Security Centric Scalable Architecture for Distributed Learning and Knowledge Preservation

Rudolf Erdei[1], Daniela Delinschi[1], and Oliviu Matei[1]

Abstract This article presents the architecture, design and validation of a distributed learning approach, that provides support for knowledge preservation. The architecture is able to provide support for Collaborative Data Mining, Context Aware Data Mining, but also Federated Learning. Improving User Experience, providing support for research activities and providing a framework for production-grade machine learning pipeline automations were the primary objectives for the design of the proposed architecture. Third party service support is available out of the box, maintaining the loose-coupling of the system. Obtained results are promising, the system being validated with a use case on Collaborative Data Mining.

1 Introduction

Nowadays, *Distributed Learning (DL)* [6, 1] is one of the key research areas in the field of *Artificial Intelligence* and *Data Science*. The need for DL arises from the ever evolving need for *data security*, *data privacy*, *resources optimization*, and, in general, *cost control*. Traditional (centralized) Machine Learning (ML) methodologies use a single node to implement the entire *ML pipeline*, that has to be executed over the entire dataset, in order to obtain a working model. This working model can later be used by the Edge Nodes (or third-party locations) to obtain certain estimations or classifications for other input data.

This approach raises some important problems: training data must be uploaded from the Edge Nodes to the Cloud Component, that on the other hand has to be able to retain all that information while also possessing the computing power needed to process it. Time consuming pipeline operations are executed over the entire dataset at once.

Technical University of Cluj Napoca, North University Centre of Baia Mare, Romania e-mail: {rudolf.erdei,oliviu.matei}@cunbm.utcluj.ro

While the problem of data-privacy can be partially mitigated using *Homomorphic Encryption* [22], this operation does not resolve the bandwidth problem. The latter could be mitigated by using *Dimensionality Reduction* and *Data Reduction* [8] techniques, or even newer formats for data compression (like Parquet files [4]).

The question arises if both the privacy and the bandwidth problems could be optimally mitigated, by taking advantage of the steady increase in *low-energy high-power computing* that is currently available on the Edge Nodes. *Federated Learning* [26, 17] creates and trains partial models on the edge nodes, uploading them to the Cloud Component, that only has to integrate these partial models. This approach will mostly eliminate the need for a central node with high storage and processing requirements. Secure Computing [11] employs different strategies for providing Data Security, while also retaining the full power of Cloud Computing and delivering consistent results.

The structure of this article is as follows: Section 2 describes the latest work in the field of *Federated Learning* (FL), section 3 presents the use-case that inspired the work, section 4 presents the functional and non-functional requirements for the system, Section 5 discusses the architecture, section 7 presents advantages/disadvantages of the current architecture and also a comparison to a similar work, and lastly section 8 discusses conclusions and future work.

2 Related Work

As *Distributed Systems* (DS) progress turning the Internet into a huge scale organism, the importance of software and platform architectures grows. In this context, the *Pareto Principle* [24] enhances the importance of a well-designed architecture, that will save on implementation time later on in the life-cycle of the software platform.

Matei et al. [20] have developed a flexible architecture that makes use of *Serverless Components* for a face-recognition app, with very good results, thus demonstrating the effectiveness of the *Serverless Architecture*, and also a multi-layer architecture for soil moisture prediction [18] that is done in a *Context-Aware* methodology. Both these architectures make use of flexibility as well as layering in software components, in order to optimize all aspects of development and practical exploitation.

Data Silos are used heavily in *Distributed Learning* (actually every edge node is treated as a *Data Silo*). Durrant et al. [10] discusses about the role of DL in the Agri-Food sector. A soybean yield forecasting use-case is used for demonstration purposes, with good results, as the resulting model performed better than models trained on single data-sources.

Gupta et al. [11] present a methodology that uses *Machine Learning Agents* which work on subsets of the data, generating partial *Deep Neural Networks* (DNN) and then centralizing them and producing a final integrated model. The only data transfer that occurs between the central component and the edge nodes is composed of tensor parameters and gradients.

Li et al. [16] propose a modified version of *Federated Learning* architecture, where the system works on an overall model at once, agents being able to compute gradients and send them back to the central node. The system can handle *byzantine workers*, that send corrupt responses, due to local data corruption or any other causes (including maleficent behavior).

DL can also be regarded as a Bias Averaging System (reduction of overall bias in models by averaging multiple biased models). Alireza et al. [9] discuss about a methodology that brings *Adaptive Bias Compensation* in an energy efficient way. Their system uses a two-stage Analog to Digital Converter (ADC) with a two-fold adaptive feedback loop for signal processing and preparing the application of Data Mining Algorithms. Biases can thus be mitigated, if not eliminated completely.

Federated Learning Architecture for Smart Agriculture (SA) is discussed by Kumar et al.[15] in their article about the *PEFL* Framework, that securely raises the accuracy of the predictions up to 98%.

Model Trust is a topic discussed by Chakraborty et al. [5], where they propose storing the models within a blockchain. This approach, however, will affect the *privacy* of the model, as blockchains are inherently public, and also any encryption method will eventually be hacked in the future. The inability to remove blockchain entries can be detrimental for some use-cases.

3 Agricultural Use-Case for Distributed Learning

While data in the *Agri-Food* field may not be as sensitive as in other fields (eg.: the medical field), the sheer amount of sensor data generated by sensors of all kinds (often multiple sensor stations spread over a large area) can mean a huge bandwidth and storage space would be required just to upload and store the required data. Model parameters, however, would require much less storage and bandwidth, making this choice preferred, over the former.

MUSHNOMICS[1] is a three-stage holistic platform that enables mushroom growers to *1. prepare the substrate*, *2. grow the mushrooms* and *3. dispose the spent substrate* in an economic and eco-friendly way. The digital platform assesses the entire infrastructure at all times, through several sensing components, allowing the overseer to accurately predict the final yield of the growing system, in order to better plan the required logistics.

As mushroom growers often work with private data and there is some competition between them, the platform needs to be able to work with client specific models. This constraint, however, leads the way to generalization of the platform for all kinds of other Agri-Food use-cases, not just mushroom specific, that can be applied in the same platform, thus raising the flexibility of the result. Collaborative models for data-mining can also be implemented, if consortia of producers will agree to share their models (not their data), in order to better transfer new knowledge within the

[1] https://mushnomics.org/

group. Also, science use-cases will benefit from the new found knowledge, as the system should have a knowledge preserving system (ontologically modeled).

4 Functional and Non-Functional Requirements for the Distributed Learning Architecture

Within the project, requirements are elicited after brain-storming sessions that also included project beneficiaries as well as relevant stakeholders. After the initial idea generation, the structuring of requirements was done by using the *MoSCoW* methodology [2], that prioritizes features with most benefits for the end-result, and also using the *ATAM* Method [13]. Assessment of user expectations and the market demand for a privacy-centric solution were some other factors that were taken into consideration when designing these requirements.

Functional Requirements are formulated as high-level platform functionality. Each element was then split in more specific and atomic requirements needed for the development, testing and validation phases. From these, the most important ones are:

- FR1 - Ability to use multiple DL Methodologies, including *Federated Learning*, but also provide support for a classic (centralized) ML pipeline;
- FR2 - The ability of the system to provide both research support [3] as well as production grade automatic pipelines [21];
- FR3 - The ability of the system to accommodate several agri-producers in a secure, data-private way, but also enable support for consortia of agri-producers;
- FR4 - The system's ability to provide its services in a privacy centric way;
- FR5 - Ability to retain discovered knowledge, per project;
- FR6 - Ability to connect 3^{rd} party client applications/integrations to the system.

An excerpt from the **Non-Functional Requirements**, as performance and security aspects that the platform needs in order to provide quality services, are:

- NFR1 - *Security*, *Scalability* and *Stability* of the resulting system;
- NFR2 - *Platform Resilience*, as the ability to *manage and recover several failure* classes within the system;
- NFR3 - *Low-spec hardware requirements* for the Cloud Component (leveraging Serverless Components);
- NFR4 - *High-Performance* and *Flexible API* for 3^{rd} party integrations;
- NFR5 - *Portability/Accessibility*, as the possibility of the platform to be accessed from various operating systems, including mobile;
- NFR6 - *Flexibility*, as the platform should be used both for scientific as well as production purposes.

5 Distributed Learning System Architecture

DL methodologies have a huge benefit of using the edge nodes for generating a partial model. But, in order for the *Model Federation* to be obtainable, the models themselves have to be *compatible* [14]. The usual way of federating models is by means of averaging the parameter values over all the partial models.

Model characteristics are sent as metadata, so the Cloud Component can estimate an importance factor for it, factor that will be translated into the model's weight in the average. These operations ensure the robustness of the model and initial resistance against problems that can arise, including edge node byzantine failures [27].

Model compatibility is obtained in two steps:

1. **Model Build Characteristics** (like number of hidden layers and/or number of tensors per layer, for Neural Networks) and an initial model are obtained in the *research* part of the working methodology. This provides the best settings for the model to be built. These characteristics, that are unique per project, are then sent to the Edge Nodes;
2. Edge Nodes use the received characteristics and local data, to generate a **partial model** that will be uploaded to the Cloud Component. By using the same settings, averaging the models can be done in a more straightforward way.

In the case of FL, the Cloud Component does not permanently store the partial models (as this would pose some privacy risks), rather use the partial models to update the global model and then discard the used partial model.

The high level architecture is presented in Figure 1. We can observe the multiple *Edge Nodes*, that provide required project inputs to the *Cloud Component*, and also receive model parameters and other platform settings (like sensor read frequency). The *Cloud Component* centralizes either the full data (for classical ML methodologies) or the resulting partial models, in order to provide analytics, data/model storage, knowledge preservation and an advanced *model creation pipeline* that can either work in a classical manner, or in a decentralized (federated) one. Platform access can be provided either through the *Web UI*, or through an *API*.

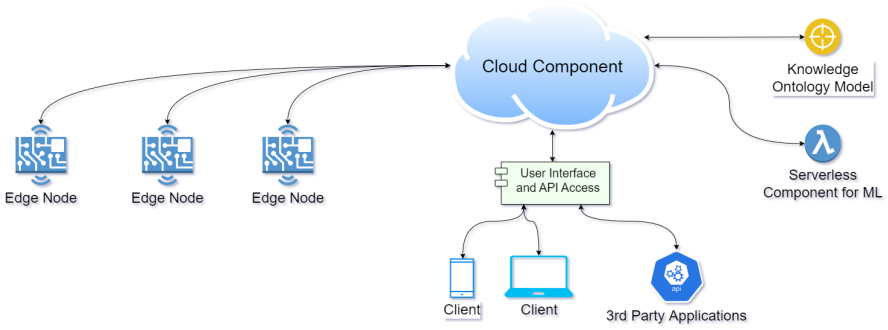

Fig. 1 High Level Architecture for Distributed Learning

The detailed architecture is presented in Figure 2.

The roles and responsibilities of the three large components of the system are:

The **Edge Nodes** are responsible for the local data gathering devices, centralizing and pre-processing data from sensors. Depending on the methodology, the data will be used either for generating/enhancing a local model, or uploading it to the *Cloud Component*, via the `Data Export Module`. If needed, the *Edge Nodes* can have a local administration UI, that will also present local statistics and *Node Health*.

The **Cloud Platform** Component is responsible for aggregating either data or partial models, generating the initial model scheme, providing support for *Edge Nodes Management*, *System Overview* and *Knowledge Management*.

The **Platform UI** Component is responsible with managing the communications with the clients, through the `Web UI Interface`, so its modules are focused solely on this task.

5.1 Cloud Component Architecture

It is composed of several subsystems that provide important services for the platform. These subsystems have very specialized roles, each composed of one or several modules that will cooperate in order to fully implement the designated requirements. A short description for the subsystems is:

- **Edge Communication Manager** - involves all the modules that receive or send information to/from the edge components, and is governed by the *Access Control Module*. The modules also provide the required encryption/decryption mechanisms that the platform will use;
- **Multi-Paradigm Data Ingestion Pipeline** - manages the *import* of data/partial models into the platform, checking the *data validity* and calculating *model quality* (see Section 8);
- **Data Persistence** - assesses if the received data can/must be persisted, and acts on this decision. Also, forwards received models to the FL methodology, that will trigger the *General Model Update*;
- **Knowledge Preservation** - gathered knowledge will enhance/update the already existing knowledge (the ontological model). These modules will keep the knowledge on a project basis, allowing the ontologies to be published or shared;
- **Multi-Paradigm Machine Learning Pipeline** - has multiple methodologies that can be selected per project, that use the data to generate models, or use partial models to update a general model.

6 Architecture Validation

Validation of the architecture was done by using *Software Architecture Analysis Method* (SAAM) [25] and *Active Reviews for Intermediate Designs* (ARID) [7]

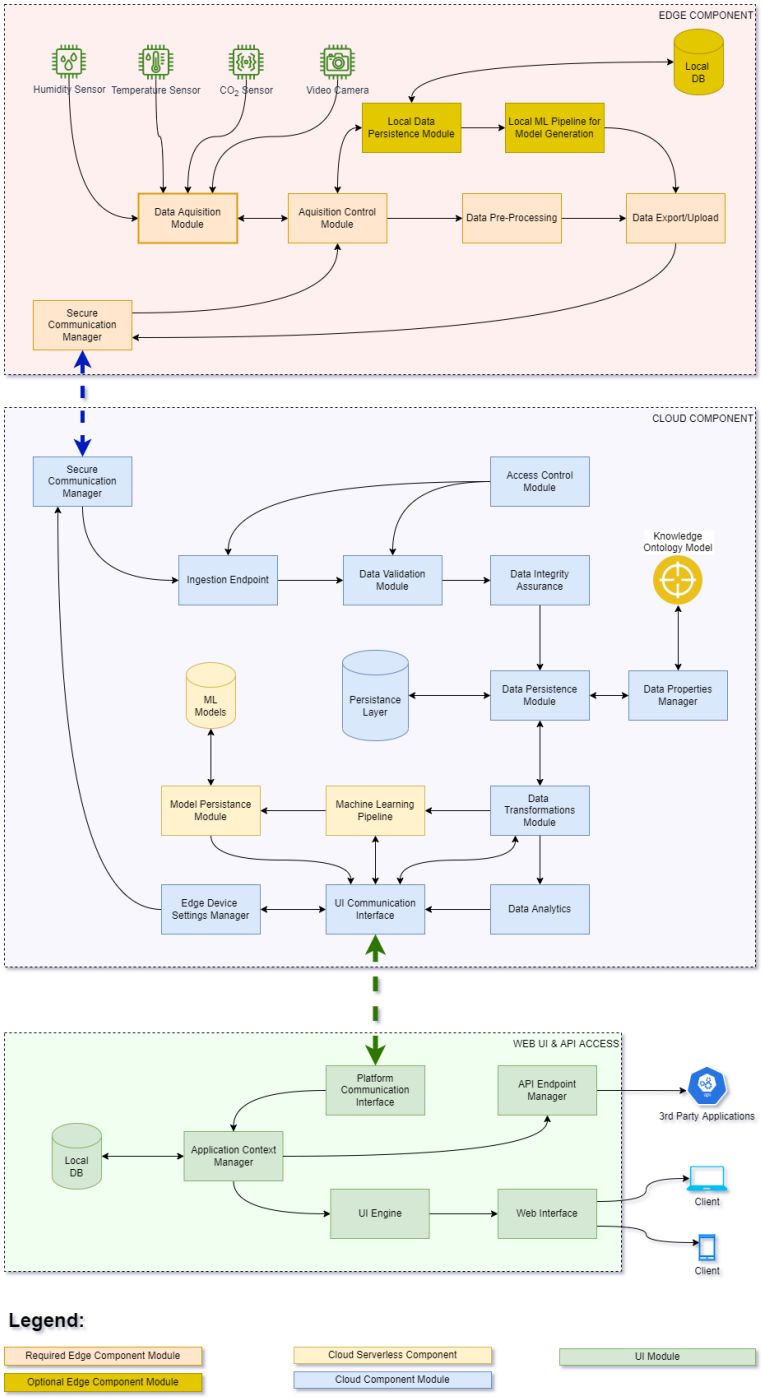

Fig. 2 Detailed Architecture for the Distributed Learning System

methodologies, that were used in architecture review meetings with the involved stakeholders.

The mapping between platform components and requirements was thoroughly analyzed and discussed, dissecting any raised concerns done by stakeholders and mitigating any potential issues. These methodologies lead to an Incremental design for the final architecture, having at least 4 increments until the architecture was finished. At the end of the validation process, the final architecture was proven to support all the requirements that the business case has required.

Table 1 Correlation between the Functional Requirements and modules withing the architecture

Functional Requirement	**Modules that are used for functionality implementation**
FR1	ML Modules within Edge Components, Model Integration pipeline in the Cloud Component
FR2	Flexible Model Generation Pipeline in the Cloud Component, UI Component, Communication with Edge Components
FR3	Access Control Module, Model Persistence Module
FR4	Federated Learning Methodology within the Machine Learning Pipeline
FR5	Access Control Module, Model Persistence Module, Knowledge Ontology Model
FR6	Access Control Module, API Endpoint Manager

7 Discussions

The proposed architecture provides the required architectural flexibility in order to integrate several Data Mining paradigms, including Federated Learning, which is the main focus of this research. The platform also provides secure communications, by encrypting all data transfer between edge nodes and the central Cloud Component. By limiting the amount of data transferred between components and changing way learning is tone, the required bandwidth is limited to a minimum, so the communication channel is a non-critical part of the system, unlike in many other ML paradigms.

Disadvantages include the inability of the system to perform realtime/time-critical operations. In some specific cases this aspect might be crucial. In our agricultural use-case, the time window for solving specific issues that the platform might predict is rather large (hours or maybe even days), so realtime performance is not needed in this case.

When compared to the *Collaborative Data Mining* (CDM) Architecture developed by Matei et al. [19], we can already observe some key differences:

- The proposed architecture will be able to also integrate CDM, that becomes in this case a local methodology;

- When using DL Methodologies (like FL), *Edge Nodes* will not need to upload data to the cloud, so privacy of user data is enhanced;
- *Physical Devices* are abstracted into *Edge Nodes*, that have greater importance and perform a larger range of *Data Transformations*;
- ML can also be done on the *Edge Nodes*;

8 Conclusions and Further Research

The presented architecture successfully integrates all functional and non-functional requirements, successfully addressing safety, security and trust issues regarding the edge nodes, over the medium of communication. Several layers of trust have been designed, in order to provide a *Multiple Point Fault Detection* system [12] providing a base for further research over *Building Trust in Distributed Machine Learning Paradigms*.

Further research will be done on the assessment for the *Quality of the Model*, as a subset of *Model Trust* [23], that has a huge importance as it defines the weight the model will have in the overall average of the project. A *Model Quality Index* is being researched, that will allow better general models to be generated from this methodology.

Acknowledgment. This research was made possible by funding from the ICT-AGRI-FOOD 2019 Joint Call. This work was supported by a grant of the Romanian National Authority for Scientific Research and Innovation, CCCDI-UEFISCDI, project number COFUND-ICT-AGRI-FOOD-MUSHNOMICS 205.2020, within PNCDI III.

Acknowledgment. This research was made possible by funding from the ICT-AGRI-FOOD 2019 Joint Call. This work was supported by a grant of the Romanian National Authority for Scientific Research and Innovation, CCCDI-UEFISCDI, project number COFUND-ICT-AGRI-FOOD-GOHYDRO-2 200.2020, within PNCDI III.

References

1. Maryam Alavi, George M Marakas, and Youngjin Yoo. A comparative study of distributed learning environments on learning outcomes. *Information Systems Research*, 13(4):404–415, 2002.
2. Thomas Bebensee, Inge van de Weerd, and Sjaak Brinkkemper. Binary priority list for prioritizing software requirements. In *International working conference on requirements engineering: foundation for software quality*, pages 67–78. Springer, 2010.
3. Hendrik Blockeel and Joaquin Vanschoren. Experiment databases: Towards an improved experimental methodology in machine learning. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 6–17. Springer, 2007.

4. Aikaterini Boufea, Richard Finkers, Martijn van Kaauwen, Mark Kramer, and Ioannis N Athanasiadis. Managing variant calling files the big data way: Using hdfs and apache parquet. In *Proceedings of the Fourth IEEE/ACM International Conference on Big Data Computing, Applications and Technologies*, pages 219–226, 2017.
5. Sarthak Chakraborty and Sandip Chakraborty. Proof of federated training: Accountable cross-network model training and inference. *arXiv preprint arXiv:2204.06919*, 2022.
6. Mingzhe Chen, Deniz Gündüz, Kaibin Huang, Walid Saad, Mehdi Bennis, Aneta Vulgarakis Feljan, and H Vincent Poor. Distributed learning in wireless networks: Recent progress and future challenges. *IEEE Journal on Selected Areas in Communications*, 2021.
7. Paul C Clements. Active reviews for intermediate designs. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 2000.
8. Ireneusz Czarnowski, Piotr Jedrzejowicz, Kuo-Ming Chao, and Tülay Yildirim. Overcoming "big data" barriers in machine learning techniques for the real-life applications, 2018.
9. Alireza Danaee, Rodrigo C de Lamare, and Vítor H Nascimento. Energy-efficient distributed learning with adaptive bias compensation for coarsely quantized signals. In *2021 IEEE Statistical Signal Processing Workshop (SSP)*, pages 61–65. IEEE, 2021.
10. Aiden Durrant, Milan Markovic, David Matthews, David May, Jessica Enright, and Georgios Leontidis. The role of cross-silo federated learning in facilitating data sharing in the agri-food sector. *Computers and Electronics in Agriculture*, 193:106648, 2022.
11. Otkrist Gupta and Ramesh Raskar. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, 116:1–8, 2018.
12. Rolf Isermann. Model-based fault-detection and diagnosis–status and applications. *Annual Reviews in control*, 29(1):71–85, 2005.
13. Rick Kazman, Mark Klein, Mario Barbacci, Tom Longstaff, Howard Lipson, and Jeromy Carriere. The architecture tradeoff analysis method. In *Proceedings. fourth ieee international conference on engineering of complex computer systems (cat. no. 98ex193)*, pages 68–78. IEEE, 1998.
14. Thomas Kühne. On model compatibility with referees and contexts. *Software & Systems Modeling*, 12(3):475–488, 2013.
15. Prabhat Kumar, Govind P Gupta, and Rakesh Tripathi. Pefl: Deep privacy-encoding based federated learning framework for smart agriculture. *IEEE Micro*, 2021.
16. Liping Li, Wei Xu, Tianyi Chen, Georgios B Giannakis, and Qing Ling. Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1544–1551, 2019.
17. Ji Liu, Jizhou Huang, Yang Zhou, Xuhong Li, Shilei Ji, Haoyi Xiong, and Dejing Dou. From distributed machine learning to federated learning: A survey. *Knowledge and Information Systems*, pages 1–33, 2022.
18. O Matei, C Anton, A Bozga, and P Pop. Multi-layered architecture for soil moisture prediction in agriculture 4.0. In *Proceedings of international conference on computers and industrial engineering, CIE*, volume 2, pages 39–48, 2017.
19. Oliviu Matei, Carmen Anton, Sebastian Scholze, and Claudio Cenedese. Multi-layered data mining architecture in the context of internet of things. In *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, pages 1193–1198. IEEE, 2017.
20. Oliviu Matei, Rudolf Erdei, Alexandru Moga, and Robert Heb. A serverless architecture for a wearable face recognition application. In *International Conference on Pattern Recognition*, pages 642–655. Springer, 2021.
21. Randal S Olson and Jason H Moore. Tpot: A tree-based pipeline optimization tool for automating machine learning. In *Workshop on automatic machine learning*, pages 66–74. PMLR, 2016.
22. Luis Bernardo Pulido-Gaytan, Andrei Tchernykh, Jorge M Cortés-Mendoza, Mikhail Babenko, and Gleb Radchenko. A survey on privacy-preserving machine learning with fully homomorphic encryption. In *Latin American High Performance Computing Conference*, pages 115–129. Springer, 2020.

23. Keng Siau and Weiyu Wang. Building trust in artificial intelligence, machine learning, and robotics. *Cutter business technology journal*, 31(2):47–53, 2018.
24. P Vishal and Sweta Bhattacharya. Application of the pareto principle in rapid application development model. *Citeseer*, 2013.
25. Christian Vogel. Saam (software architecture analysis method). *Universität Karlsruhe*, page 1, 2008.
26. Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. Federated learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 13(3):1–207, 2019.
27. Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, pages 5650–5659. PMLR, 2018.