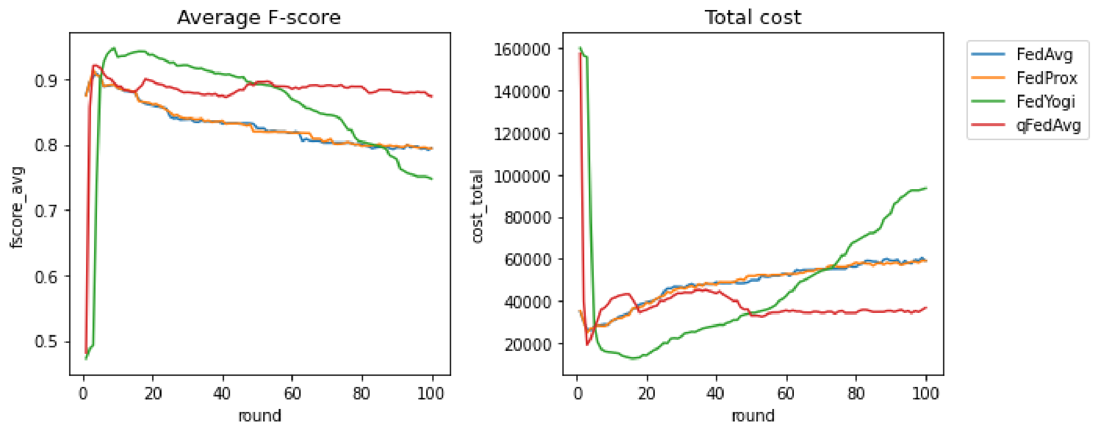
Figure A.6: Training of FL algorithms, Scania trucks, feature distribution skew case



(a) 10 clients

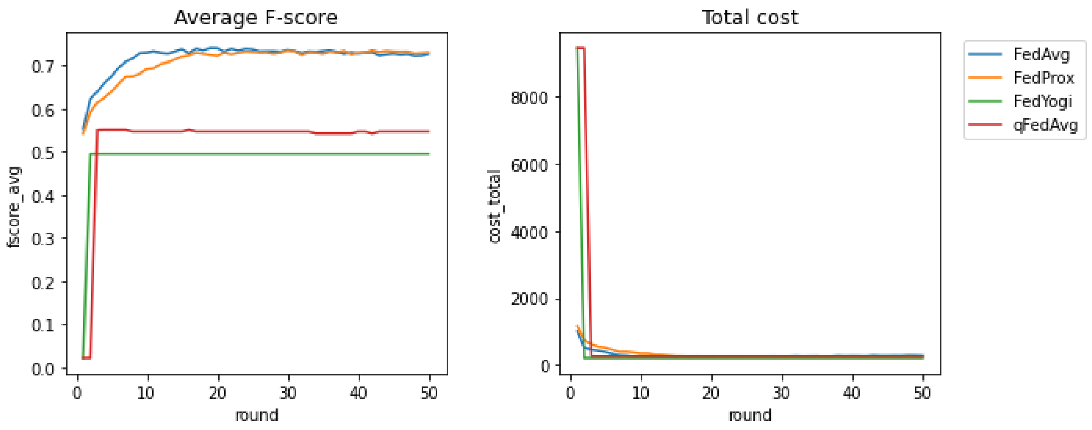


(b) 20 clients

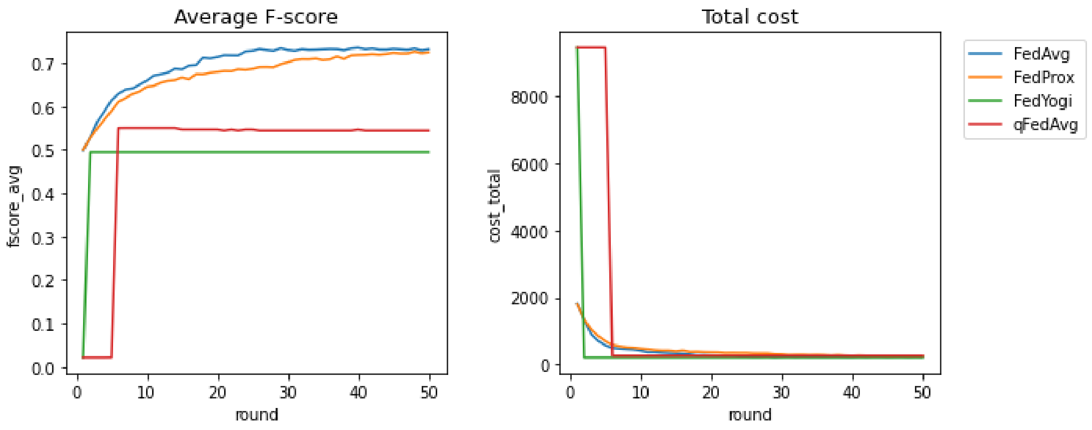


(c) 30 clients

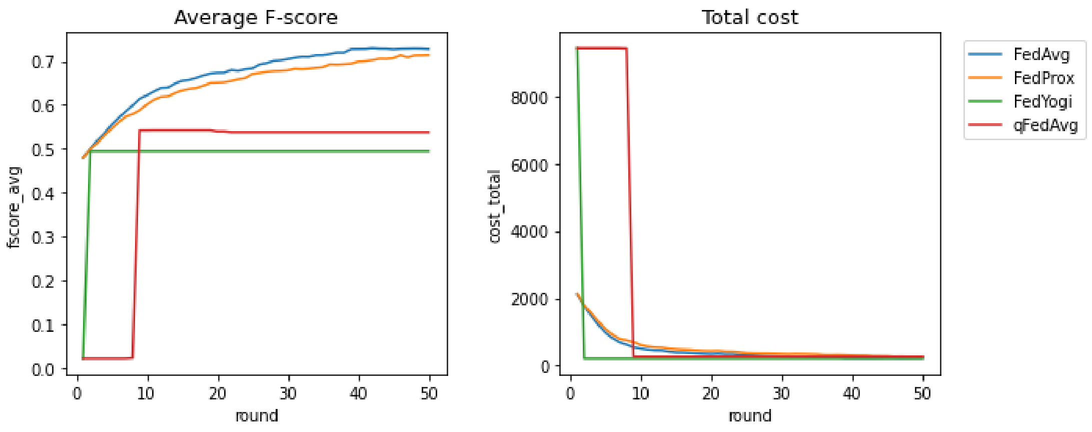
Figure A.7: Training of FL algorithms, Scania trucks, quantity skew case



(a) 3 clients

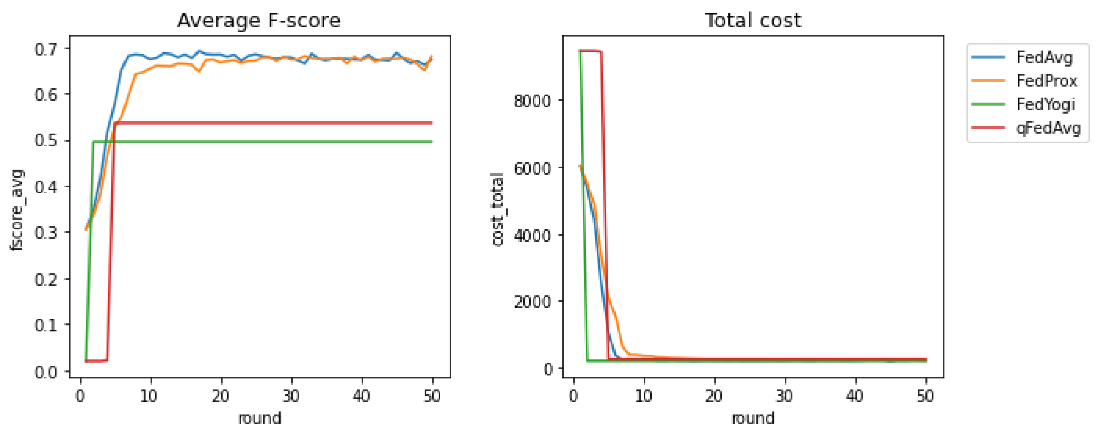


(b) 10 clients

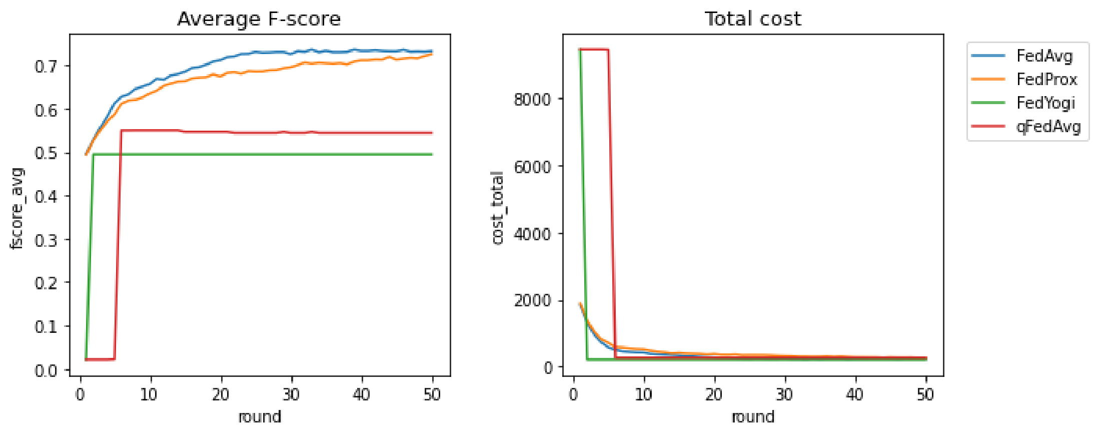


(c) 20 clients

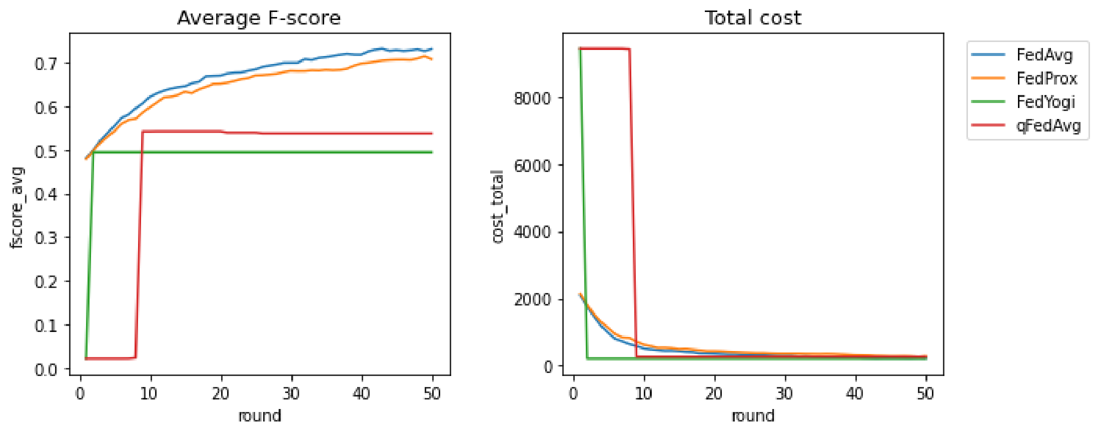
Figure A.8: Training of FL algorithms, Hard drives, i.i.d. case



(a) 3 clients

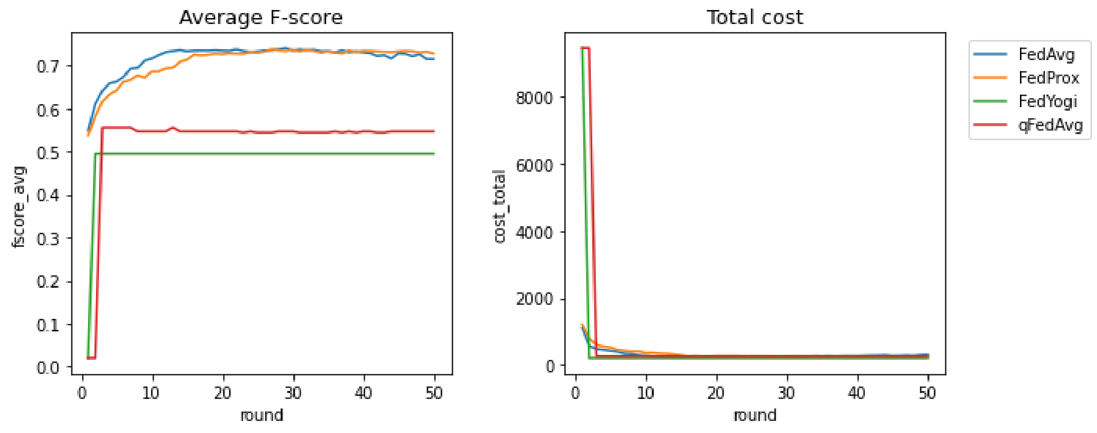


(b) 10 clients

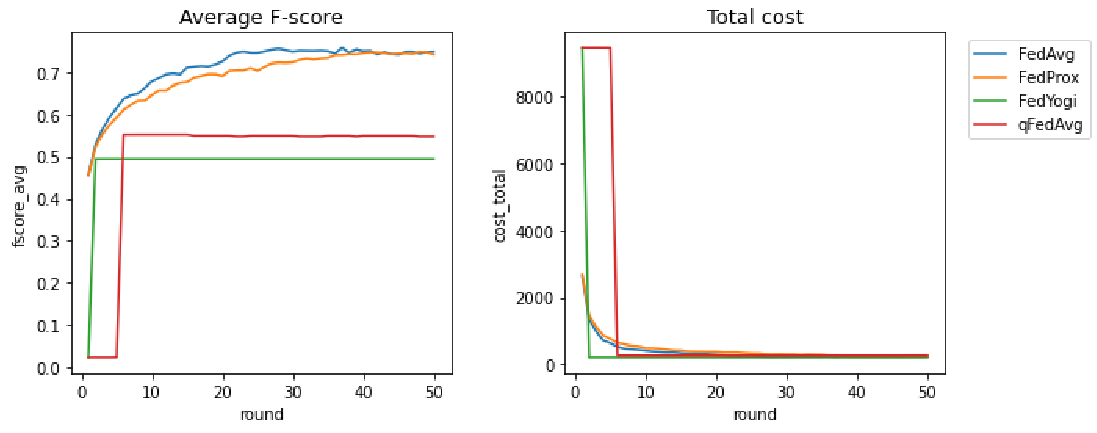


(c) 20 clients

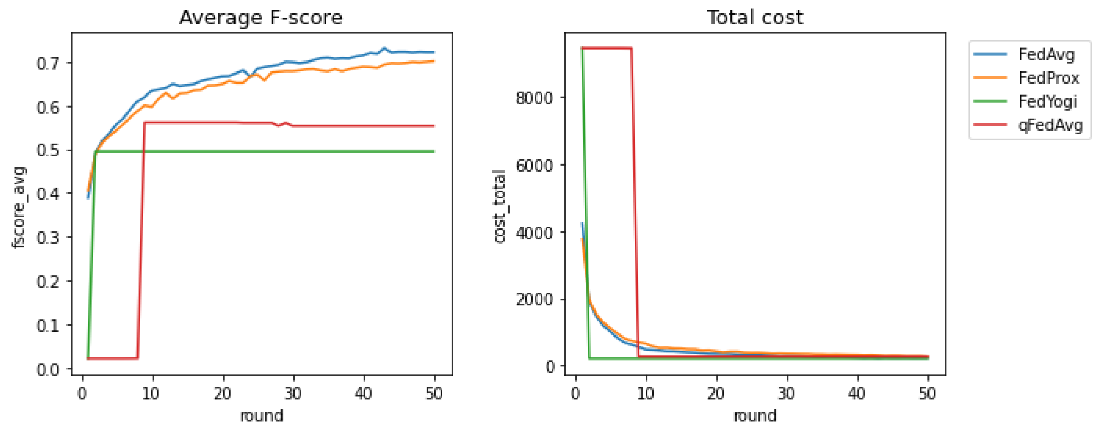
Figure A.9: Training of FL algorithms, Hard drives, feature distribution skew case



(a) 3 clients



(b) 10 clients



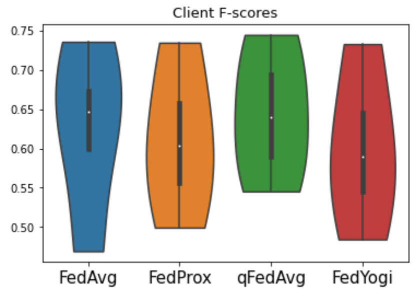
(c) 20 clients

Figure A.10: Training of FL algorithms, Hard drives, quantity skew case

APPENDIX B

Communication costs and fairness of the federated learning algorithms

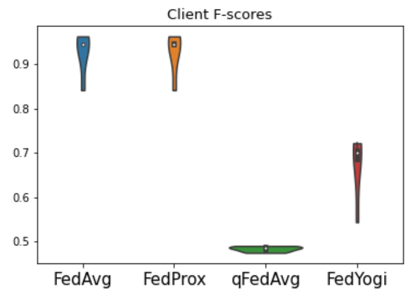
In Figures B.2, B.3, B.4, B.5, B.6, B.6 B.8, B.9, B.10, B.1 we display the distributions of individual clients' test set F-scores at the moment, when each algorithm reaches its best value. We also indicate the communication round and the fairness value.



	FedAvg	FedProx	qFedAvg	FedYogi
F-scores entropy	1.98188	1.98573	1.99015	1.98375
round	94	471	88	167
F-score avg	0.625	0.61	0.643	0.599

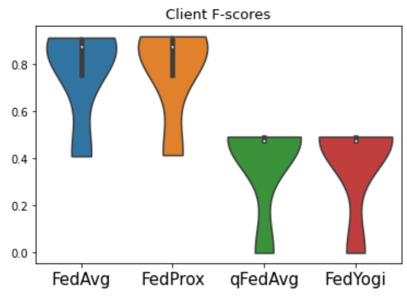
(a) 4 clients

Figure B.1: Fairness and communication cost of FL algorithms, Automotive dataset, feature distribution and quantity skew case



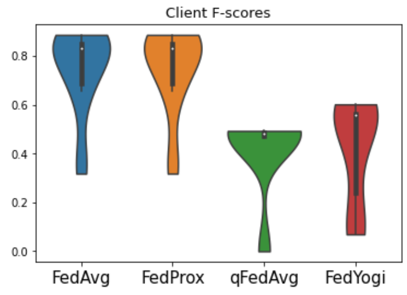
	FedAvg	FedProx	qFedAvg	FedYogi
F-scores entropy	2.3203	2.32029	2.32183	2.31481
round	100	97	7	6
Total cost	159	158	966	609

(a) 5 clients



	FedAvg	FedProx	qFedAvg	FedYogi
F-scores entropy	3.27545	3.27569	2.99979	2.99979
round	97	100	19	7
Total cost	300	296	966	966

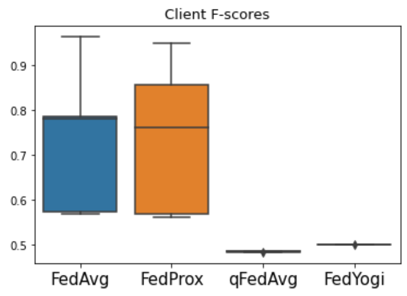
(b) 10 clients



	FedAvg	FedProx	qFedAvg	FedYogi
F-scores entropy	3.86257	3.86257	3.70024	3.68061
round	99	100	26	8
Total cost	399	399	957	910

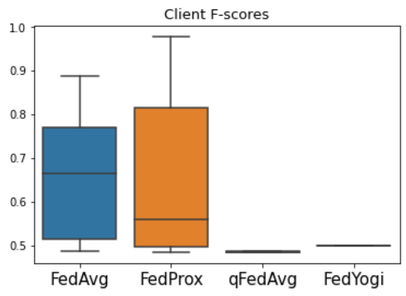
(c) 15 clients

Figure B.2: Fairness and communication cost of FL algorithms, Synthetic dataset, i.i.d. case



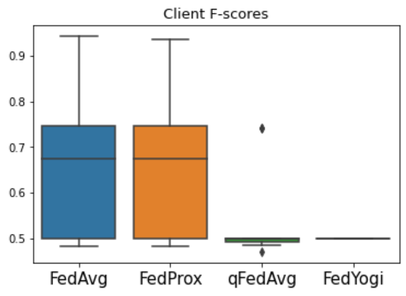
F-scores entropy	2.29248	2.29029	2.32193	2.32193
round	98	81	11	1
Total cost	526	514	966	1020

(a) 5 clients



F-scores entropy	3.28926	3.26634	3.32192	3.32193
round	33	50	37	1
Total cost	662	665	966	1020

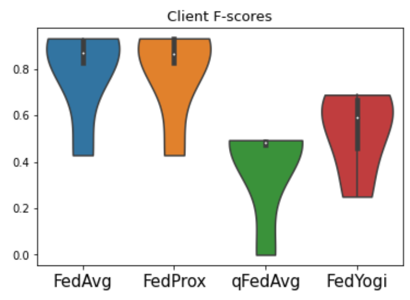
(b) 10 clients



F-scores entropy	3.86572	3.86609	3.88977	3.90689
round	56	56	24	1
Total cost	653	654	974	1020

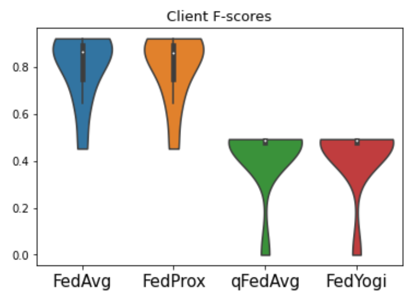
(c) 15 clients

Figure B.3: Fairness and communication cost of FL algorithms, Synthetic dataset, feature distribution skew case



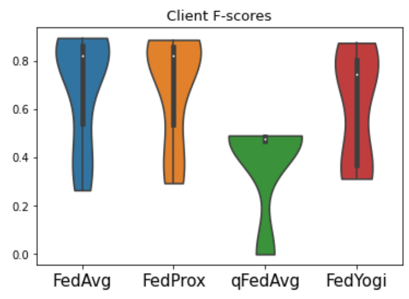
	FedAvg	FedProx	qFedAvg	FedYogi
F-scores entropy	2.27611	2.27597	1.99982	2.24646
round	100	99	7	6
Total cost	241	241	965	761

(a) 5 clients



	FedAvg	FedProx	qFedAvg	FedYogi
F-scores entropy	3.29608	3.29615	3.16809	3.16809
round	100	100	20	7
Total cost	320	321	962	962

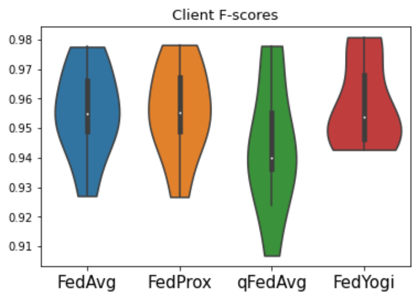
(b) 10 clients



	FedAvg	FedProx	qFedAvg	FedYogi
F-scores entropy	3.82616	3.82882	3.58478	3.80718
round	99	92	35	7
Total cost	358	368	960	607

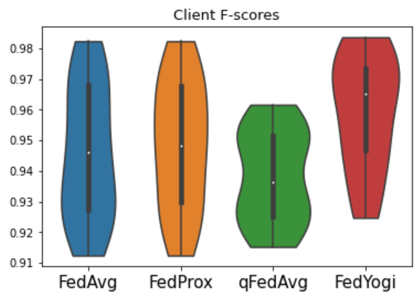
(c) 15 clients

Figure B.4: Fairness and communication cost of FL algorithms, Synthetic dataset, quantity skew case



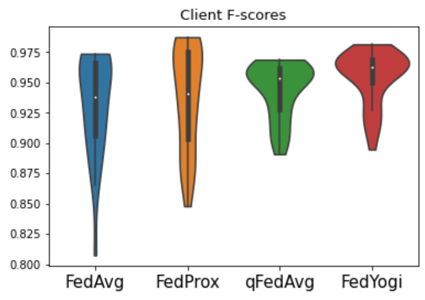
	FedAvg	FedProx	qFedAvg	FedYogi
f-scores entropy	3.32176	3.32175	3.32158	3.32178
round	2	2	2	13
total cost	14660	14510	19420	14300

(a) 10 clients



	FedAvg	FedProx	qFedAvg	FedYogi
f-scores entropy	4.32156	4.32159	4.32175	4.32167
round	3	3	2	12
total cost	19340	18350	19800	14180

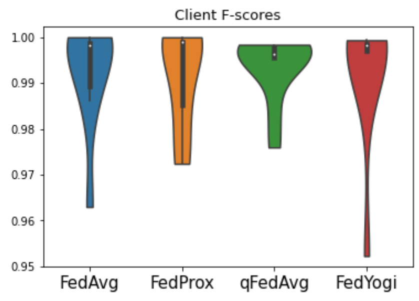
(b) 20 clients



	FedAvg	FedProx	qFedAvg	FedYogi
f-scores entropy	4.90553	4.90545	4.90648	4.90651
round	4	6	3	11
total cost	24140	23940	18650	14840

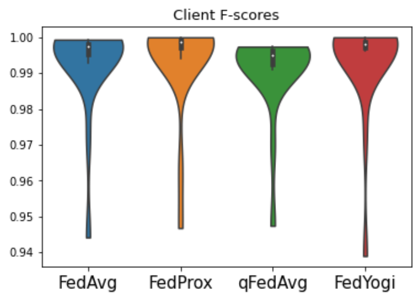
(c) 30 clients

Figure B.5: Fairness and communication cost of FL algorithms, Scania trucks, i.i.d. case



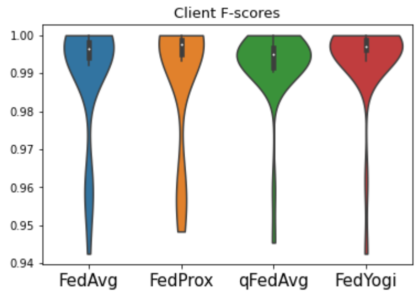
	FedAvg	FedProx	qFedAvg	FedYogi
f-scores entropy	3.32184	3.32185	3.32189	3.32178
round	11	13	56	49
total cost	2450	3060	2060	2420

(a) 10 clients



	FedAvg	FedProx	qFedAvg	FedYogi
f-scores entropy	4.32181	4.3218	4.32183	4.32179
round	15	31	100	66
total cost	2130	2000	2790	1960

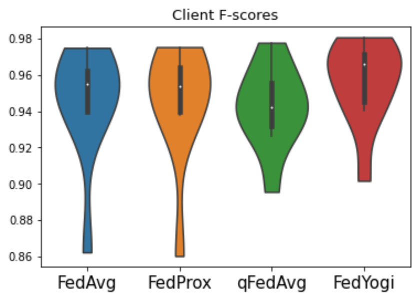
(b) 20 clients



	FedAvg	FedProx	qFedAvg	FedYogi
f-scores entropy	4.90671	4.90668	4.9068	4.90679
round	32	34	87	93
total cost	3480	3680	2570	1920

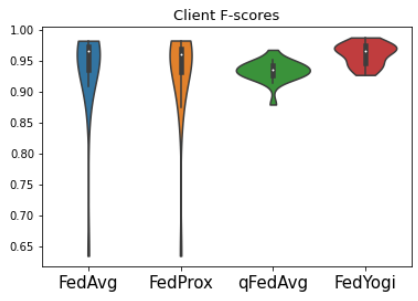
(c) 30 clients

Figure B.6: Fairness and communication cost of FL algorithms, Scania trucks, feature distribution skew case



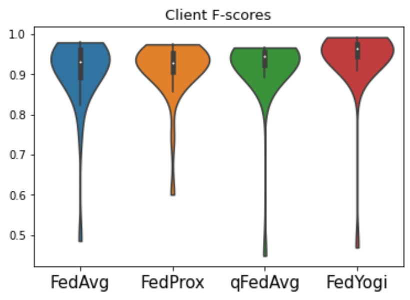
	FedAvg	FedProx	qFedAvg	FedYogi
f-scores entropy	3.32118	3.32113	3.32155	3.32154
round	2	2	2	12
total cost	15670	15650	18610	13850

(a) 10 clients



	FedAvg	FedProx	qFedAvg	FedYogi
f-scores entropy	4.31686	4.31676	4.32168	4.32169
round	4	4	2	16
total cost	16700	17940	19140	12150

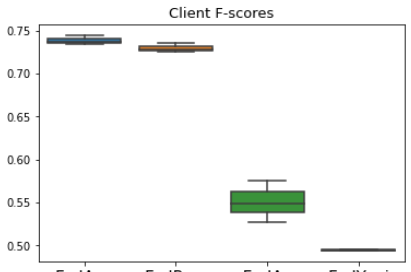
(b) 20 clients



	FedAvg	FedProx	qFedAvg	FedYogi
f-scores entropy	4.89826	4.90178	4.89824	4.89882
round	4	3	3	16
total cost	25970	24790	19030	12640

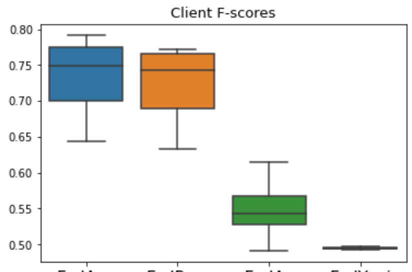
(c) 30 clients

Figure B.7: Fairness and communication cost of FL algorithms, Scania trucks, feature distribution skew case



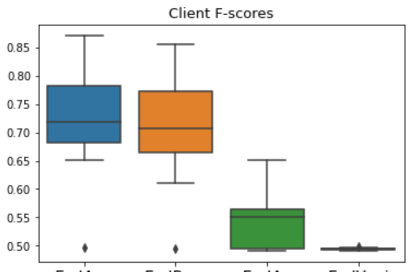
F-scores entropy	1.58494	1.58494	1.58405	1.58496
round	17	29	4	2
Average F-score	0.739	0.73	0.55	0.495

(a) 3 clients



F-scores entropy	3.31872	3.31903	3.3187	3.32192
round	29	48	7	2
Average F-score	0.735	0.727	0.55	0.495

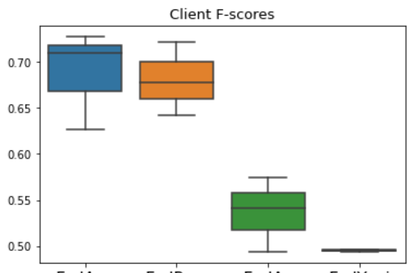
(b) 10 clients



F-scores entropy	4.3118	4.31143	4.31655	4.32192
round	47	50	11	2
Average F-score	0.729	0.714	0.542	0.495

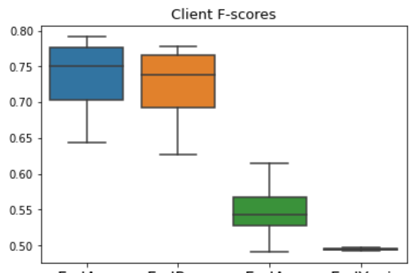
(c) 20 clients

Figure B.8: Fairness and communication cost of FL algorithms, Hard drives, i.i.d. case



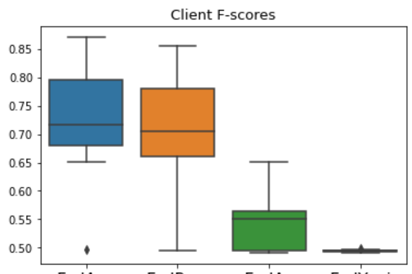
	FedAvg	FedProx	qFedAvg	FedYogi
F-scores entropy	1.58195	1.58331	1.58213	1.58496
round	45	50	6	2
Average F-score	0.688	0.681	0.536	0.495

(a) 3 clients



	FedAvg	FedProx	qFedAvg	FedYogi
F-scores entropy	3.31878	3.31885	3.3187	3.32192
round	33	50	8	2
Average F-score	0.737	0.727	0.55	0.495

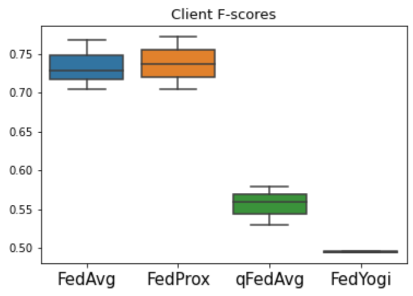
(b) 10 clients



	FedAvg	FedProx	qFedAvg	FedYogi
F-scores entropy	4.31164	4.31124	4.31655	4.32192
round	43	49	11	2
Average F-score	0.732	0.715	0.542	0.495

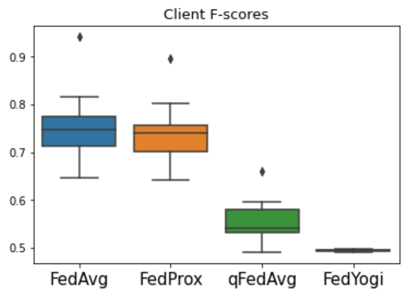
(c) 20 clients

Figure B.9: Fairness and communication cost of FL algorithms, Hard drives, feature distribution skew case



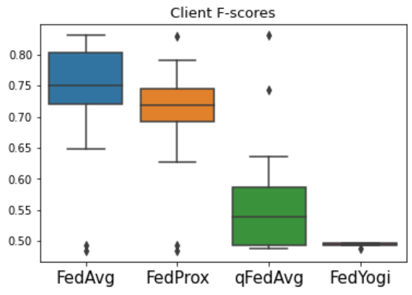
F-scores entropy	1.58406	1.58394	1.58402	1.58496
round	13	27	4	2
Average F-score	0.733	0.738	0.555	0.495

(a) 3 clients



F-scores entropy	3.31494	3.3164	3.31645	3.32192
round	37	47	8	2
Average F-score	0.758	0.744	0.552	0.494

(b) 10 clients



F-scores entropy	4.30847	4.31081	4.30469	4.32191
round	43	50	11	2
Average F-score	0.731	0.701	0.561	0.495

(c) 20 clients

Figure B.10: Fairness and communication cost of FL algorithms, Hard drives, quantity skew case

List of Figures

List of Tables

List of Algorithms

Bibliography

[ADTZ21] Maxime Amram, Jack Dunn, Jeremy J. Toledano, and Ying Daisy Zhuo. Interpretable predictive maintenance for hard drives. *Machine Learning with Applications*, 5:100042, 2021.

[AUDIK$^+$19] Muhammad Ammad-Ud-Din, Elena Ivannikova, Suleiman A Khan, Were Oyomno, Qiang Fu, Kuan Eeik Tan, and Adrian Flanagan. Federated collaborative filtering for privacy-preserving personalized recommendation system. *arXiv preprint arXiv:1901.09888*, 2019.

[AW] Karthik Nama Anil Parth Tamane Vaishnavi Kannan Akarshit Wal, Gnanaguruparan Aishvaryaadevi. Prediction of hard drive failure using s.m.a.r.t statistics. `https://github.com/KarthikNA/Prediction-of-Hard-Drive-Failure`. last accessed: 2022-05-01.

[AWG93] C. Apte, S. Weiss, and G. Grout. Predicting defects in disk drive manufacturing: A case study in high-dimensional classification. In *Proceedings of 9th IEEE Conference on Artificial Intelligence for Applications*, pages 212–218, 1993.

[BDP19] Adrian Binding, Nicholas Dykeman, and Severin Pang. Machine learning predictive maintenance on data in the wild. In *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, pages 507–512. IEEE, 2019.

[BS15] Sailendu Biswal and GR Sabareesh. Design and development of a wind turbine test rig for condition monitoring studies. In *2015 international conference on industrial instrumentation and control (icic)*, pages 891–896. IEEE, 2015.

[BTM$^+$20] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, and Nicholas D Lane. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.

[ÇANZ+20] Zeki Murat Çınar, Abubakar Abdussalam Nuhu, Qasim Zeeshan, Orhan Korhan, Mohammed Asmael, and Babak Safaei. Machine learning in predictive maintenance towards sustainable smart manufacturing in industry 4.0. *Sustainability*, 12(19):8211, 2020.

[CB17] Hongge Chen and Duane Boning. Online and incremental machine learning approaches for ic yield improvement. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 786–793. IEEE, 2017.

[CBHK02] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, Jun 2002.

[CFJ+21] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Dimitrios Papadopoulos, and Qiang Yang. Secureboost: A lossless federated learning framework. *IEEE Intelligent Systems*, 36(6):87–98, 2021.

[CJK04] Nitesh V Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Special issue on learning from imbalanced data sets. *ACM SIGKDD explorations newsletter*, 6(1):1–6, 2004.

[CJL+19] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9268–9277, 2019.

[CLH+21] Junbin Chen, Jipu Li, Ruyi Huang, Ke Yue, Zhuyun Chen, and Weihua Li. Federated learning for bearing fault diagnosis with dynamic weighted averaging. In *2021 International Conference on Sensing, Measurement & Data Analytics in the era of Artificial Intelligence (ICSMD)*, pages 1–6. IEEE, 2021.

[CLL+20] Yu Chen, Fang Luo, Tong Li, Tao Xiang, Zheli Liu, and Jin Li. A training-integrity privacy-preserving federated learning scheme with trusted execution environment. *Information Sciences*, 522:69–79, 2020.

[CMOB19] Mingqing Chen, Rajiv Mathews, Tom Ouyang, and Françoise Beaufays. Federated learning of out-of-vocabulary words. *arXiv preprint arXiv:1903.10635*, 2019.

[CN16] Camila Ferreira Costa and Mario A Nascimento. Ida 2016 industrial challenge: Using machine learning for predicting failures. In *International Symposium on Intelligent Data Analysis*, pages 381–386. Springer, 2016.

[DLFB21] Siddharth Divi, Yi-Shan Lin, Habiba Farrukh, and Z. Berkay Celik. New Metrics to Evaluate the Performance and Fairness of Personalized Federated Learning. *arXiv e-prints arXiv:2107.13173*, 2021.

[DZW$^+$19] Sijing Duan, Deyu Zhang, Yanbo Wang, Lingxiang Li, and Yaoxue Zhang. Jointrec: A deep-learning-based joint cloud video recommendation framework for mobile iot. *IEEE Internet of Things Journal*, 7(3):1655–1666, 2019.

[fat] An industrial grade federated learning framework. `https://fate.fedai.org/`. last accessed: 2022-04-18.

[FMNC19] Amir Falamarzi, Sara Moridpour, Majidreza Nazem, and Samira Cheraghi. Prediction of tram track gauge deviation using artificial neural network and support vector regression. *Australian Journal of Civil Engineering*, 17(1):63–71, 2019.

[FMO20] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. *Advances in Neural Information Processing Systems*, 33:3557–3568, 2020.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[GHP22] Anita Graser, Clemens Heistracher, and Viktorija Pruckovskaja. On the role of spatial data science for federated learning. *UC Santa Barbara: Center for Spatial Studies*, 2022.

[GLZL21] Ning Ge, Guanghao Li, Li Zhang, and Yi Liu. Failure prediction in production line based on federated learning: An empirical study. *Journal of Intelligent Manufacturing*, pages 1–18, 2021.

[GSB$^+$17] Adrià Gascón, Phillipp Schoppmann, Borja Balle, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans. Privacy-preserving distributed linear regression on high-dimensional data. *Proc. Priv. Enhancing Technol.*, 2017(4):345–364, 2017.

[HGLM18] Binxuan Hu, Yujia Gao, Liang Liu, and Huadong Ma. Federated region-learning: An edge computing based framework for urban environment sensing. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7. IEEE, 2018.

[HLL$^+$20a] Meng Hao, Hongwei Li, Xizhao Luo, Guowen Xu, Haomiao Yang, and Sen Liu. Efficient and privacy-enhanced federated learning for industrial artificial intelligence. *IEEE Transactions on Industrial Informatics*, 16(10):6532–6542, 2020.

[HLL+20b] Sixu Hu, Yuan Li, Xu Liu, Qinbin Li, Zhaomin Wu, and Bingsheng He. The oarf benchmark suite: Characterization and implications for federated learning systems. *arXiv preprint arXiv:2006.07856*, 2020.

[HM19] Daniel Frank Hesser and Bernd Markert. Tool wear monitoring of a retrofitted cnc milling machine using artificial neural networks. *Manufacturing letters*, 19:1–4, 2019.

[HMV17] Mark Haarman, Michel Mulders, and Costas Vassiliadis. Predictive maintenance 4.0: predict the unpredictable. *PwC and Mainnovation*, 4, 2017.

[HQB19] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

[HYF+20] Li Huang, Yifeng Yin, Zeng Fu, Shifa Zhang, Hao Deng, and Dianbo Liu. Loadaboost: Loss-based adaboost federated machine learning with reduced computational complexity on iid and non-iid intensive care data. *Plos one*, 15(4):e0230706, 2020.

[JK19] Justin M Johnson and Taghi M Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1):1–54, 2019.

[JPL+19] Shaoxiong Ji, Shirui Pan, Guodong Long, Xue Li, Jing Jiang, and Zi Huang. Learning private neural language modeling with attentive aggregation. In *2019 International joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2019.

[JSP+21] Shaoxiong Ji, Teemu Saravirta, Shirui Pan, Guodong Long, and Anwar Walid. Emerging trends in federated learning: From model fusion to federated x learning. *arXiv preprint arXiv:2102.12920*, 2021.

[KB14] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2014.

[KJK+20] Sai Praneeth Karimireddy, Martin Jaggi, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. Mime: Mimicking centralized stochastic algorithms in federated learning. *arXiv preprint arXiv:2008.03606*, 2020.

[KKM+20] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[KMA+19] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[KMY+16] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtarik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. In *NIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[KYF+20] Ivan Kholod, Evgeny Yanaki, Dmitry Fomichev, Evgeniy Shalugin, Evgenia Novikova, Evgeny Filippov, and Mats Nordlund. Open-source federated learning frameworks for iot: A comparative review and analysis. *Sensors*, 21(1):167, 2020.

[Lee00] Sauchi Stephen Lee. Noisy replication in skewed binary classification. *Computational statistics & data analysis*, 34(2):165–191, 2000.

[LFTL20] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, 149:106854, 2020.

[LGG+17] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[LGN+20] Yi Liu, Sahil Garg, Jiangtian Nie, Yang Zhang, Zehui Xiong, Jiawen Kang, and M Shamim Hossain. Deep anomaly detection for time-series data in industrial iot: A communication-efficient on-device federated learning approach. *IEEE Internet of Things Journal*, 8(8):6348–6358, 2020.

[LGN+21] Yi Liu, Sahil Garg, Jiangtian Nie, Yang Zhang, Zehui Xiong, Jiawen Kang, and M. Shamim Hossain. Deep anomaly detection for time-series data in industrial iot: A communication-efficient on-device federated learning approach. *IEEE Internet of Things Journal*, 8(8):6348–6358, 2021.

[LHW+21] Dun Li, Dezhi Han, Tien-Hsiung Weng, Zibin Zheng, Hongzhi Li, Han Liu, Arcangelo Castiglione, and Kuan-Ching Li. Blockchain for federated learning toward secure distributed machine learning systems: a systemic survey. *Soft Computing*, pages 1–18, 2021.

[LJK+20] Yi Liu, JQ James, Jiawen Kang, Dusit Niyato, and Shuyu Zhang. Privacy-preserving traffic flow prediction: A federated learning approach. *IEEE Internet of Things Journal*, 7(8):7751–7763, 2020.

[LKCS10] Tian Lan, David Kao, Mung Chiang, and Ashutosh Sabharwal. An axiomatic theory of fairness in network resource allocation. In *2010 Proceedings IEEE INFOCOM*, pages 1–9, 2010.

[LKX+20] Yang Liu, Yan Kang, Chaoping Xing, Tianjian Chen, and Qiang Yang. A secure federated transfer learning framework. *IEEE Intelligent Systems*, 35(4):70–82, 2020.

[LLL+20] Yang Liu, Yingting Liu, Zhijie Liu, Yuxuan Liang, Chuishi Meng, Junbo Zhang, and Yu Zheng. Federated forest. *IEEE Transactions on Big Data*, 2020.

[LSBS19] Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. Fair resource allocation in federated learning. *arXiv preprint arXiv:1905.10497*, 2019.

[LSTS20] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[LSZ+18] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

[LWH20] Qinbin Li, Zeyi Wen, and Bingsheng He. Practical federated gradient boosting decision trees. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 4642–4649, 2020.

[LWS+20] Beibei Li, Yuhao Wu, Jiarui Song, Rongxing Lu, Tao Li, and Liang Zhao. Deepfed: Federated deep learning for intrusion detection in industrial cyber–physical systems. *IEEE Transactions on Industrial Informatics*, 17(8):5615–5624, 2020.

[LWW+21] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. A survey on federated learning systems: vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[Mat20] Stephan Matzka. Explainable artificial intelligence for predictive maintenance applications. In *2020 Third International Conference on Artificial Intelligence for Industries (AI4I)*, pages 69–74, 2020.

[MMC$^+$12] Jonathan Masci, Ueli Meier, Dan Ciresan, Jürgen Schmidhuber, and Gabriel Fricout. Steel defect classification with max-pooling convolutional neural networks. In *The 2012 international joint conference on neural networks (IJCNN)*, pages 1–6. IEEE, 2012.

[MMR$^+$17] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[Mob02] R Keith Mobley. *An introduction to predictive maintenance*. Elsevier, 2002.

[MRTZ18] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *International Conference on Learning Representations*, 2018.

[MTDC19] Nishat I Mowla, Nguyen H Tran, Inshil Doh, and Kijoon Chae. Federated learning-based cognitive detection of jamming attack in flying ad-hoc network. *IEEE Access*, 8:4338–4350, 2019.

[NCP$^+$21] Thong Phi Nguyen, Seungho Choi, Sung-Jun Park, Sung Hyuk Park, and Jonghun Yoon. Inspecting method for defective casting products with convolutional neural network (cnn). *International Journal of Precision Engineering and Manufacturing-Green Technology*, 8(2):583–594, 2021.

[NWI$^+$13] Valeria Nikolaenko, Udi Weinsberg, Stratis Ioannidis, Marc Joye, Dan Boneh, and Nina Taft. Privacy-preserving ridge regression on hundreds of millions of records. In *2013 IEEE symposium on security and privacy*, pages 334–348. IEEE, 2013.

[pad] Baidu paddlepaddle releases 21 new capabilities to accelerate industry-grade model development. `http://research.baidu.com/Blog/index-view?id=12`. last accessed: 2022-04-18.

[PDH19] Stephen R Pfohl, Andrew M Dai, and Katherine Heller. Federated and differentially private learning for electronic health records. *arXiv preprint arXiv:1911.05861*, 2019.

[pys] Pysyft: A library for answering questions using data you cannot see. `https://blog.openmined.org/tag/pysyft/`. last accessed: 2022-04-18.

[QMM$^+$18] Juan C Quiroz, Norman Mariun, Mohammad Rezazadeh Mehrjou, Mahdi Izadi, Norhisam Misron, and Mohd Amran Mohd Radzi. Fault detection of broken rotor bar in ls-pmsm using random forests. *Measurement*, 116:273–280, 2018.

[RCZ+21] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H. Brendan McMahan. Adaptive federated optimization, 2021.

[RLD+18] Hao Ren, Hongwei Li, Yuanshun Dai, Kan Yang, and Xiaodong Lin. Querying in internet of things with privacy preserving: Challenges, solutions and opportunities. *IEEE Network*, 32(6):144–151, 2018.

[Ros58] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[RTID21] Rahul Rai, Manoj Kumar Tiwari, Dmitry Ivanov, and Alexandre Dolgui. Machine learning in manufacturing and industry 4.0 applications, 2021.

[SER+20] Micah J Sheller, Brandon Edwards, G Anthony Reina, Jason Martin, Sarthak Pati, Aikaterini Kotrotsou, Mikhail Milchenko, Weilin Xu, Daniel Marcus, Rivka R Colen, et al. Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. *Scientific reports*, 10(1):1–12, 2020.

[SGR+19] Santiago Silva, Boris A Gutman, Eduardo Romero, Paul M Thompson, Andre Altmann, and Marco Lorenzi. Federated learning in distributed medical databases: Meta-analysis of large-scale subcortical brain data. In *2019 IEEE 16th international symposium on biomedical imaging (ISBI 2019)*, pages 270–274. IEEE, 2019.

[SH18] Chuan-Jun Su and Shi-Feng Huang. Real-time big data analytics for hard disk drive predictive maintenance. *Computers & Electrical Engineering*, 71:93–101, 2018.

[SHN+19] Yuris Mulya Saputra, Dinh Thai Hoang, Diep N. Nguyen, Eryk Dutkiewicz, Markus Dominik Mueck, and Srikathyayani Srikanteswara. Energy demand prediction with federated learning for electric vehicle networks, 2019.

[SLZ+20] Jiahao Shi, Zhenye Li, Tingting Zhu, Dongyi Wang, and Chao Ni. Defect detection of industry wood veneer based on nas and multichannel mask r-cnn. *Sensors*, 20(16):4398, 2020.

[SMWB18] Stephan Spiegel, Fabian Müller, Dorothea Wiesmann, and John Bird. Cost-sensitive learning for predictive maintenance. *ArXiv*, abs/1809.10979, 2018.

[SSVFSdSAdS19] Gustavo Scalabrini Sampaio, Arnaldo Rabello de Aguiar Vallim Filho, Leilton Santos da Silva, and Leandro Augusto da Silva. Prediction of motor failure time using an artificial neural network. *Sensors*, 19(19):4342, 2019.

[Ste] Doug Steen. Beyond the f-1 score: A look at the f-beta score. `https://medium.com/@douglaspsteen/beyond-the-f-1-score-a-look-at-the-f-beta-score-3743ac2ef6e3`. last accessed: 2022-02-01.

[SVG18] Konstantin Sozinov, Vladimir Vlassov, and Sarunas Girdzijauskas. Human activity recognition using federated learning. In *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, pages 1103–1111. IEEE, 2018.

[TDTN20] Canh T Dinh, Nguyen Tran, and Josh Nguyen. Personalized federated learning with moreau envelopes. *Advances in Neural Information Processing Systems*, 33:21394–21405, 2020.

[ten] Tensorflow federated: machine learning on decentralized data. `https://www.tensorflow.org/federated`. last accessed: 2022-04-18.

[VHKN07] Jason Van Hulse, Taghi M Khoshgoftaar, and Amri Napolitano. Experimental perspectives on learning from imbalanced data. In *Proceedings of the 24th international conference on Machine learning*, pages 935–942, 2007.

[VSP$^+$17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[VWJ$^+$19] Dinesh C Verma, Graham White, Simon Julier, Stepehen Pasteris, Supriyo Chakraborty, and Greg Cirincione. Approaches to address the data skew problem in federated learning. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, page 110061I. International Society for Optics and Photonics, 2019.

[VYJ08] Jaideep Vaidya, Hwanjo Yu, and Xiaoqian Jiang. Privacy-preserving svm classification. *Knowledge and Information Systems*, 14(2):161–178, 2008.

[Wik] Wikipedia. S.M.A.R.T. — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=S.M.A.R.T.&oldid=1078605550`. last accessed: 2022-02-01.

[WLL+20] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020.

[WLW20] Xing Wu, Zhaowang Liang, and Jianjia Wang. Fedmed: A federated learning framework for language modeling. *Sensors*, 20(14):4048, 2020.

[WSRS16] Daniel Weimer, Bernd Scholz-Reiter, and Moshe Shpitalni. Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection. *CIRP Annals*, 65(1):417–420, 2016.

[WY07] Achmad Widodo and Bo-Suk Yang. Support vector machine in machine condition monitoring and fault diagnosis. *Mechanical systems and signal processing*, 21(6):2560–2574, 2007.

[WYS+20] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020.

[XHL18] Shili Xiang, Dong Huang, and Xiaoli Li. A generalized predictive framework for data driven prognostics and diagnostics using machine logs. In *TENCON 2018-2018 IEEE Region 10 Conference*, pages 0695–0700. IEEE, 2018.

[XLD+19] Guowen Xu, Hongwei Li, Yuanshun Dai, Kan Yang, and Xiaodong Lin. Enabling efficient and geometric range query with access control over encrypted spatial data. *IEEE Transactions on Information Forensics and Security*, 14(4):870–885, 2019.

[YAG+19] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *International Conference on Machine Learning*, pages 7252–7261. PMLR, 2019.

[YZY+19] Wensi Yang, Yuhang Zhang, Kejiang Ye, Li Li, and Cheng-Zhong Xu. Ffd: A federated learning based method for credit card fraud detection. In *International conference on big data*, pages 18–32. Springer, 2019.

[ZdCdRR+20] Tiago Zonta, Cristiano André da Costa, Rodrigo da Rosa Righi, Miromar Jose de Lima, Eduardo Silveira da Trindade, and Guann Pyng Li. Predictive maintenance in the industry 4.0: A systematic literature review. *Computers & Industrial Engineering*, 150:106889, 2020.

[ZGC+21] Zehui Zhang, Cong Guan, Hui Chen, Xiangguo Yang, Wenfeng Gong, and Ansheng Yang. Adaptive privacy preserving federated learning for fault diagnosis in internet of ships. *IEEE Internet of Things Journal*, 2021.

[ZHA19] Jan Zenisek, Florian Holzinger, and Michael Affenzeller. Machine learning based concept drift detection for predictive maintenance. *Computers Industrial Engineering*, 137:106031, 2019.

[ZLM+21] Wei Zhang, Xiang Li, Hui Ma, Zhong Luo, and Xu Li. Federated learning for machinery fault diagnosis with dynamic validation and self-supervision. *Knowledge-Based Systems*, 213:106679, 2021.

[ZLY+20] Weishan Zhang, Qinghua Lu, Qiuyu Yu, Zhaotong Li, Yue Liu, Sin Kit Lo, Shiping Chen, Xiwei Xu, and Liming Zhu. Blockchain-based federated learning for device failure detection in industrial iot. *IEEE Internet of Things Journal*, 8(7):5926–5937, 2020.

[ZWL+13] Bingpeng Zhu, Gang Wang, Xiaoguang Liu, Dianming Hu, Sheng Lin, and Jingwei Ma. Proactive drive failure prediction for large scale storage systems. In *2013 IEEE 29th symposium on mass storage systems and technologies (MSST)*, pages 1–5. IEEE, 2013.

[ZWZ+21] Jinglin Zhang, Yanbo Wang, Kai Zhu, Yi Zhang, and Yuanjiang Li. Diagnosis of interturn short-circuit faults in permanent magnet synchronous motors based on few-shot learning under a federated learning framework. *IEEE Transactions on Industrial Informatics*, 17(12):8495–8504, 2021.

[ZXB+21] Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. A survey on federated learning. *Knowledge-Based Systems*, 216:106775, 2021.

[ZYW19] Weiting Zhang, Dong Yang, and Hongchao Wang. Data-driven methods for predictive maintenance of industrial equipment: A survey. *IEEE Systems Journal*, 13(3):2213–2227, 2019.

[ZZL+21] Jiehan Zhou, Shouhua Zhang, Qinghua Lu, Wenbin Dai, Min Chen, Xin Liu, Susanna Pirttikangas, Yang Shi, Weishan Zhang, and Enrique Herrera-Viedma. A survey on federated learning and its applications for accelerating industrial internet of things. *arXiv preprint arXiv:2104.10501*, 2021.