Privacy Preserving Technologies for Trusted Data Spaces

Susanna Bonura and Davide Dalle Carbonare and Roberto Díaz-Morales and Marcos Fernández-Díaz and Lucrezia Morabito and Luis Muñoz-González and Chiara Napione and Ángel Navia-Vázquez and Mark Purcell*

Abstract The quality of a machine learning model depends on the volume of data used during the training process. To prevent low accuracy models, one needs to generate more training data or add external data sources of the same kind. If the first option is not feasible, the second one requires the adoption of a federated learning ap-

---

*All authors have contributed equally and they are listed in alphabetical order:
Susanna Bonura
Engineering Ingegneria Informatica SpA - Piazzale dell'Agricoltura, 24 00144 Rome Italy, e-mail: susanna.bonura@eng.it

Davide Dalle Carbonare
Engineering Ingegneria Informatica SpA - Piazzale dell'Agricoltura, 24 00144 Rome Italy, e-mail: davide.dallecarbonare@eng.it

Roberto Díaz- Morales
Tree Technology - Parque Tecnológico de Asturias, CL Faya 18 33428 Llanera, Asturias, Spain e-mail: roberto.diaz@treetk.com

Marcos Fernández-Díaz
Tree Technology - Parque Tecnológico de Asturias, CL Faya 18 33428 Llanera, Asturias, Spain e-mail: marcos.fernandez@treetk.com

Lucrezia Morabito
COMAU Spa - Via Rivalta, 30 10095 Grugliasco, Italy, e-mail: lucrezia.morabito@comau.com

Luis Muñoz-González
Imperial College London - 180 Queen's Gate, SW7 2AZ, London, United Kingdom, e-mail: l.munoz@imperial.ac.uk

Chiara Napione
COMAU Spa - Via Rivalta, 30 10095 Grugliasco, Italy, e-mail: chiara.napione@comau.com

Ángel Navia-Vázquez
University Carlos III of Madrid, Avda. Universidad, 30, 28911-Leganés, Madrid, Spain. e-mail: angel.navia@uc3m.es

Mark Purcell
IBM Research Europe, IBM Campus, Damastown Industrial Estate, Mulhuddart, Dublin 15, Ireland. e-mail: markpurcell@ie.ibm.com

proach, where different devices can collaboratively learn a shared prediction model. However, access to data can be hindered by privacy restrictions. Training machine learning algorithms using data collected from different data providers while mitigating privacy concerns is a challenging problem. In this chapter we first introduce the general approach of federated machine learning and the H2020 MUSKETEER project, which aims to create a federated, privacy-preserving machine learning Industrial Data Platform. Then, we describe the Privacy Operations Modes designed in MUSKETEER as an answer for more privacy before looking at the platform and its operation using these different Privacy Operations Modes. We eventually present an efficiency assessment of the federated approach using the MUSKETEER platform. This chapter concludes with the description of a real use case of MUSKETEER in the manufacturing domain.

1 Introduction

In recent years, the advancements in Big Data technologies have fostered the penetration of AI and machine learning in many application domains, producing a disruptive change in society and in the way many systems and services are organized. Machine learning technologies provide significant advantages to improve the efficiency, automation, functionality and usability of many services and applications. In some cases, machine learning algorithms are even capable of outperforming humans. It is also clear that machine learning is one of the key pillars for the fourth industrial revolution and it will have a very significant impact in the future economy. Machine learning algorithms learn patterns and they are capable of extracting useful information from data [1]. But in some cases, the amount of data needed to achieve a high level of performance is significant. A few years ago, Peter Norvig, Director of Research at Google recognized: *"we don't have better algorithms, we just have more data."* Thus, apart from the expertise and the computational resources needed to develop and deploy machine learning systems, for many companies, the lack of data can be an important obstacle to participate in this new technological revolution. This can have a profound negative effect on SMEs, which, in many cases, will not be able to compete with the largest companies in this sector. In recent years this problem has been alleviated by the growth and development of data markets, which enables companies to have access to datasets to develop AI and machine learning models. However, in many sectors, the access to these data markets is very difficult because of the sensitivity of the data or privacy restrictions that impede companies to share or commercialize the data. This can be the case, for example, for healthcare applications, where the privacy of the patients must be preserved and where often the data cannot leave the data owner's facilities. The chapter relates to the technical priority "Mechanisms ensuring data protection and anonymisation, to enable the vast amounts of data which are not open data (and never can be open data) to be part of the Data Value Chain" of the European Big Data Value Strategic Research & Innovation Agenda [28]. It addresses the horizontal concern "Data Protection" of the

BDV Technical Reference Model. It addresses the vertical concerns “Data sharing platforms, Industrial/Personal”. The chapter relates to the "Systems, Methodologies, Hardware and Tools" cross-sectorial technology enablers of the AI, Data and Robotics Strategic Research, Innovation & Deployment Agenda [27]. In this chapter we present a solution developed for the MUSKETEER project based on a federated learning approach (Section 2) [15, 16, 18, 24], detailing the different Privacy Operations Modes designed to offer higher standards of privacy to the stakeholders of the platform (Section 3). From this conceptual level, we then deep dive into the concrete part with an architecture description of the platform (Section 4) and an efficiency assessment of the federated approach using the MUSKETEER platform (Section 5) before concluding with a real use case description (Section 6).

2 Tackling privacy concerns: the solution of federated learning

Federated learning is a machine learning technique that allows building collaborative learning models at scale with many participants whilst preserving the privacy of their datasets [15, 16, 18, 24]. Federated learning aims to train a machine learning algorithm using multiple datasets stored locally in the facilities or devices of the clients participating in the collaborative learning task. Thus, the data is not exchanged between the clients and always remains in their own facilities.

2.1 Building collaborative models with federated learning

In federated learning, there is a central node (or server) that orchestrates the learning process and aggregates and distributes the information provided by all the participants (or clients). Typically, the server first collects and aggregates the information provided by the different clients, updating the global machine learning model. Then, the server sends the parameters of the updated global model back to the clients. On the other side, the clients get the new parameters sent by the server and trains the model locally using their own datasets. This communication process between the server and the clients is repeated for a given number of iterations to produce a high-performance collaborative model. Depending on the application, there are different variants for learning federated machine learning models [16]. For example, in some cases, a global collaborative model is learned from all the clients’ data. In other scenarios, only part of the parameters of the machine learning models are shared between participants, whereas the rest of the parameters are local (specific for each client). This allows a level of customisation specific to the clients. This approach is also useful in cases where the datasets from the different participants are not completely aligned, i.e. they do not have exactly the same features.

2.2 Where can we use federated machine learning?

To reduce the amount of data to be stored in the cloud, as the computational power of small devices like smartphones and other IoT devices increases, federated machine learning offers new opportunities to build more efficient machine learning models at scale by keeping the data locally on these devices and using their computational power to train the model. This approach was used in the one of the first successful use cases of federated learning: *"Gboard"* for Android, a predictive keyboard developed by Google which leverages the smartphones of millions of users to improve the precision of the machine learning model that makes the predictions. In this case Google did not require upload of the text messages from millions of users to the cloud, which can be troublesome from a regulatory perspective, helping to protect the privacy of the users' data. This use case showed that federated machine learning can be applied in settings with millions of participants that have small datasets. Federated learning is also a suitable solution in applications where data sharing or access to data markets is limited. In these cases, the performance of the machine learning models achieved using smaller datasets for training can be unacceptable for their practical application. Thus, federated machine learning boosts performance by training a collaborative model that uses the data from all the participants in the learning task but keeps the data private. The performance of this collaborative model is similar to the one we would obtain by training a standard machine learning model merging the datasets from all the participants, which would require data sharing.

2.3 MUSKETEER's vision

Our mission in MUSKETEER is to develop an industrial data platform with scalable algorithms for federated and privacy-preserving machine learning techniques, including detection and mitigation of adversarial attacks and a rewarding model capable of fairly monetizing datasets according to the real data value. In Fig. 1 we represent the IDP concept.

In MUSKETEER we are addressing specific fundamental problems related to the privacy, scalability, robustness and security of federated machine learning, including the following objectives:

- To create machine learning models over a variety of privacy-preserving scenarios: in MUSKETEER we are defining and developing a platform for federated learning and distributed machine learning with different privacy operation modes to provide compliance with the legal and confidentiality restrictions of most industrial scenarios.
- To ensure security and robustness against external and internal threats: in MUSKETEER we are not only concerned with the security of our software but also with the security and robustness of our algorithms to internal and external threats. We are investigating and developing new mechanisms to make the algorithms ro-

bust and resilient to failures, malicious users and external attackers trying to compromise the machine learning algorithms [20].
- To provide a standardized and extendable architecture: the MUSKETEER platform aims to enable interoperability with Big Data frameworks by providing portability mechanisms to load and export the predictive models from/to other platforms. For this, the MUSKETEER design aims to comply with the Reference Architecture model of the International Data Spaces Association (IDSA) [13].
- To enhance the data economy by boosting cross-domain sharing: the MUSKETEER platform will help to foster and develop data markets, enabling data providers to share their datasets to create predictive models without explicitly disclosing their datasets.
- To demonstrate and validate in two different industrial scenarios: in MUSKETEER we want to show that our platform is suitable, providing effective solutions in different industrial scenarios such as manufacturing and healthcare, where federated learning approaches can bring important benefits in terms of cost, performance and efficiency.

3 Privacy Operation Modes (POMs)

Training models with data from different contributors is an appealing approach, since when more and more data is used, the performance of the resulting models is usually better. A centralized solution requires that the data from the different users is gathered in a common location, something that is not always possible due to privacy/confidentiality restrictions. The MUSKETEER platform aims at solv-

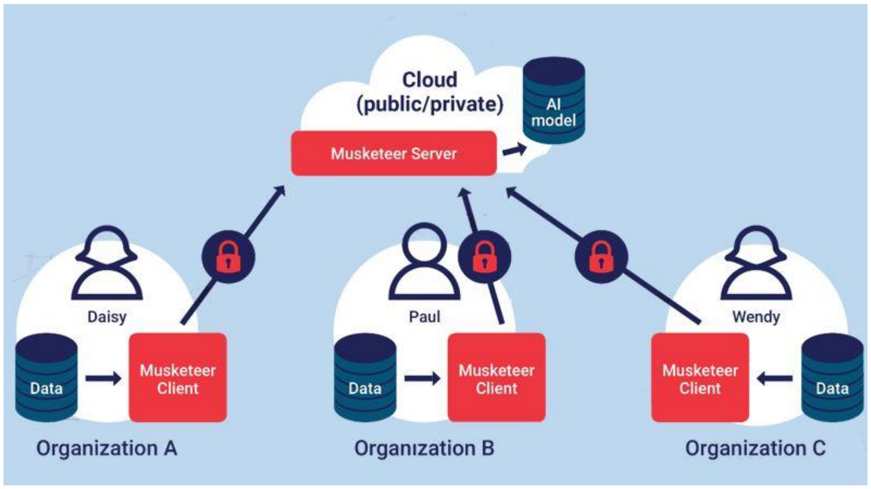

Fig. 1 Musketeer Industrial Data Platform to share data in a federated way.

ing Machine Learning (ML) problems using data from different contributors while preserving the privacy/confidentiality of the data and/or the resulting models. Essentially, it aims at deploying a distributed ML setup (Fig. 2) such that a model equivalent to the one obtained in the centralized setup is obtained. Nevertheless, even under the assumption that the raw data of the users is never directly exchanged or shared among the participants in the training process, any distributed privacy preserving approach requires the exchange of some information among the participants or at least some interaction among them, otherwise no effective learning is possible.

In the MUSKETEER Machine Learning Library (MMLL) we foresee different possible Privacy Operation Modes (POMs) with different assumptions/characteristics. In what follows, we briefly describe every one of the implemented POMs, each offering a variety of machine learning models/algorithms, but always respecting the privacy restrictions defined in that POM. In what follows, we will name as Master Node (MN) or Aggregator the central object or process that controls the execution of the training procedure and the Workers or participants will be running at the end user side as a part of the MUSKETEER client, and they have direct access to the raw data provided by every user.

3.1 POM 1

This POM is designed for scenarios where the final trained model is not private, since at the end of the training every worker node and also the master node have a copy the model. This POM implements a federated-learning framework based on

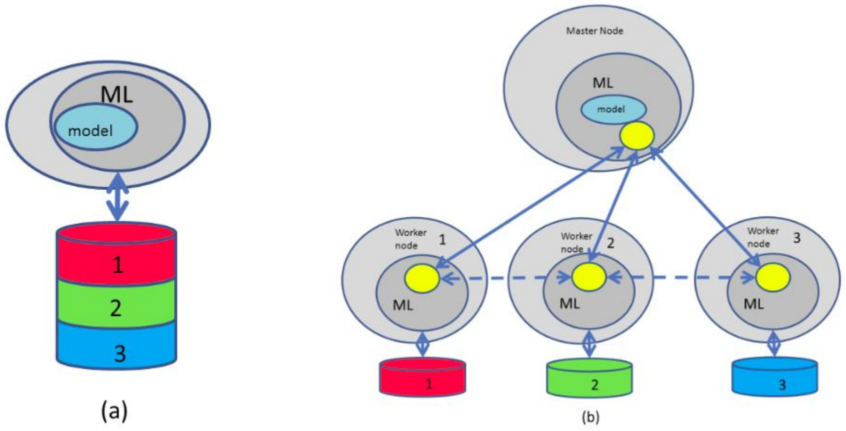

Fig. 2 Centralized approach (a) vs our privacy preserving distributed scenario (b) where every user provides a portion of the training set.

the concept introduced by Google in 2016 [15]. Under this paradigm [25], a shared global model is trained under the coordination of the central node, from a federation of participating devices. It enables different devices to collaboratively learn a shared prediction model while keeping all the training data on device, decoupling the ability to perform machine learning from the need to store the data in the cloud. Using this approach, data owners can offer their data to train a predictive model without being exposed to data leakage or data attacks. In addition, since the model updates are specific to improving the current model, there is no reason to store them on the server once they have been applied. Any model trainable by gradient descent or model averaging [14] can be deployed under this scheme. The model is not private, since every worker and the central node will receive it at every iteration. The algorithms developed under this POM conform to the following steps: The aggregator defines the model, which is sent to every worker. Every participant computes the gradient of the model or updates the model (for a model averaging schema) with respect to his data and sends aggregated gradient or the model back to the aggregator, which joins the contributions from all participants and updates the model. This process is repeated until a stopping criterion is met. Under this POM the model is sent to the workers unencrypted and the workers send an averaged gradient vector/updated model to the aggregator.

3.2 POM 2

In some use cases, data owners belong to the same company (e.g. different factories of the same company) and the server that orchestrates the training is in the cloud. The work proposed in [22] shows that having access to the predictive model and to the gradients it is possible to leak information. Since the orchestrator has access to this information in POM1, if is not completely under our control (e.g. Azure or AWS cloud), POM2 solves the problem by protecting the gradients over the honest-but-curious cloud server. This POM also implements the Federated Machine Learning (FML) paradigm described in POM1, but uses additively homomorphic encryption [26] to preserve model confidentiality from the central node. All gradients are encrypted and stored on the cloud server and the additive property enables the computation across the gradients. This protection of gradients against the server comes with the cost of increased computational and communication between the learning participants and the server. At the end of the training stage the model is known by all the workers but not by the aggregator. In this POM all the participants use the same pair of public/private keys for encryption/decryption. The aggregator has access to the public key but not to the private one, meaning that it cannot have access to the data encrypted by the workers. It comprises the following steps: every worker has the same pair of public/private keys, the public key is shared with the aggregator, which defines the model and encrypts it. The encrypted model is sent to every participant, which decrypts it with the private key and computes the gradients. Then, it sends back to the master the encrypted computed gradients. The aggregator finally averages the received gradients and updates the local copy of model weights in the encrypted domain thanks to the homomorphic encryption properties. The process is repeated until the end of the training. In this case the model is sent to the workers encrypted, the workers send encrypted gradient vector/updated model to the aggregator, who updates the model in the encrypted domain with no possibility of decryption.

3.3 POM 3

In POM2, every data owner trusts each other and they can share the private key of the homomorphic encryption (e.g. different servers with data that belongs to the same owner). Using the same key, every data owner uses the same encrypted domain. In many situations it is not possible to transfer the private key in a safe way. POM3 is an extension of POM2 that makes use of a proxy re-encryption [7] protocol to allow that every data owner can handle her/his own private key. This POM is an extension of POM2 that makes use of a proxy re-encryption protocol to allow that every data owner can handle her/his own private key [11]. The aggregator has access to all public keys and is able to transform data encrypted with one public key to a different public key so that all the participants can share the final model. The learning operation under this POM is as follows: every worker generates a different key pair randomly and sends the public key to the aggregator, which defines the initial model, such that it is sent unencrypted to every participant. Here the model is sent to every worker in its own encrypted domain in a sequential communication, the workers send encrypted gradient vector/updated model to the aggregator and the aggregator updates the model in the encrypted domain and uses proxy re-encrypt techniques to translate among different encrypted domains.

3.4 POM 4

This POM uses an additively homomorphic cryptosystem to protect the confidentiality of the data and requires the cooperation of an special node named as Crypto Node (CN), which is an object/process providing support for some of the cryptographic operations not supported by the homomorphism [9]. The CN is able to decrypt but it only receives encrypted data previously masked by the aggregator. The scheme is cryptographically secure if we guarantee that CN and Master Node (MN) do not cooperate to carry out any operation outside of the established protocols ("honest but curious" assumption). Therefore, the scheme is cryptographically secure if we guarantee that there is no collusion between the MN and the CN. The steps to train a given model are as follows: the MN asks the CN some general public encryption key and distributes it to the workers, which will use it to encrypt the data and send it to the aggregator. The MN starts the training procedure by operating on the (encrypted) model parameters and (encrypted) users data. The MN is able to perform some operations on the encrypted data (the ones supported by the homomorphism), but for the unsupported ones, it needs to establish a secure protocol with the CN consisting in sending some masked data to the CN, which decrypts the data, computes the unsupported operation in clear text, encrypts the result and sends it back to the MN, which finally removes the masking to obtain the encrypted desired result. As a result of repeating these operations during the training process, the MN never sees the data or the result in clear text, and the CN only sees the clear text of blinded/masked data. The training procedure continues until a stopping criterion is met. This POM is especially useful when the participants cannot actively cooperate during the training process (computational outsourcing) and the crypto node and master node are computationally powerful such that operations on the encrypted domain are highly efficient. Under this POM the MN, CN and Worker Node (WN) are assumed not to collude and they do not operate outside of the specified protocols ("honest but curious" assumption).

3.5 POM 5

This POM is able to operate only with the aggregator and worker nodes. It uses an additively homomorphic cryptosystem to protect the confidentiality of the model [8]. The data is also protected, since it does not leave the worker facilities, and only some operation results are sent to the aggregator. The MN will help in some of the unsupported operations, this is, the MN will play the role of CN. The scheme is cryptographically secure if we guarantee that the workers and aggregator do not operate outside of the protocols. The steps to train a given model are: the MN generates public and private keys and the public keys are distributed to all participants. The initial model parameters are generated at random by the MN. The MN encrypts the model parameters with the secret key and sends the encrypted model to the WNs, which starts the training procedure by operating on the (encrypted) model and (un-encrypted) users data. The WN is able to perform some operations on the encrypted data (the homomorphically supported ones), and for the unsupported ones it establishes a secure protocol with the MN such that the WN sends some encrypted data with blinding to the MN, which decrypts it, computes the unsupported operation in clear text, encrypts the result and sends it back to the worker, which finally removes the blinding to obtain the result. As a result of this protocol, the MN never sees the data or the result in clear text, and the WN only sees the encrypted model. The procedure is repeated for every needed operation of every algorithm, until a stopping criterion is met. The MN, CN and WN are assumed not to collude and they do not operate outside of the specified protocols ("honest but curious" assumption).

3.6 POM 6

This POM does not use encryption; it relies on Secure Two-Party Computation [4] protocols to solve some operations on distributed data such that both model and data privacy is preserved [2]. Under this POM, raw data is not encrypted, but it is never sent outside the WN. The model trained in the MN can also be kept secret to the WN. Some transformations of the data can be exchanged with the MN, such as aggregated or correlation values, but always guaranteeing that the raw data cannot be derived from them by reverse engineering. Every implemented algorithm will explicitly describe which information in particular is revealed to the MN, such that the participants are aware of this potential partial disclosure before participating in the training process. Some important operations can be directly implemented using protocols for secure dot product or secure matrix multiplication. The security of these operations will be as described in the reference sources of protocol and in general terms they prove that the raw data is not exposed. An improved security is achieved during some aggregation operations if a “round robin” or “ring” protocol is used such that global aggregations can be computed without revealing the specific contributions from every participant. POM6 is not a generally applicable procedure, it requires that every algorithm is implemented from scratch, and also it is not guaranteed that any algorithm can be implemented under this POM. The MN, CN and WN are assumed not to collude and they do not operate outside of the specified protocols (“honest but curious” assumption).

3.7 Algorithms

Over the different POMs, in the MMLL library, there are implementations of several algorithms. The library will contain algorithms capable to infer functions of different nature: Linear models: A simple but widely used class of Machine Learning models, able to make a prediction by using a linear combination of the input features [8]. MMLL includes alternatives for classification (logistic regression) and regression (linear regression) with different regularization alternatives and cost functions. Kernel Methods: They comprise a very popular family of Machine Learning models. The main reason of their success is their ability to easily adapt linear models to create non-linear solutions by trans-forming the input data space onto a high dimensional one where the inner product between projected vectors can be computed using a kernel function [12]. MMLL provides solutions for classification (SVMs) and regression (SVRs), possibly under model complexity restrictions (budgeted models). [19, 5, 6]. Deep Neural Networks: Deep learning architectures [17] such as multilayer perceptrons or convolutional neural networks are currently the state of art over a wide variety of fields including computer vision, speech recognition, natural language processing, audio recognition, machine translation, bioinformatics and drug design, where they have produced results comparable to and in some cases superior to human experts [10]. Clustering: Unsupervised learning is the machine learning task of inferring a function to describe hidden structure from "unlabelled" data. The library will include algorithms for clustering, that is the task of dividing the population or data into a number of groups such that data points in the same groups are more similar to other data points in the same group than those in other groups [21]. In simple words, the aim is to segregate groups with similar characteristics and assign them into clusters. The library will include general purpose clustering algorithms such as k-means [23].

4 Setting your own federated learning test case: technical perspective

How to train machine learning algorithms using data collected from different data providers while mitigating privacy concerns is a challenging problem. The concept of a trusted data space serves the need of establishing trusted networks where data can be transferred, accessed, and used in a secure mode, so to offer secure data access and transfer. This section presents the MUSKETEER platform in relation to how it leverages new paradigms and advancements in machine learning research, such as Federated Machine Learning, Privacy Preserving Machine Learning and protection against Adversarial Attacks. Moreover, the MUSKETEER platform, following actual industrial standards, it implements a trusted and secure data space, to enable scalable privacy-preserving machine learning in a decentralized dataset ownership scenario.

4.1 How to properly train your Machine Learning model?

That is definitely not an easy question and it depends on many different aspects. As it will take too much time to make a complete analysis of all of them, we will concentrate only on one of them: the amount of training data. The quality of a Machine Learning model depends on the volume of training data used during the training process. Small amount of data can produce low accuracy models that cannot be really usable. In this case, we can consider two options to solve the problem: (i) produce more training data by yourself or (ii) increase the training data volume by adding more data sources of the same kind. If the first option is not feasible (e.g. for technical or economic reasons), you can explore the second one by looking for other subjects with the same need. Here is where the concept of federation comes into play. In short, the model that can be produced thanks to the collaboration of the federation participants is better than the one produced by each participant on their own. This paradigm was initially introduced by Google and refers to different devices which collaboratively learn a shared prediction model while keeping all the training data on each device, decoupling the ability to do machine learning from the need to transfer data as well. The collaboration among the federation participants can be implemented with different levels of complexity and has to take into consideration other non-technical aspects. The simplest case is the one that concentrates all data in a single place and the training operation of the model is done using that single data repository. In this case, confidentiality and privacy should not be strong requirements. When the training data cannot be disclosed (e.g. business and/or legal reasons), a more sophisticated configuration has to be adopted. Every single federated participant will train an ML model locally (at their premises and not sending data outside) and will share only the model parameters. All the models produced by the participants are collected by a single subject that aggregates all of them and produces a new one that incorporates all contributions.

4.2 The MUSKETEER Platform

The main result of the MUSKETEER project is the implementation of an industrial data platform with scalable algorithms for federated and privacy-preserving machine learning techniques. The solution is based on the federation of a number of stakeholders contributing together, to build a Machine Learning model, in a collaborative (or co-operative) way. Different roles are to be assigned: (i) the aggregator, starting the process and taking charge of the computation of the final ML model; (ii) the participants, taking part in a single Machine Learning model process, built using their own (local) training datasets. From the architectural point of view, the MUSKETEER platform enables the interoperability between a number of distributed big data systems (federation participants) by providing a mechanism to send and retrieve Machine Learning models. That interoperability mechanism is based on the principles defined in the Reference Architecture Model of the International Data Space Association (IDSA) [13]. The MUSKETEER platform architecture consists of a server side and a client side. The server part is hosted in the cloud (as a number of micro-services) and it makes use of message queues for asynchronous exchange of information among the federation participants, that are often widely distributed geographically. One of the main activities of the server component is to coordinate the exchange of machine learning models between participants and aggregators. Besides the exchange of information for the execution of the actual federated learning tasks, the server side also provides services to manage tasks throughout their lifecycle, such as: creating new tasks, browsing created tasks, aggregating tasks, joining tasks as a participant or deleting tasks. The meta-information that is required for task management is stored in a cloud-hosted database.

4.3 MUSKETEER Client Connector components

The client side is represented by the MUSKETEER Client Connector that is a self-contained component that each user has to deploy on-premise in order to work with the MUSKETEER platform. We consider now the case where the training data is stored locally (e.g. in hard drives, Network Attached Storage (NAS) or removable devices that are attached to a single computer) and we want to make use of them to create predictive models without explicitly transferring datasets outside of our system. In this case, the MUSKETEER Client Connector can be deployed in any environment using Docker in order to containerize the Client Connector application itself. Docker containers ensure a lightweight, standalone and executable package of the software that includes everything needed to run the MUSKETEER Client Connector: operating system, code, runtime, system tools, libraries and settings. In this way the whole application can be easily made available in a sandbox that runs on the host operating system of the user. The MUSKETEER Client Connector consists of five core components and two additional ones that are loaded (as external plug-ins) after the application is up and running: the communication messenger and the federated machine learning library.

1. User Interface is a local web application that performs a set of functionalities where the main ones are: (i) to access to the target server platform; (ii) to connect the local data for executing the federated ML model training; (iii) to manage the different tasks for taking part in the federation.
2. Client Back-End acts as a RESTful Web Service that handle all user requests, ranging from local operations (e.g. to connect user data to the Client Connector) to server operations (e.g. tasks and user management); these operations need to use a Communication Messenger library to communicate with a target external server.
3. Data Connector connects user data, which may come from different sources or storage layers, to the Client Connector. In addition, to connect data from different source types, the component can manage and support different kinds of data: in

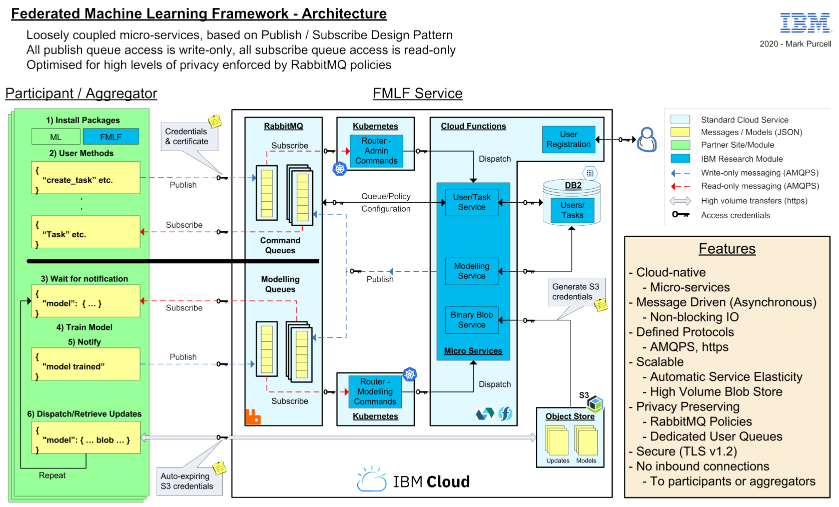

Fig. 3 MUSKETEER Platform Architecture.

fact, a user can load a *.csv* tabular data from the file system, images files, binary data, a table from a database and so on.

4. Abstract Communication Interface allows the import and use an implementation of the communication library. In the MUSKETEER project the Communication Messenger library used is the pycloudmessenger library developed by IBM [1]. After such a library is configured and installed, the MUSKETEER Client Connector can use the APIs to communicate with the MUSKETEER cloud server provided by IBM.
5. Execution component instantiates and runs federated machine learning algorithms according to interfaces defined by the Federated Machine Learning library imported into the MUSKETEER client Connector. In the MUSKETEER project, the FML library imported is provided by the partners TREE Technology and Carlos III de Madrid University.

4.4 Give it a try

The first prototype of the MUSKETEER Client Connector is available as open source software from GitHub repositories (Client Connector Backend[2], Client Connector

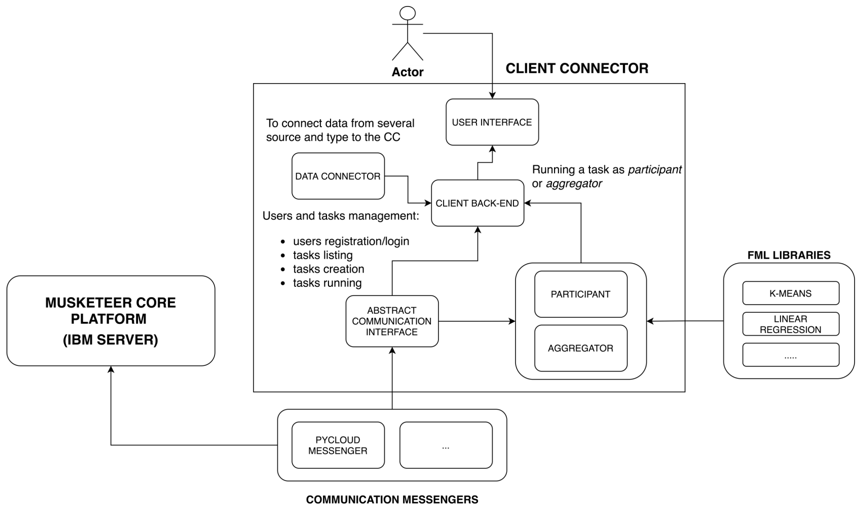
Actor
CLIENT CONNECTOR
USER INTERFACE
To connect data from several source and type to the CC
DATA CONNECTOR
Running a task as *participant* or *aggregator*
CLIENT BACK-END
Users and tasks management:
• users registration/login
• tasks listing
• tasks creation
• tasks running
FML LIBRARIES
K-MEANS
LINEAR REGRESSION
.....
PARTICIPANT
AGGREGATOR
MUSKETEER CORE PLATFORM (IBM SERVER)
ABSTRACT COMMUNICATION INTERFACE
PYCLOUD MESSENGER
...
COMMUNICATION MESSENGERS

Fig. 4 MUSKETEER Client Connector Architecture.

[1] https://github.com/IBM/pycloudmessenger

[2] https://github.com/Engineering-Research-and-Development/musketeer-client-connector-backend

Frontend[3]). Together with the source code you can also find the installation guide. We kindly invite you to download the Client Connector components and try it by setting up your test case of Federated Machine Learning process.

5 Federated Machine Learning in Action: An Efficiency Assessment

This section provides a test use case to show a practical application of Federated Machine Learning in the manufacturing context. The test bed aims to demonstrate, in a easily intuitive way, the potentiality of such algorithms in production environments.

5.1 Robots learn from each other

COMAU is an automation provider and its robots are installed in dozens of plants, in the automotive domain. In this context, these industrial automation customers are not eager to share their data. Nevertheless, those data are precious for the robot maker with regards to both business (new added value services for customers) and technical aspects (to improve robot performance and quality). Robot joints contain a belt that naturally loses its elasticity over time. With time and usage, the belt tensioning changes and actually, in order to prevent failures caused by a wrong tension, operators have to regularly check the belt status with a manual maintenance intervention. These operations require time, effort and eventually a production stop. Moreover, these manual tasks bear the risk to be useless if the belt status is still good. The identified solution implemented for this case is a Machine Learning model able to predict the status of the belt based on the analysis of a set of specific parameters. In order to achieve a solution that provides a reasonable quality of the prediction, it is extremely important to identify the proper training set to use. Given that the same kind of Robot may be operating in different plants and under control of different owners (customers of COMAU), it is possible to assume that: (i) each one of the customers may be interested in running the same predictive ML model for those specific class of robots so as to increase their efficiency; (ii) each one of the customers may face the issue of not having enough training data to produce an effective Machine Learning model suitable for the production: (iii) customers are not keen to open their data to third parties. In a similar scenario, the MUSKETEER platform comes into play enabling data sharing in a privacy preserving way for COMAU and its customers and thereafter with the possibility to use all available training data to build a classification model based on federated machine learning. To test the MUSKETEER platform functionalities and in particular the MUSKETEER

[3] https://github.com/Engineering-Research-and-Development/musketeer-client-connector-frontend

Client Connector, two test beds called RobotBox have been setup. The first RobotBox (from now on called “real RobotBox”) is located in COMAU headquarters in Turin and the second one (the “virtual RobotBox”) is a virtual simulator hosted in a ENGINEERING facility in Palermo. Both test beds are simulating two different customer plants.

5.2 Defining quality data

Going a bit more into the details, the RobotBox replicates an axis of a COMAU robot. It is composed of a motor, a belt, a gearbox reducer and a weight (yellow part in Fig. 5). In order to collect data at different belt tensioning levels, we had to space out the motor and gearbox from each other. To do this we installed a slicer to move the motor and a dial gauge to measure the distance between the two parts. We have decided to consider 3 different belt tensioning levels. The RobotBox always performs the same movement, called cycle. After each 24 second-cycle, we collected signals from the RobotBox motion controller. For each cycle we calculated 141 features describing the different signals aspects (e.g. mean value, maximum, minimum, root mean square, skewness, integral, etc.). Those features have been chosen based on the knowledge and literature about belt tensioning. For each level, we have performed around 6000 cycles, for a total of 18000 samples for each RobotBox, considering the 3 different belt tension levels. For this scenario, we have chosen to train a classification model using an artificial neural network. COMAU plays the role of the aggregator for the federated machine learning task and for this use case also the role of the first participant. On the other side, ENGINEERING plays the role

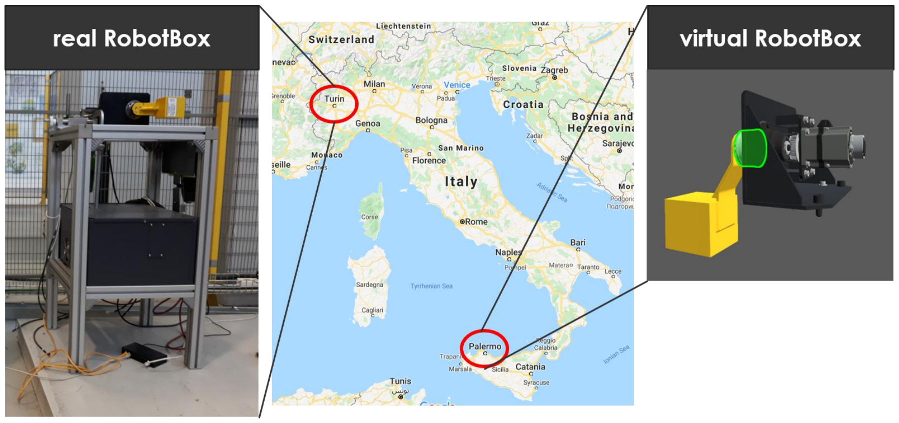

Fig. 5 RobotBox testbeds used for the experiment.

of the second participant. So, COMAU (located in Turin) created the task choosing the MUSKETEER Privacy Operation Mode 1 (POM1), where data cannot leave the facilities of the participants, the predictive models are transferred without encryption and the artificial neural network as algorithm. Moreover, the maximum number of iterations was set to 300, the learning rate to 0.00015. The architecture of the model shaped as a neural network is a 3 layer-neural network with 48, 16 and 3 units tuned in the preparatory phase of this evaluation. After the task creation, including validation and test data, COMAU joined the federated process as participant that provides the training data of the real RobotBox. Finally, ENGINEERING joined the same task providing the training data of the virtual RobotBox.

5.3 Federated data sharing is better than playing alone

The final results of the federated task are very promising in comparison with the results of a non-federated approach. Looking the right side of Fig. 6 it is possible to see how the overall accuracy obtained by the trained model, thanks to the federated task, with both RobotBoxes is 89% and the related confusion matrix is very diagonal. On the other hand, on the left side of Fig. 6, the accuracy of a model trained only with the data of the real RobotBox or only with the data of the virtual RobotBox is lower, respectively 86% and 81%.

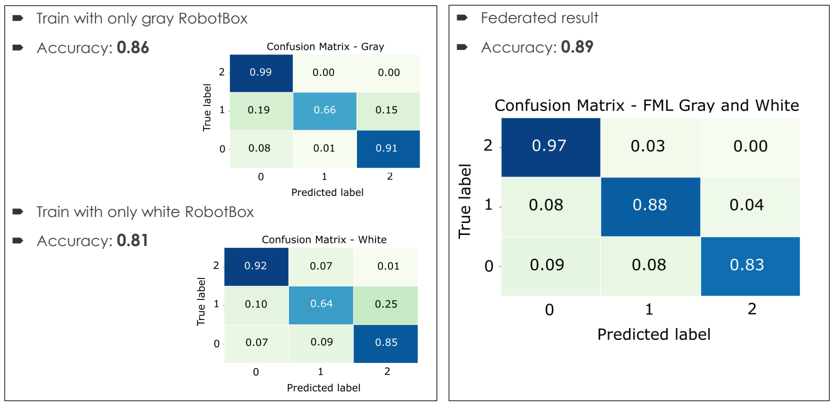

Fig. 6 Comparison of the non-federated and federated approach.

6 Use case scenario: Improving welding quality assessment thanks to federated learning

The emerging data economy holds the promise of bringing innovation and huge efficiency gains to many established industries. However, the confidentiality and proprietary nature of data is often a barrier as companies are simply not ready to give up their data sovereignty. Solutions are needed to realize the full potential of these new profit pools.

6.1 A twofold challenge

MUSKETEER offers to tackle these two dimensions by bringing efficiency while respecting the sovereignty of data providers in industrial assembly lines. Two challenges are presented:

- Improving welding quality assessment to develop predictive maintenance for robots while increasing product safety at the same time.
- Training a welding quality assessment algorithm on large datasets from multiple factories.

The presence of a huge number of machines in industrial automation factories and the elevated cost of downtime result in large expenses for production line maintenance. Getting a more accurate evaluation of robot performance helps to avoid damaging the production capacity contingently (by 5 to 20% in certain cases) as presented in the report Predictive maintenance and the smart factory (Deloitte) [3]. The welding quality assessment can be improved using machine learning algorithms which supports the status monitoring of machinery. But a single factory might offer too few data points to create such algorithms. It requires accessing larger datasets from Comau's robots located in different places to boost the robustness and quality of the machine learning model. However, Comau's customers can be competitors. Those companies do not intend to share data with competitors and simply waive their data sovereignty. With federated machine learning techniques, Comau can offer an appropriate level of security for customers and save them costs at the same time. Besides, the aforementioned data might include personal information regarding operators working in the manufacturing plant which can raise additional privacy concerns that have to be tackled by the solution.

6.2 Training an algorithm while preserving the sovereignty of data providers

At each piece's welding point, some welding parameters are recorded automatically from the welding robot (represented in Fig. 7), such as the current between the electrodes, resistance, number of points already welded by those electrodes and so on.

Then, a minimum amount of those pieces are sampled from the line to make an ultrasonic non destructive test to assess the welding spot quality. An operator applies a probe on each welded spot which sends a signal to a computer to be interpreted (graph representing the reflected sound energy versus time). The operator then classifies the welding points into different categories. In a production line only one or two pieces are verified each day whereas the total number of welded pieces goes up to 400 per day. Therefore, only a scarce percentage of pieces are subject to ultrasound testing. Besides, the manual test is limited to a few critical welding points on each piece, called Q+ points, that are always the same. On each car body, there are more than 3,000 welding spots while the Q+ points represent only a very small percentage of them. As an example, Fig. 8 shows which are the few Q+ points in a car door.

Our action here consists of collecting this manual ultrasound testing data and combining it with the welding data from the robot in order to locally train the algorithm. In parallel, this machine learning model is trained on different datasets from other factories. Trained models are eventually merged on the MUSKETEER platform (in different location) to provide the final version of the model. The entire flow is represented in Fig. 9.

As mentioned, collecting data from different factories also raises privacy issues. These data can be sensitive company data but also lead to personal data concerns (e.g. data can include information about operators working at the plant). Using the MUSKETEER platform provides a robust solution mixing a federated machine learning approach (local training) with privacy preserving technologies (highly customized encryption, data poisoning attacks mitigation) while respecting sovereignty of the

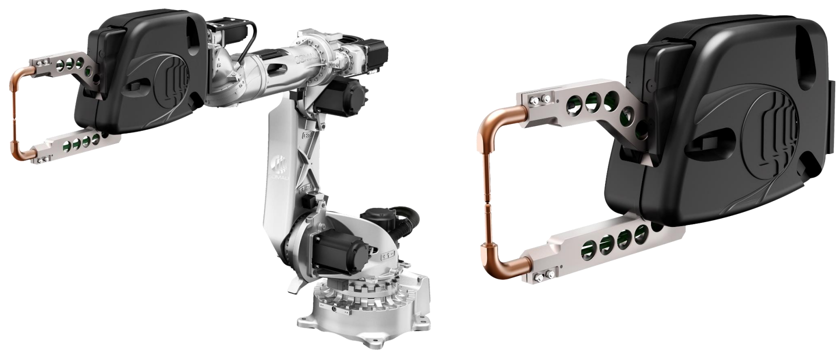

Fig. 7 Model of a Comau robot and welding gun used for the welding of car parts.

stakeholders as defined in the Reference architecture model of the International Data Spaces Association (IDSA) [13].

6.3 Less data but more information

Based on the combined data coming from robots and ultrasound tests, a robust model is built. In this case the Privacy Operation Mode selected is POM3 (Porthos), where the final trained model is private, the model is hidden to the aggregator and the encryption is implemented with a different private key for each data owner. One of the main reasons to adopt this model is because there is no trust among the data owners, for example in a scenario in which the data providers are competitors. Once the model is trained and has a satisfactory accuracy, thanks to the federated approach, it becomes possible to provide the classification of the welding spot directly from the welding data. This leads to numerous advantages over the limited manual testing:

- Opportunity to estimate the quality of all the welding points (not only the Q+ points) and raise the safety of products accordingly.
- Opportunity to understand if a specific combination of parameters helps to weld with fewer defects.
- Data sharing is allowed while sovereignty of each participant is preserved, and privacy concerns are tackled.

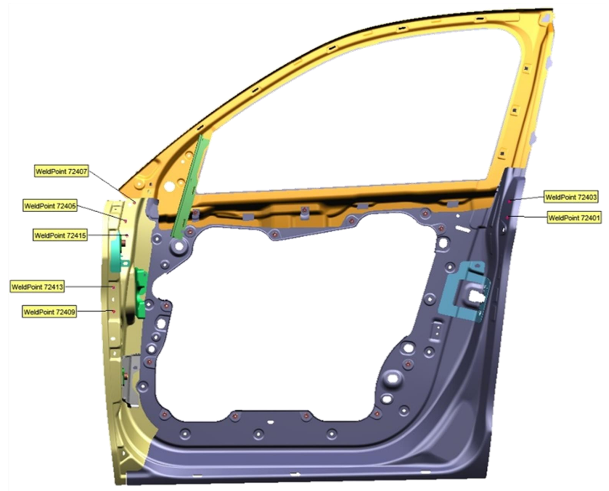

Fig. 8 Presentation of Q+ points on a car door.

7 Conclusion

In this chapter, we introduced MUSKETEER, a federated machine-learning-based platform and the custom Privacy Operations Modes designed for it, aiming at increasing privacy when sharing data. We presented an efficiency assessment of the federated learning process. The results showed an improved overall accuracy of a model trained in a federated task. The experiment was done using the MUSKETEER platform. The chapter concluded with the presentation of a real use case in the automotive sector to optimize the welding quality of robots in production lines. The data coming from different locations was used to create a federated task.

References

1. Bishop, C.M.: Pattern recognition and machine learning. Springer (2006)
2. Chen, V., Pastro, V., Raykova, M.: Secure Computation for Machine Learning with SPDZ. arXiv preprint arXiv:1901.00329 (2019)
3. Coleman, C., Damodaran, S., Deuel, E.: Predictive Maintenance and the Smart Factory. Tech. rep., Deloitte (2017)
4. Cramer, R., Damgård, I.B., Nielsen, J.B.: Secure Multiparty Computation. Cambridge University Press (2015)
5. Díaz-Morales, R., Navia-Vázquez, Á.: LIBIRWLS: A parallel IRWLS library for full and budgeted SVMs. Knowledge-Based Systems **136**, 183–186 (2017)
6. Díaz-Morales, R., Navia-Vázquez, Á.: Distributed Nonlinear Semiparametric Support Vector Machine for Big Data Applications on Spark Frameworks. IEEE Transactions on Systems, Man, and Cybernetics: Systems **50**(11), 4664–4675 (2018)

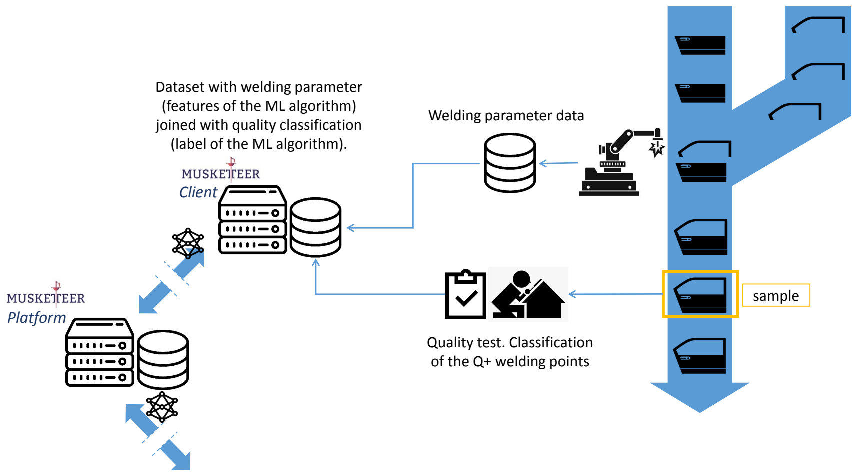
Dataset with welding parameter (features of the ML algorithm) joined with quality classification (label of the ML algorithm).
Welding parameter data
MUSKETEER Client
MUSKETEER Platform
sample
Quality test. Classification of the Q+ welding points

Fig. 9 Data collection and processing in federated learning environment.

7. Fuchsbauer, G., Kamath, C., Klein, K., Pietrzak, K.: Adaptively Secure Proxy Re-encryption. In: IACR International Workshop on Public Key Cryptography, pp. 317–346 (2019)
8. Giacomelli, I., Jha, S., Joye, M., Page, C.D., Yoon, K.: Privacy-Preserving Ridge Regression with only Linearly-Homomorphic Encryption. In: International Conference on Applied Cryptography and Network Security, pp. 243–261 (2018)
9. González-Serrano, F.J., Navia-Vázquez, Á., Amor-Martín, A.: Training Support Vector Machines with Privacy-Protected Data. Pattern Recognition **72**, 93–107 (2017)
10. Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y.: Deep Learning. MIT Press Cambridge (2016)
11. Hassan, A., Hamza, R., Yan, H., Li, P.: An Efficient Outsourced Privacy Preserving Machine Learning Scheme with Public Verifiability. IEEE Access **7**, 146322–146330 (2019)
12. Hearst, M.A., Dumais, S.T., Osuna, E., Platt, J., Scholkopf, B.: Support Vector Machines. IEEE Intelligent Systems and their applications **13**(4), 18–28 (1998)
13. IDSA: International Data Spaces Association (IDSA) Reference Architecture Model Version 3.0. Dortmund. https://internationaldataspaces.org/use/reference-architecture/ (2019)
14. Kamp, M., Adilova, L., Sicking, J., Hüger, F., Schlicht, P., Wirtz, T., Wrobel, S.: Efficient Decentralized Deep Learning by Dynamic Model Averaging. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 393–409 (2018)
15. Konečnỳ, J., McMahan, H.B., Ramage, D., Richtárik, P.: Federated Optimization: Distributed Machine Learning for On-Device Intelligence. arXiv preprint arXiv:1610.02527 (2016)
16. Konečnỳ, J., McMahan, H.B., Yu, F.X., Richtarik, P., Suresh, A.T., Bacon, D.: Federated Learning: Strategies for Improving Communication Efficiency. In: NIPS Workshop on Private Multi-Party Machine Learning (2016)
17. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
18. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-Efficient Learning of Deep Networks from Decentralized Data. In: Procs. of AISTATS, pp. 1273–1282 (2017)
19. Morales, R.D., Vázquez, Á.N.: Improving the Efficiency of IRWLS SVMs using Parallel Cholesky Factorization. Pattern Recognition Letters **84**, 91–98 (2016)
20. Muñoz-González, L., Co, K.T., Lupu, E.C.: Byzantine-Robust Federated Machine Learning through Adaptive Model Averaging. arXiv preprint arXiv:1909.05125 (2019)
21. Omari, A., Zevallos, J.J.C., Morales, R.D.: Nonlinear Feature Extraction for Big Data Analytics. Big Data Analytics: Tools and Technology for Effective Planning p. 267 (2017)
22. Phong, L.T., Aono, Y., Hayashi, T., Wang, L., Moriai, S.: Privacy-Preserving Deep Learning via Additively Homomorphic Encryption. IEEE Transactions on Information Forensics and Security **13**(5), 1333–1345 (2018)
23. Rao, F.Y., Samanthula, B.K., Bertino, E., Yi, X., Liu, D.: Privacy-Preserving and Outsourced Multi-User K-Means Clustering. In: IEEE Conference on Collaboration and Internet Computing (CIC), pp. 80–89 (2015)
24. Shokri, R., Shmatikov, V.: Privacy-preserving deep learning. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 1310–1321 (2015)
25. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated Machine Learning: Concept and Applications. ACM Transactions on Intelligent Systems and Technology (TIST) **10**(2), 1–19 (2019)
26. Yi, X., Paulet, R., Bertino, E.: Homomorphic Encryption. In: Homomorphic Encryption and Applications, pp. 27–46. Springer (2014)
27. Zillner, S., Bisset, D., Milano, M., Curry, E., García Robles, A., Hahn, T., Irgens, M., Lafrenz, R., Liepert, B., O'Sullivan, B., Smeulders, A., (Eds.): Strategic Research, Innovation and Deployment Agenda - AI, Data and Robotics Partnership. Third Release. September 2020, Brussels. BDVA, euRobotics, ELLIS, EurAI and CLAIRE (2020)
28. Zillner, S., Curry, E., Metzger, A., Auer, S., Seidl, R., (Eds.).: European Big Data Value Strategic Research & Innovation Agenda. Big Data Value Association (2017)

Acknowledgements This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 824988.