CHAPTER 4

Federated learning for enabling cooperation between the Royal Netherlands Navy and external parties in developing predictive maintenance

Anna C. Vriend, Wieger W. Tiddens & Relinde P.M.J. Jurrius

Abstract

Nowadays, Machine Learning is used successfully in many applications. Two important challenges are involved with its implementation: i) in many industries, data is stored in separate places; ii) the growing demand for Machine Learning that respects the privacy of the training data. Conventional Machine Learning is based on centralised data collection and is thus unable to face these challenges. Federated Learning is a solution to this problem. This work focuses on how the Royal Netherlands Navy can use Federated Learning to train a Long Short Term Memory Machine Learning model with data sets from an external party, Royal Van Oord, without sharing raw data. The Federated Learning approach exceeds the accuracy of central Machine Learning and can be used on the Ministry of Defence's intranet.

Keywords: Machine learning, Federated learning, Predictive maintenance, Government-industry collaboration

4.1. Introduction

Data from military platforms (such as sensor and failure data) is increasingly being used to develop predictive maintenance algorithms. These learning algorithms offer an opportunity to identify failures at an earlier stage, better plan maintenance and reduce the (corrective) workload aboard ships with the help of data analysis and Machine Learning (ML). ML is the use of algorithms to analyse data, learn from it and then give advice or predictions to a user. This large amount of data offers many possibilities, but data is often privacy-sensitive or not accessible. As a result, the demand for privacy-friendly ways to use data for ML is growing.

One potential way to solve the problems outlined is Federated Learning (FL). This new ML technique ensures that the owner of the data remains in control of their own data and that privacy-sensitive data does not have to be shared with external parties[1]. The term FL was coined by Google in 2017 and has had an increasing number of uses since then. Google itself uses it, for example, to improve suggestions for the Gboard keyboard and for the Health Studies app[2,3]. Users do not share their raw data, but Google does learn from this data through FL. FL can also be used for Natural Language Processing (NLP) and Predictive Maintenance, among other things.

A constant pressure on reducing the Royal Netherlands Navy's ships' crews and an increasing complexity of systems aboard naval ships creates pressure on the maintenance of future naval ships. The RNLN therefore works on a transition from planned periodic maintenance with a high corrective workload towards predictive maintenance based on advanced condition monitoring and data analysis techniques. Predictive maintenance is a preventive maintenance approach that uses analytics to inform the RNLN as owner-operator about the current, and preferably also the future state of their physical assets[4,5]. The RNLN uses ML models to predict the maintenance needs of its fleet based on (sensor) data of (machinery aboard) their ships. As an example, for a diesel engine this data may include measurements like temperatures of the bearings, fuel flow, engine speeds and power.

To facilitate this development, the RNLN needs to collaborate with knowledge institutions, industry and other external parties. When several parties work together with similar tools, there is a greater amount of data available and this potentially results in more accurate predictions on for example the required maintenance of diesel engines or pumps aboard ships. These collaborations can be made possible through the use of FL, so that the RNLN does not have to share raw data with external parties, as this data is oftentimes classified. This paper focuses on a specific case study of sharing an ML model with Royal Van Oord. Van Oord is one of the largest dredging companies in the world and has a lot of sensor data available from their ships.

In this chapter an FL architecture is set up, tested, and compared with central ML on aspects such as the model accuracy and the required data traffic. A Long Short Term Memory (LSTM) neural network (NN) from the RNLN[6] is used to test central ML and FL. Data sets from one of Van Oord's vessels, the Flexible Fall Pipe Vessel (FFPV) Stornes, were made available as training data. Experiments to train the NN federated were carried out with several laptops that served as clients and the central server. The chosen NN evaluation parameters were used to compare the results of FL and central ML. The selected LSTM network, a Recurrent Neural Network (RNN), is able to detect anomalies in ship sensor data. This model is specifically developed within the RNLN to detect anomalies in diesel engines. To train

this model, various sensors can serve as input; a specific data set is not required. The outputs of this LSTM result in a warning table, Figure 4.1 shows an example. This warning table helps maintenance engineers in their day-to-day maintenance decision making (i.e., deciding whether to inspect, repair or replace an item or to continue operation).

This paper is structured as follows. Section 2 gives a description of FL. Section 3 discusses the designed system architecture of FL with one of the more flexible FL frameworks suitable for using time series data, the Flower framework. In Section 4, the reliability and validity of the study is discussed. Next, Section 5 compares our FL architecture with central ML. Finally, conclusions, limitations and general reflections will be given in Section 6.

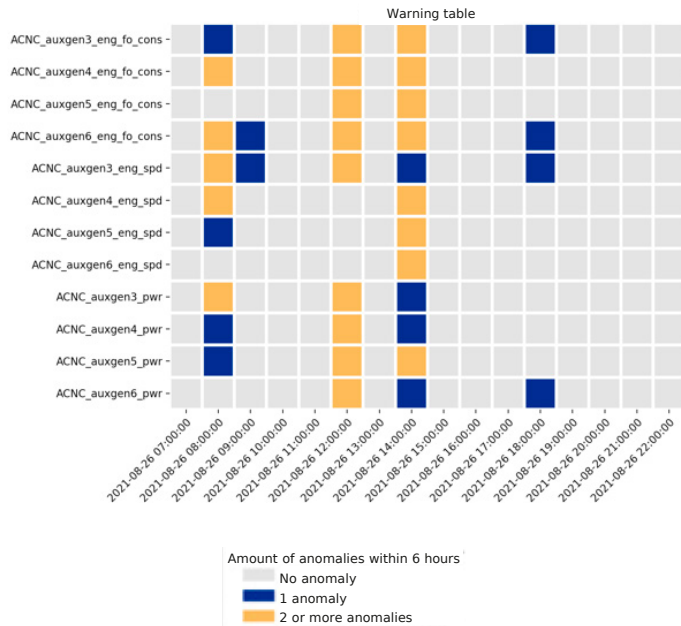

Fig. 4.1. An example of a warning table generated by the LSTM model shows the number of anomalies generated within blocks of six hours of generators 3, 4, 5 and 6 of FFPV Stornes.

4.2. Federated learning

The idea of Machine Learning (ML) is to allow systems to make data-driven decisions instead of programming the systems explicitly to execute a certain task. An early example of ML is a spam filter for email. Instead of trying to describe to the

spam filter what makes a message spam, the system is provided with many examples of spam and non-spam emails and learns the difference itself. This training is done via an ML algorithm, where the result of the training is an ML model.

Nowadays, Machine Learning is used successfully in many applications. There are two important challenges involved with the implementation of this technique. The first is that in many industries, data is stored in separate places. The second challenge is the growing demand for ML that respects the privacy of the training data. Conventional ML is based on centralised data collection and is thus unable to face these challenges[7].

A solution to this increasing fragmentation and isolation of data is found in FL. This is an ML approach that makes it possible to train models with decentralised data. The idea of FL is that the algorithm is brought to the data instead of the data to the algorithm[8]. This is visualised in Figure 4.2. FL is fundamentally different in this respect from, for example, the ML technique of distributed learning, in which the data is transferred to the algorithm.

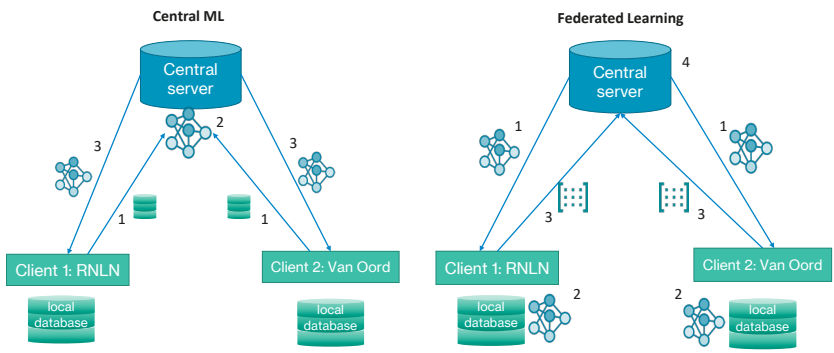

Fig. 4.2. On the left Central ML where in 1 data sets are sent from the client to the server, in 2 the server trains the model and in 3 the trained model is sent back to the clients. On the right FL where in 1 the ML model is initialized, in 2 the model is trained locally, in 3 the clients model gradients are sent to the server and in 4 the sub models are aggregated.

In FL, sub-models are trained at different parties, called *clients*, using only local data. A client can be a company, a ship, but also a mobile device. Client raw data is stored locally and not shared to ensure user privacy and data confidentiality. The various parties then share their sub-models with a central server, who *aggregates* the sub-models into a global model. The process must be carefully designed so that no party can trace the confidential data of other parties[9,10,11].

4.3. Characteristics of federated learning

FL is a relatively new concept and as a result there are different interpretations of the term in existing literature. This section examines five characteristics of FL as adopted in this study to avoid misinterpretation of terms and concepts. The characteristics are based on various definitions of FL[12,13].

4.3.1. Central orchestration

In setting up the training process, the central server has to take a number of aspects into account. The design of the server is driven by the number of clients, the size of model updates and the amount of data traffic. The number of clients can vary from tens to hundreds or millions, depending on the application of the model. The updates that are collected and communicated can also vary in size from kilobytes to tens of megabytes. Finally, the amount of data traffic from certain clients can vary greatly throughout the day.

4.3.2. Local data processing

With FL, data is generated locally at clients and remains decentralised. Raw data is not shared with other clients or with the central server. The data generated by clients is often not independently identically distributed (non-i.i.d.), hence the model may be *biased*. There are several solutions to counteract a bias in the model, for example local reweighting[14].

4.3.3. Communication

Communication is a bottleneck in *federated* networks. The communication links between clients and the server can be slow and unstable and clients can be physically distributed across the earth. It is therefore important to develop efficient communication methods that frequently send small messages or model updates for the training of the model.

Local updating methods, such as *federated averaging* (FedAvg), can reduce the number of communication rounds required for training. Furthermore, using model compression schemes significantly reduces the size of communication messages each round. Decentralised training also makes communication more efficient. It is faster than centralised training on networks with low bandwidths or high data transfer delays.

4.3.4. Privacy and confidentiality

Privacy and confidentiality are often very important for parties that use FL. FL protects local data by sharing model updates, instead of the raw data. Although this is a step towards data protection, it may still be possible for a third party or the central server to recover raw data from the model updates during the training process.

Methods that aim to improve privacy within FL, such as *Secure Multiparty Computation* (SMC) or *Differential Privacy* (DP), come at the cost of model performance or system efficiency. A balance must be struck between these two considerations when realising FL systems.

4.3.5. System heterogeneity

The communication, computing and storage capabilities of each client in a federated network often differ. This is due to a variation in hardware, network connections (3G, 4G, 5G or Wi-Fi) and computing power. In addition, the network size and system-related limitations of the clients imply that only a small part of the clients is active at the same time. It also happens that active clients drop out and disconnect.

Challenges such as FL process delays from lagging clients and fault tolerance are compounded by these characteristics. Thus, FL methods must accept heterogeneous hardware, anticipate low participation, and be robust enough to accept failed clients.

4.4. Choosing the federated learning framework

The first step in designing an FL-system is choosing a framework. A framework is an interface for the developer with built-in features to develop the systems more easily and quickly. Every framework has different characteristics, making them suitable for different applications.

In this research six frameworks have been chosen for evaluation based on popularity. The RNLN prefers the frameworks to be compatible with Python. These frameworks and their characteristics can be seen in Table 4.1[15].

Every FL framework can be used in combination with one or more ML frameworks. These ML frameworks are libraries with standard algorithms to create ML models. The mode of the framework can be local simulation on one system only or both simulation and federated mode. For the RNLN it was important the framework could be used in federated mode to be able to test the FL system with different laptops that serve as a central server and distributed clients. Also, the framework has to be compatible with the Operating System (OS) Linux.

The ship data of the RNLN is expressed in the form of time series (such as temperatures, engine speeds and pressures), so the framework must also be able to work with time series. DP stands for Differential Privacy and is a technology that ensures privacy of the individual data sets of clients by adding noise to the local gradients. Lastly, it was important for the continuation of the research to have enough examples of implementations online.

Table 4.1. Overview characteristics FL frameworks

	Federated Learning Framework					
	TensorFlow Federated 0.19.0	**PySyft 0.6.0**	**Flower 0.17.0**	**FATE 1.7.1**	**OpenFL 1.2.1**	**PaddleFL 1.2.0**
ML framework	TensorFlow	PyTorch	Any	TensorFlow	PyTorch and TensorFlow	PaddlePaddle
Mode	Local simulation only	Local simulation only	Simulation and federated	Simulation and federated	Simulation and federated	Simulation and federated
Operating System	Mac, Linux	Mac, Linux, Win, iOS, Android	Mac, Linux, Win, iOS, Android	Mac, Linux	Mac, Linux	Mac, Linux, Win
Data type	Time series and pictures	Pictures	Time series and pictures	Time series	Time series and pictures	Time series and pictures
DP	Yes	Yes	Yes	No	No	Yes
Algorithms	(A)NN, CNN, RNN, LogR, PR, LR	(A)NN, CNN, RNN, LogR, PR, LR	(A)NN, CNN, RNN, LogR, PR, LR	GBDT, (A)NN, CNN, RNN, LogR, PR, LR	(A)NN, CNN	(A)NN, CNN, RNN, LogR, PR, LR
Available resources	Basic examples in the documentation	Tutorials and examples in the documentation, blogs and community projects	Tutorials and examples in the documentation, blogs and boilerplate examples	Basic examples in the documentation	Tutorials and basic examples in the documentation	Basic examples in the documentation

In Table 4.2 the overview of the RNLN criteria can be seen. The FL framework Flower possesses all of these characteristics and is the most flexible framework. It

can be used with every common ML framework, works on various OS and has the most literature available.

Table 4.2. Criteria for choosing the FL framework

Criteria	TensorFlow Federated 0.19.0	PySyft 0.6.0	Flower 0.17.0	FATE 1.7.1	OpenFL 1.2.1	PaddleFL 1.2.0
Python	✓	✓	✓	✓	✓	✓
Federated mode	×	×	✓	✓	✓	✓
OS Linux	✓	✓	✓	✓	✓	✓
Time series	✓	×	✓	✓	✓	✓
Examples of implementations	×	✓	✓	×	×	×

4.5. Design of the architecture with the flower framework

Now that we have chosen Flower as the FL framework, a central server and associated clients must be set up. Many different aspects must be considered in this design, such as how the server communicates with clients, which parameters of the ML model are shared, and how they are secured.

This section first describes the general Flower framework architecture. Then we explain the design of the server and clients in this study and how this could be used in practice. Figure 4.3 shows an overview of our set-up.

The figure shows on the server side the strategy, the FL loop and the Remote Procedure Call (RPC). The strategy determines how clients are selected, the training configuration, how the model updates are aggregated and the evaluation of the model. The FL loop monitors the learning process and ensures that progress is made. It requires a strategy in order to configure a round in the FL process and sends those configurations to selected clients via the RPC server. The resulting client updates from the local training are sent back through the RPC server and aggregated based on the chosen strategy.

Clients can connect to the RPC server responsible for sending and receiving Flower protocol messages and checking connections. The client side of Flower is simpler, as it only manages its own connection to the server and responds to incoming messages by calling the programmed training and evaluation functions[16].

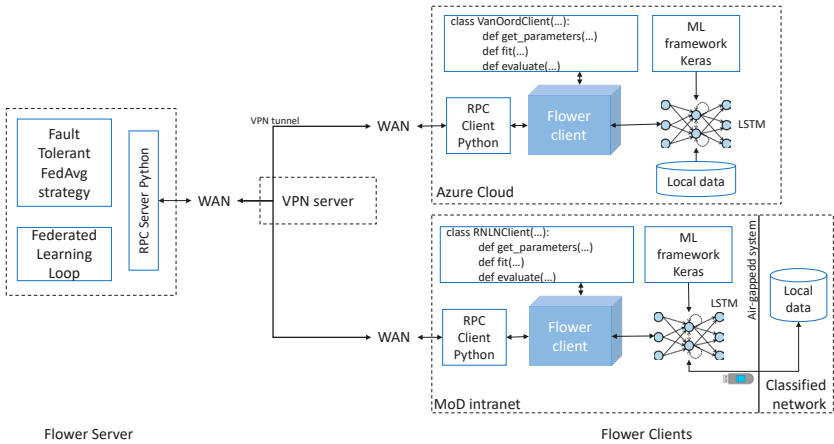

Fig. 4.3. The designed Flower framework architecture.

4.5.1. Build-in strategies

Section 2 describes the main characteristics of FL and the challenges arising from it. One of the challenges is client heterogeneity. Clients may communicate at different speeds, have different amounts of data points, or send wrong model updates to the server. Some of these challenges can be solved by choosing the right strategy depending on the amount and type of clients. Default built-in strategies within Flower are FedAvg, Fault Tolerant FedAvg, FedProx, qFedAvg and FedOptim.

Federated Averaging (FedAvg) is the standard FL strategy. It is a communication efficient algorithm in which the client first performs several updates locally and once per round the server averages, or aggregates, the model updates. Fault Tolerant FedAvg is a FedAvg variant that can deal with disconnecting or lagging clients[17].

FedProx basically works the same as FedAvg in that local updates are performed on the model and aggregated to the server once per round. It differs from FedAvg in that FedProx accepts that clients perform non-uniformly distributed amounts of work, making FedProx suitable for heterogeneous network conditions. FedOptim consists of a number of server-side optimizations for more efficient communication and consists of FedAdagrad, FedYogi and FedAdam[18].

In Section 2 the problem of bias in FL was introduced. If the aforementioned strategies are used on clients with non-i.i.d. data sets or on different clients where a certain type of device is predominant, the trained model can give a distorted picture and thus have a bias. qFedAvg is a method to solve q-Fair Federated Learning (q-FFL) and get more accurate solutions for non-i.i.d. data sets. q-FFL ensures that

clients with a higher loss and higher relative weight are given less variation in the distribution of accuracy[19].

Different strategies have been tested with an LSTM model and i.i.d and non-i.i.d. data sets[20]. In case of non-i.i.d. data sets, the accuracy of qFedAvg was higher than the other strategies, but the accuracy of qFedAvg was lower than the other strategies when training with i.i.d. data sets. Van Oord's data sets have an equal number of data points and are therefore i.i.d. There is also no question of network heterogeneity in this (test) set-up. Fault Tolerant FedAvg has been chosen as the strategy because it has a higher accuracy with i.i.d. data sets and it can deal with clients that fail also by a larger number of clients.

4.5.2. Choices in the design of the federated network

The client and central server can easily communicate with each other if both are on the same Local Area Network (LAN), but if several parties, such as RNLN and Van Oord, want to train an ML model with each other, a Virtual Private Network (VPN) can be used with which they can connect remotely.

In practice, it has to be taken into account that most ship data of the RNLN is classified and stored on an air-gapped network. Two solutions are possible. Preferably, the network has to be adjusted to run Python. Once this is possible, the FL process can be set up in such a way that the model gradients are brought via a USB stick to, for example, the Ministry of Defence (MoD) intranet and then sent to the central server. Another solution is to physically transfer the data sets to the MoD intranet, thereby lowering their classification level. On the MoD intranet it is possible to train ML models with Python and connect with a VPN. Since solving this issue was beyond the scope of this research, our proof of concept only uses data from Van Oord.

At Van Oord, the data sets are not stored on an air-gapped system, but in the Azure Cloud. It is possible to train ML models in Azure Cloud and connect to a VPN. So for future cooperation, no change in infrastructure at Van Oord is required.

Wireguard was used as a VPN server in this study. Wireguard is a communication protocol with free open-source software that implements VPNs. To use this, the clients were sent configuration files, allowing them to connect to the VPN and participate in the FL process remotely.

4.5.3. Communication during the FL process

As mentioned at the beginning of this chapter, the communication of the Flower server and Flower clients goes via the RPC. It is important to think about what kind of information is being sent by each party.

Communication starts from the Flower server which selects any number of clients to run the FL process with. When this minimal number of clients is connected, the server starts FL training.

Depending on the type of strategy and the number of connected clients, client dropout and reconnection affects the FL process. Fault Tolerant FedAvg was chosen as the strategy in this study, hence client dropout and reconnection are accepted if there are enough other clients left to meet the minimum number of clients. Depending on how many parties RNLN collaborates with, this number can be adjusted.

During the FL process, the central server and the clients communicate by sending each other local and global gradients. The local gradients are the model updates from the clients and the global gradients are the server aggregated model parameters. The type of gradients and their size depend on the type of model. The LR model in this study has only one gradient (the slope of the regression line), but an NN like the LSTM in this study has many more and this means more data traffic.

In this research, the client is a Python script, where the clients themselves install the necessary libraries to run FL[21].

4.5.4. Security and privacy

FL is often used in situations where the security of data and privacy of clients is very important. Although FL solves some problems of classic ML, there are still several possibilities for attackers.

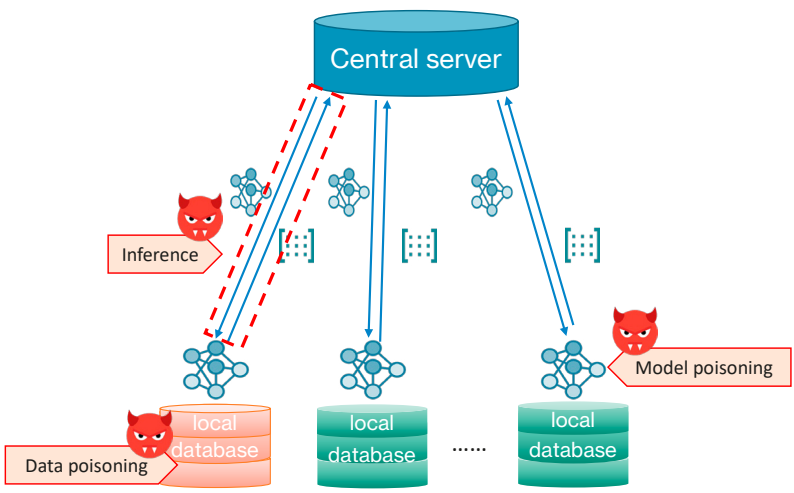

Fig. 4.4. Possibilities for attackers in the FL process[22].

Risks within FL are *poisoning* and *inference* attacks. Examples of these attacks are given in Figure 4. Poisoning can occur because a client adjusts the data set or model gradients or because the central server adjusts the global model. Inference can take place if outsiders can listen in on the data traffic between the server and clients.

Model poisoning or data poisoning is possible in FL because client data sets are not visible and clients are randomly selected. It is therefore not possible to check the quality of data sets or to check the history of a specific client. However, model poisoning or data poisoning can be prevented by regularly checking the trustworthiness of clients, for example by authentication via the VPN, and evaluating the behaviour of clients by checking for deviations.

If FL is to be used by RNLN and Van Oord, this will initially be with a small group of parties and a Trusted Third Party (TTP). This makes it easier for the central server to evaluate the reliability and behaviour of the clients. When RNLN collaborates with a larger group of external parties, random selection of clients in each round makes the effectiveness and invisibility of a poisoning attack more difficult[23,24].

In addition to poisoning, the FL process can be eavesdropped on by clients, the central server or outsiders, who thus obtain some of the local or global model gradients. With these gradients, attackers can reconstruct the data set by, for example, a Model Inversion Attack (MIA). The possibility of inference by outsiders is already reduced by using a VPN. In addition, gradient inversion methods are often unsuccessful in practice, because the architecture of the model and the global and local gradients must be known to the attacker[25,26]. If these components are all known to the attacker, *Differential Privacy* can be used to counter the attack.

4.6. Reliability and validity of the FL implementation

This section discusses the reliability and validity of the FL implementation. First, the used data sets of Van Oords's FFPV Stornes will be discussed. Next, a linear regression (LR) model was used to demonstrate the validity of the FL process. Subsequently, a script with the LSTM network was written for FL clients and a number of tests were performed to ensure the validity and reliability of the results of this model.

4.6.1. Available data of the FFPV Stornes

This section presents Van Oord's FFPV Stornes and the available data that is used in this paper for testing the FL architecture. The FFPV Stornes was designed for installation of rock on the ocean floor (in water depths up to 1500 meters) to stabilise and protect pipelines, cables and subsea templates[27]. The dataset of the

FFPV Stornes contains time series data from three of its auxiliary power generators and navigation data. Such generators are typically equipped with sensors (i.e. the engine speed, the power produced and the generator's fuel consumption) to provide for the remote monitoring and control services. The data of these generators is used for the LSTM model. The LSTM model learns the default behaviour for the generators by training on this data set. Detection of anomalies in all sensors may indicate a general problem in electrical operation, and detection of anomalies in the sensors of a specific generator may indicate a problem with this generator. The anomalies can be read in the warning table by a ship's crew or maintenance engineers ashore. This can lead to additional inspections on a generator and early detection of a problem if something is wrong with that generator. The navigation data is used to demonstrate and test whether the FL process works correctly using a LR model (see section 4.2).

The data set consists of NetCDF (NC) files, each containing data about the vessel for a time period of one day. The original frequency of the data by the monitoring system is equal to 25 points every second. In order to provide easier data handling and to train the model faster, the time interval has been reduced to 5 data points per second. No relevant information is lost in this process because the duration of anomaly is significantly longer than the new sampling rate. The available data of the vessel has been collected between 26/08/2021 and 04/09/2021. The exploited data of the FFPV Stornes is reported in Table 4.3

Table 4.3. Used sensor data of FFPV Stornes

Variable name	**Unit**
Timestamp	[t]
Latitude	[°]
Longitude	[°]
Heading	[°]
Heave	[°]
Roll	[°]
Pitch	[°]
Auxiliary Generators: power	[kW]
Auxiliary Generators: fuel oil consumption	[l/h]
Auxiliary Generators: engine speed	[rpm]

4.6.2. Linear Regression

The LR model was used to test the FL server and FL clients in our model. In Van Oord's data set, a linear correlation was coincidentally found between the change in latitude and longitude of FFPV Stornes, because the ship was moving more or less in a straight line. This data was therefore used as input for the LR model.

To test whether the FL process works correctly, different laptops were used with different OS (Linux, Windows and Mac) on which a data set from Van Oord and the FL client script were loaded. In the FL process, during the training of the LR model at the clients, several parameters were displayed every round on the server, and it was checked that they matched the values of the parameters at the clients.

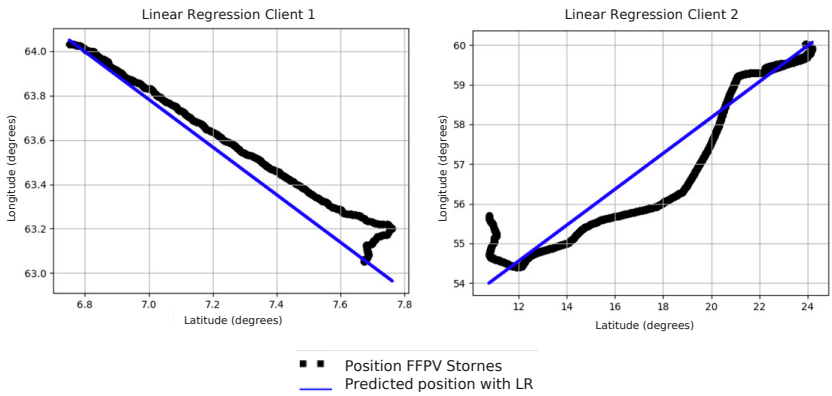

Fig. 4.5. Linear Regression results for FL with two clients.

Figure 4.5 shows an example of the output of the LR model of the two clients. These results indicate that during the FL process, the model's evaluation parameters are displayed correctly on the server and that the measurement is valid. Note that no aggregation is used here, since there was no linear correlation expected between the latitude and longitude on different days.

4.6.3. LSTM-network

An LSTM-network had already been deployed by RNLN in order to detect anomalies in sensor data of ships[28]. Our FL architecture uses an LSTM-network as an ML algorithm and uses the same hyperparameters and set-up as in previous research by Maaike Teunisse (2021)[29]. We refer there for a detailed explanation of how the LSTM-network is applied[30].

The LSTM network was first tested with a small part of the data set where there was a clear deviation in the four signals at one time point. The LSTM network is trained on the data set with the status parameters of the *heading*, *heave*, *roll* and *pitch* of FFPV Stornes. Figure 4.6 clearly shows an anomaly in the four signals at one time during the month.

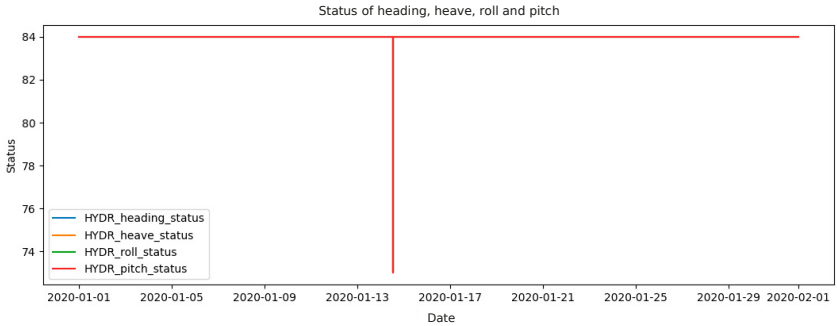

Fig. 4.6. Graph with status parameters per day of FFPV Stornes (note: dimensionless status lines are overlapping).

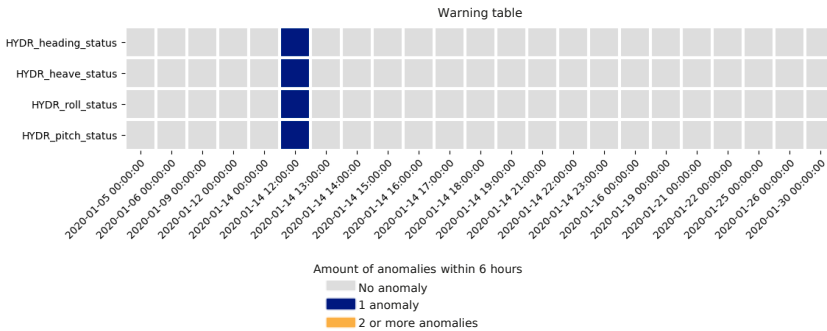

Fig. 4.7. Warning table with the detected anomalies in the data from Figure 4.6. Units as given in Table 4.3: fuel oil consumption in [l/h], engine speed in [rpm], power in [kW].

The LSTM model was tested for anomalies using the data set from Figure 4.6. The result of the training is the warning table shown in Figure 4.7. In this figure, a clear anomaly can be seen for all systems and this corresponds to the time of the deviation in Figure 4.6.

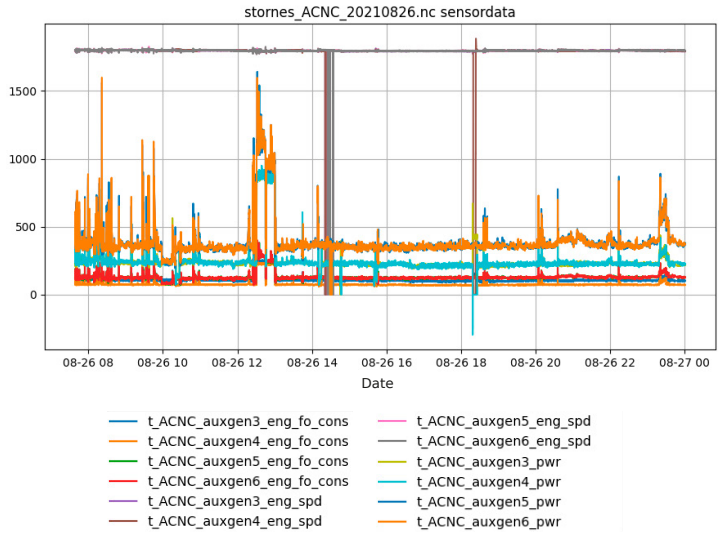

Fig. 4.8. Sensor data generators 3, 4, 5 and 6 from FFPV Stornes.

There was little variation in the sensor data in this data set, but the LSTM network successfully carried out the training. The same test was then performed on a data set with less obvious deviations. Figure 4.8 shows the signals from the data set from generators 3, 4, 5 and 6 over the period of one day. At 12:00, 14:00 and 18:00 deviations can be seen, and these deviations are reflected in the warning table in Figure 4.1.

A simpler example with one clear anomaly and an example where the anomalies were more difficult to read showed that the LSTM model is valid and the training of the model was successful.

To guarantee the validity of the study, the fit of the model was also taken into account. The appropriate parameters for the evaluation of the LSTM model have been investigated and scenarios where clients use non-i.i.d. data sets[31].

In addition to validity, a number of factors were taken into account to increase the reliability of the study. An important part of this is the repeatability of this study. During the research, the model was often trained, modified and retrained again. The functions *seed(...)* and *set_seed(...)* were used to ensure that identical results are generated for training rounds with the same software and data sets. These functions ensure that the random processes that take place in the NN can be repeated.

4.7. Results: central machine learning versus federated learning

The goal of this research is to evaluate the possibilities for RNLN to use FL for collaboration with external partners. To achieve this, we compare the LSTM model obtained from this FL set-up with central ML. For the computer code and the tables used for creating the graphs in this section, we refer to the appendices of the research of Anna Vriend (2022)[32]. To compare FL with central ML, the parameters *loss* and *accuracy* were used. In addition, the data traffic between the central server and the clients is monitored with Wireshark.

First, the loss and accuracy of a centrally trained LSTM model were obtained by training the LSTM model with one day of data from FFPV Stornes. After this, the LSTM network was trained again with FL with various laptops via a VPN, using the same data. To obtain the loss and accuracy, the global model was evaluated by the server after aggregation of the gradients. To simulate an increasing number of clients, each client had as local data a data set from FFPV Stornes of one day. The same FL process was performed in two rounds, whereby the central server thus aggregated the gradients twice. The data traffic was still below the limit of central ML.

The traffic at central ML is calculated by adding the size of the client data sets to the size of the LSTM model multiplied by the number of clients. This is done because with central ML the data sets must first be collected at a TTP and then the trained model must be distributed to all clients.

The results are summarised in two figures. Figure 4.9 shows an overview of the evaluation parameters of the LSTM model at one and two rounds of FL. The accuracy increases in both cases with a larger number of clients, and therefore more data. With one round of FL, 6 clients are needed to exceed the accuracy of central ML and with two rounds of FL, this already happens with 2 clients. The loss at two rounds of FL is lower than at one round, but in both cases the loss is not as low as at central ML.

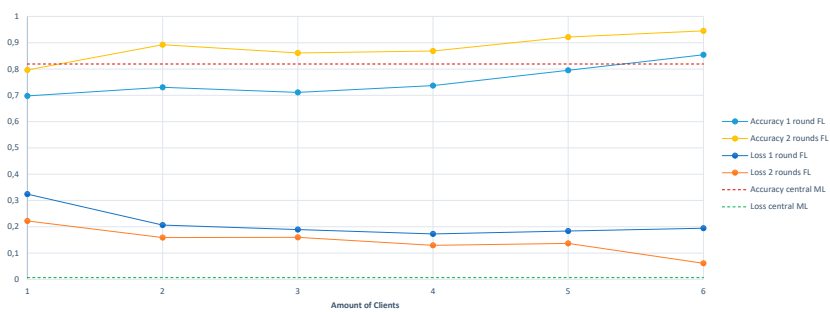

Fig. 4.9. Evaluation parameters LSTM-model 1 versus 2 rounds of FL.

In addition to the evaluation parameters for one and two rounds of FL, the data traffic between the clients and server at FL versus central ML was also examined. Figure 4.10 shows that the amount of data traffic during FL at one and two rounds remains below the amount of data traffic during central ML. The amount of data traffic during central ML increases linearly.

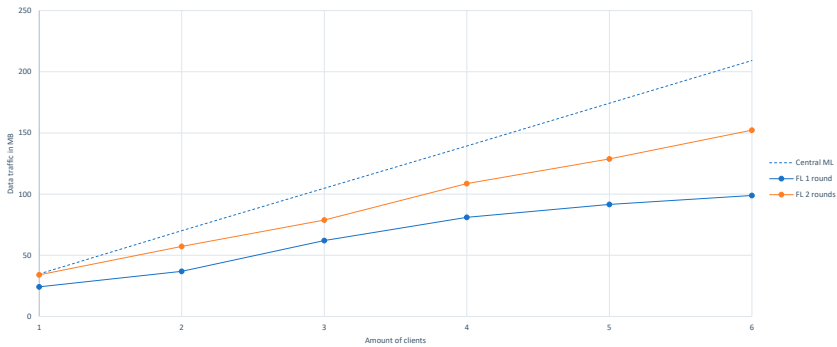

Fig. 4.10. Comparison of data traffic central ML versus FL with 1 and 2 rounds.

4.8. Conclusion

Nowadays, Machine Learning is used successfully in many applications. There are two important challenges involved with the implementation of this technique. The first is that in many industries, data is stored in separate places. The second challenge is the growing demand for ML that respects the privacy of the training data. Conventional ML is based on centralised data collection and is thus unable to face these challenges[33].

In this paper, an FL architecture has been designed using the Flower FL framework to train an LSTM model of the RNLN on data of Royal Van Oord's FFPV Stornes. Tests have been conducted in this work that show that it is possible to apply FL to the execution of this ML algorithm if the central server and the clients are laptops connected to different networks. An LR model was used to test the FL server and FL clients in the designed architecture. Next, an example with one clear anomaly and an example where the anomalies were more complex showed that the LSTM model is valid, and the training of the model was successful.

In both cases, the model accuracy increased, as expected, with a larger number of FL clients. With one round, 6 FL clients were needed to exceed the accuracy of central ML and with two rounds 2 FL clients were required. The loss at two rounds

of FL was lower than at one round, but in both cases the loss was not as low as with central ML. This work herewith showed that FL with multiple clients outperforms central ML in terms of accuracy (Figure 4.9) and is therefore a valuable solution for working together on Predictive Maintenance problems. Figure 4.10 shows that the amount of data traffic during FL at one and two rounds remained below the amount of data traffic during central ML. This effect will be exaggerated when large data sets are used for training the ML model, as is often the case for Predictive Maintenance problems.

If a TTP is appointed by the RNLN and Van Oord, it is technically possible to train the discussed LSTM network using FL. The RNLN should however take into account the defence security policy before the FL client can be deployed on the MoD intranet. This security policy has not been taken into account in this study. For air-gapped or disconnected networks the FL process can be set up in such a way that the model gradients are brought via a data carrier (i.e., a USB drive) to the central server. This can be relevant for networks such as the highly classified MoD networks, but also networks aboard warships.

Following the results of this study it can be concluded that it is relevant for the RNLN and Royal Van Oort to use FL. FL is a way to achieve using more data for training their ML models without the need to share their raw data. This leads, in the generic sense, to better predictions, as the tests carried out in this work have confirmed. The RNLN's LSTM model, which was able to detect anomalies in sensor data of Van Oord's FFPV Stornes, has been trained with data sets from only one ship. In practice, value lies in collaboration with multiple parties that have different (types of) ships. It should be taken into account that anomalies in the sensors used in this study may manifest themselves differently on other ships. If the model from this research is to be used, it is important that equivalent data is available and sensor data from similar types of components is used.

Finally, opportunities for FL are recognised in applying FL for one of the parties' own fleet, the ships will be the clients and a server on shore is the central server.

Notes

1 Peter Hulsen, "Wat is (het verschil tussen) Artificial Intelligence, Machine Learning en Deep Learning," last modified August 7, 2017, https://cqm.nl/nl/nieuws/wat-is-het-verschil-tussen-artificial-intelligence-machine-learning-en-deep-learning.

2 H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson and Blaise Agüera y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," last modified February 28, 2017, https://arxiv.org/pdf/1602.05629.pdf.

3 Cem Dilmegani, "What is Federated Learning?" last modified February 9, 2022, https://research.aimultiple.com/federated-learning/.

4 Wieger Tiddens, Jan Braaksma and Tiedo Tinga. “Exploring predictive maintenance applications in industry.” Journal of quality in maintenance engineering (2020).

5 Tiedo Tinga, Axel Homborg and Chris Rijsdijk. “Data-driven maintenance of military systems – potential and challenges.” Netherlands Annual Review of Military Studies (2022).

6 Maaike Teunisse, “Anomalie detectie voor voorspelbaar onderhoud” (bachelor’s thesis, Hogeschool van Amsterdam, 2021).

7 Yong Cheng, et al. “Federated learning for privacy-preserving AI,” Communications of the ACM 63, no. 12 (2020): 33-36.

8 Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konecny, Stefano Mazzocchi, H. Brendan McMahan,Timon Van Overveldt, David Petrou, Daniel Ramage and Jason Roselander, “Towards Federated Learning At Scale: System Design,” last modified on March 22, 2019, https://arxiv.org/pdf/1902.01046.pdf.

9 Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konecny, Stefano Mazzocchi, H. Brendan McMahan,Timon Van Overveldt, David Petrou, Daniel Ramage and Jason Roselander, “Towards Federated Learning At Scale: System Design,” last modified on March 22, 2019, https://arxiv.org/pdf/1902.01046.pdf.

10 Tim d’Hondt, “Federated Learning over Local Learning: an Opportunity for Collaboration” (master’s thesis, Eindhoven University Of Technology, 2020), 16-24.

11 Yong Cheng, et al. “Federated learning for privacy-preserving AI,” Communications of the ACM 63, no. 12 (2020): 33-36.

12 Ibid.

13 Tian Li, Ameet Talwalkar, Anit Kumar Sahu and Virginia Smith, “Federated Learning: Challenges, Methods, and Future Directions,” last modified August 21, 2019, https://arxiv.org/pdf/1908.07873.pdf.

14 Annie Abay, Yi Zhou, Nathalie Baracalo, Rajamoni Shashank, Ebube Chuba and Heiko Ludwig, “Mitigating Bias in Federated Learning,” last modified on December 4, 2020, https://arxiv.org/pdf/2012.02447.pdf.

15 Anna Vriend, “Federated Learning: Onderzoek naar de toepassing van privacy vriendelijke Machine Learning voor de Koninklijke Marine” (bachelor’s thesis, Netherlands Defence Academy, 2022).

16 Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qui, Titouan Parcollet, Pedro Porto Buarquede de Gusmao and Nicolas D. Lane, “Flower: A Friendly Federated Learning Framework,” last modified on April 7, 2021, https://akhilmathurs.github.io/papers/beutel_flower2020.pdf

17 Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qui, Titouan Parcollet, Pedro Porto Buarquede de Gusmao and Nicolas D. Lane, “Flower: A Friendly Federated Learning Framework,” last modified on April 7, 2021, https://akhilmathurs.github.io/papers/beutel_flower2020.pdf

18 Sahank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konecny, Sanjiv Kumar and H. Brendan McMahan, “Adaptive Federated Optimization,” last modified September 8, 2021, https://arxiv.org/pdf/2003.00295.pdf.

19 Tian Li, Maziar Sanjabi and Virginia Smith, “Fair Resource Allocation in Federated Learning,” last modified May 25, 2019, https://arxiv.org/pdf/1905.10497v1.pdf.

20 Ajinkya Mulay, Baye Gaspard, Rakshit Naidu, and Santiago Gonzalez-Toral, Vineeth S., Tushar Semwal and Ayush Manish Agrawa, “FedPerf: A Practioners’ Guide to Performance of Federated Learning Algorithms,” last modified December 11, 2020, https://proceedings.mlr.press/v148/mulay21a/mulay21a.pdf.

21 The Python version used is 3.9 and the Flower version is 0.17.0.

[22] Pengrui Liu, Xiangrui Xu, Wei Wang, "Threats, attacks and defenses to federated learning: issues, taxonomy and perspectives," last modified February 2, 2022, https://cybersecurity.springeropen.com/articles/10.1186/s42400-021-00105-6.

[23] Pengrui Liu, Xiangrui Xu, Wei Wang, "Threats, attacks and defenses to federated learning: issues, taxonomy and perspectives," last modified February 2, 2022, https://cybersecurity.springeropen.com/articles/10.1186/s42400-021-00105-6.

[24] Xingchen Zhou, Ming Xu, Yiming Wu and Ning Zheng, "Deep Model Poisoning Attack on Federated Learning," last modified March 14, 2021, https://www.mdpi.com/1999-5903/13/3/73/htm.

[25] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz, "Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning," last modified September 14, 2017, https://arxiv.org/pdf/1702.07464.pdf.

[26] Ali Hatamizadeh, Hongxu Yin, Pavlo Molchanov, Andiry Myronenko, Wenqi Li, Prerna Dogra, Andrew Feng, Mona G. Flores, Jan Kautz, Daguang Xu and Holger R. Roth, "Do Gradient Inversion Attacks Make Federated Learning Unsafe?" last modified February 14, 2022, https://arxiv.org/pdf/2202.06924.pdf.

[27] Van Oord, "Flexible fallpipe vessel", accessed April 26, 2022, https://www.vanoord.com/en/equipsment/flexible-fallpipe-vessel/.

[28] Maaike Teunisse, "Anomalie detectie voor voorspelbaar onderhoud" (bachelor's thesis, Hogeschool van Amsterdam, 2021).

[29] Ibid.

[30] Ibid.

[31] Anna Vriend, "Federated Learning: Onderzoek naar de toepassing van privacy vriendelijke Machine Learning voor de Koninklijke Marine" (bachelor's thesis, Netherlands Defence Academy, 2022).

[32] Anna Vriend, "Federated Learning: Onderzoek naar de toepassing van privacy vriendelijke Machine Learning voor de Koninklijke Marine" (bachelor's thesis, Netherlands Defence Academy, 2022).

[33] Yong Cheng, et al. "Federated learning for privacy-preserving AI," Communications of the ACM 63, no. 12 (2020): 33-36.

References

Abay, Annie, Yi Zhou, Nathalie Baracalo, Rajamoni Shashank, Ebube Chuba, and Heiko Ludwig, "Mitigating Bias in Federated Learning." Last modified on December 4, 2020. https://arxiv.org/pdf/2012.02447.pdf.

Beutel, Daniel J., Taner Topal, Akhik Mathur, Xinchi Qui, Titouan Parcollet, Pedro Porto Buarquede de Gusmao, and Nicolas D. Lane. "Flower: A Friendly Federated Learning Framework." Last modified on April 7, 2021. https://akhilmathurs.github.io/papers/beutel_flower2020.pdf

Bonawitz, Keith, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konecny, Stefano Mazzocchi, Brendan H. McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. "Towards Federated Learning At Scale: System Design." Last modified on March 22, 2019. https://arxiv.org/pdf/1902.01046.pdf.

Cheng, Yong, Yang Liu, Tianjian Chen, and Qiang Yang. "Federated learning for privacy-preserving AI." *Communications of the ACM* 63, no. 12 (2020): 33-36.

D'Hondt, Tim. "Federated Learning over Local Learning: an Opportunity for Collaboration." Master's thesis, Eindhoven University Of Technology, 2020.

Dilmegani, Cem. "What is Federated Learning?" Last modified February 9, 2022. https://research.aimultiple.com/federated-learning/.

Hatamizadeh, Ali, Honxu Yin, Pavlo Molchanov, Andiry Myronenko, Wenqi Li, Prerna Dogra, Andrew Feng, Mona G. Flores, Jan Kautz, Daguang Xu, and Holger R. Roth. "Do Gradient Inversion Attacks Make Federated Learning Unsafe?" Last modified February 14, 2022. https://arxiv.org/pdf/2202.06924.pdf.

Hitaj, Briland, Giuseppe Ateniese, and Fernando Perez-Cruz. "Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning." Last modified September 14, 2017. https://arxiv.org/pdf/1702.07464.pdf.

Hulsen, Peter. "Wat is (het verschil tussen) Artificial Intelligence, Machine Learning en Deep Learning." Last modified August 7, 2017. https://cqm.nl/nl/nieuws/wat-is-het-verschil-tussen-artificial-intelligence-machine-learning-en-deep-learning.

Li, Tian, Maziar Sanjabi, and Virginia Smith. "Fair Resource Allocation in Federated Learning." Last modified May 25, 2019. https://arxiv.org/pdf/1905.10497v1.pdf.

Li, Tian, Ameet Talwalkar, Anit Kumar Sahu, and Virginia Smith. "Federated Learning: Challenges, Methods, and Future Directions." Last modified August 21, 2019. https://arxiv.org/pdf/1908.07873.pdf.

Liu, Pengrui, Xiangrui Xu, and Wei Wang. "Threats, attacks and defenses to federated learning: issues, taxonomy and perspectives." Last modified February 2, 2022. https://cybersecurity.springeropen.com/articles/10.1186/s42400-021-00105-6.

McMahan, H. Brendan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. "Communication-Efficient Learning of Deep Networks from Decentralized Data." Last modified February 28, 2017. https://arxiv.org/pdf/1602.05629.pdf.

Mulay, Ajinkya, Baye Gaspard, Rakshit Naidu, Santiago Gonzalez-Toral, Vineeth S. Tushar Semwal, and Ayush Manish Agrawa. "FedPerf: A Practioners' Guide to Performance of Federated Learning Algorithms." Last modified December 11, 2020. https://proceedings.mlr.press/v148/mulay21a/mulay21a.pdf.

Phi, Michael. "Illustrated Guide to LSTM's and GRU's." Last modified September 24, 2018. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb-85bf21.

Reddi, Sahank J., Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konecny, Sanjiv Kumar, Brendan H. McMahan. "Adaptive Federated Optimization." Last modified September 8, 2021. https://arxiv.org/pdf/2003.00295.pdf.

Van Oord. "Flexible fallpipe vessel." Accessed April 26, 2022. https://www.vanoord.com/equipment/flexible-fallpipe-vessel/.

Vriend, Anna. "Federated Learning: Onderzoek naar de toepassing van privacy vriendelijke Machine Learning voor de Koninklijke Marine." Bachelor's thesis, Netherlands Defence Academy, 2022.

Teunisse, Maaike. “Anomalie detectie voor voorspelbaar onderhoud.” Bachelor’s thesis, Hogeschool van Amsterdam, 2021.

Tiddens, Wieger, Jan Braaksma, and Tiedo Tinga. “Exploring predictive maintenance applications in industry.” Journal of quality in maintenance engineering (2020).

Tinga, Tiedo, Axel Homborg and Chris Rijsdijk. “Data-driven maintenance of military systems – potential and challenges.” Netherlands Annual Review of Military Studies (2022).

Zhou, Xingchen, Ming Xu, Yiming Wu, and Ning Zheng. “Deep Model Poisoning Attack on Federated Learning.” Last modified March 14, 2021. https://www.mdpi.com/1999-5903/13/3/73/htm.