2022 International Conference on Technologies and Applications of Artificial Intelligence (TAAI) | 979-8-3503-9950-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/TAAI57707.2022.00012

Federated Learning Operations (FLOps): Challenges, Lifecycle and Approaches

1st Qi Cheng
Faculty of Engineering and Information Technology
University of Technology Sydney
Sydney, Australia
qi.cheng-2@student.uts.edu.au

2nd Guodong Long
Faculty of Engineering and Information Technology
University of Technology Sydney
Sydney, Australia
guodong.long@uts.edu.au

***Abstract*—Federated Learning has witnessed a rapid growth in research and industry applications as it offers the benefits of privacy preserving while contributing to the global model training. Cross-silo federated learning systems which are usually geographically distributed and cross-organizational are becoming a reality. Although DevOps and MLOps methodologies may help improving traditional machine learning systems' development efficiency and productivity, it is still challenging for them to develop cross-silo federated learning systems in a productive way. In this paper, we propose FLOps (Federated Learning Operations), a new methodology for developing cross-silo federated learning systems efficiently and continuously. By elaborating the challenges that FLOps is facing, we construct the lifecycle of FLOps, and propose approaches to FLOps. Finally, we highlight potential research directions of FLOps.**

***Index Terms*—Software Development, Federated Learning, Machine Learning Operations, DevOps**

I. Introduction

With the increased awareness of personal data protection, Federated Learning is getting popular. Introduced by Google in 2017, Federated Learning was initially used to predict Google Keyboard typing text for Android mobile phones [1].

Cross-device FL and Cross-silo FL are two major settings of federated learning [2]. Cross-device FL is usually applied to mobilephone systems, IoT systems, wearable systems, and Edge computing systems [3]. Cross-silo FL is used widely in industries such as healthcare, finance, banks, manufacturing, and education [5] [21].

In a typical cross-silo federated Learning paradigm [2], there are one server and multiple clients. The clients train local models locally with the private data. The server has a global model and triggers the training process, collects model updates from the clients and aggregates a new global model. FedAvg [1], one of the well-known aggregation algorithms, averages all local model updates and merges into a new global model. The server then sends the new global model to clients to start a new round of training. Eventually, the model converges, and all clients can use it to do local prediction tasks on demand.

On the other hand, Software development methodologies have evolved significantly since the first appearance of the Waterfall Model in the 1960s. The Waterfall Model is followed by the Iterative Incremental Model, the Prototyping Model, the Spiral Model, then the Agile Method raised in 1990s. Agile Method employs the concept of Sprint to continuously adopt planning, coding, building, testing, and merging of the software features. DevOps was introduced in 2010s, and it emphasizes high-frequent delivery of software. DevOps, as a popular method in various companies, inspires the idea of MLOps(Machine Learning Operations) which combines machine learning, data engineering, and DevOps.

Some research works [2] [7] discuss the algorithms for cross-silo federated learning, but few articles address the process of developing a cross-silo FL system from the perspective of software engineering.

Our research question(RQ) is: *How can software engineering methodologies improve the development of cross-silo federated learning systems?*

A. Motivation

DevOps has significantly accelerated the general software delivery in the last decade. And MLOps has emerged to tackle machine learning system development. Federated Learning Systems are applied in many industries. Some of them follow the guidelines of DevOps and MLOps, and some don't. Therefore, the quality and cost of developing FL systems, especially the cross-silo FL systems, are usually out of control. MLOps obviously helps traditional machine learning systems in development and deployment efficiently, however it is still unsuitable for cross-silo federated learning systems because (1) MLOps neglects the cross-organization scenario, (2) MLOps doesn't highlight the data privacy preservation.

B. Contributions

We summarize the contributions in this paper as follows: (1) We elaborate on the challenges of building cross-silo federated learning systems using existing development methodologies. (2) We propose FLOps(Federated Learning Operations), an engineering methodology for developing and deploying cross-silo FL systems. (3) We address the key practical approaches to enable FLOps. (4) Finally, future research topics of FLOps are given for potential directions.

II. Literature Review

A. DevOps

DevOps has been widely adopted in the software industry recently, aiming to improve the software delivery process. It is

2376-6824/22/$31.00 ©2022 IEEE
DOI 10.1109/TAAI57707.2022.00012

a methodology for software development and deployment [10] [11]. It concatenates the DEV loop and the OPS loop [12], as shown in Fig. 1.

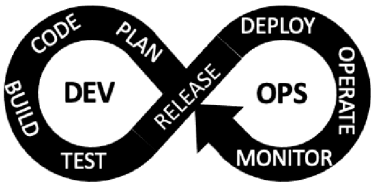

Fig. 1. Loops of DevOps[1]

DevOps delivers software continuously and efficiently including Plan, Code, Build, Test, Release, Deploy, Operation, Monitor. The key idea is monitoring software status continuously and delivering new features smoothly in a running production system. Automation is the vital concept in DevOps. DevOps pipeline continuously triggers building, testing, and publishing. The automation enables quick and frequent delivery of software, reduces the failure rate of new releases, shortens the leading time for fixes, and improves the software quality. Karamitsos et al. [13] applies DevOps practices of continuous automation for machine learning. It converts the ML manual pipeline into automation by utilising DevOps' CI/CD tools. It strongly depends on CI/CD practices but neglects the fact that data is distributed in FL.

Leite et al. [12] addresses the DevOps' challenges from three perspectives: engineers, managers, and researchers, and points out that effective communication among organizations and among departments is the key to DevOps adoption. Diel et al. [14] described the communication challenges and strategies in distributed DevOps. It highlighted three factors that impact team communications: Geographical Distance, Socio-culture Distance, Temporal Distance. The authors recommended to establish a regular formal communication channel. The limitation is that it only concentrated on the gap between the developers and the operations but neglected the paralleled teams, i.e. the developer teams in different locations. Alzoubi et al. [15] summarizes six categories of communication challenges on geographically distributed agile development. The top four categories are Distance Differences, Organizational Factors, Team Configurations, and Human Factors. It offers the potential techniques to overcome the challenges, for example, the Scrum of Scrums, information hub, and frequent review meetings. But they are too general to follow for FL system.

Forsgren et al. [16] emphasizes on measuring DevOps including system-base metrics and survey-base metrics to track the progress and gain a better view of software delivery. DevOps performance evaluations is also studied by Bezemer et al. [17]. It implicates that performance evaluation approaches must be integrated in the DevOps pipeline, which strengthens the irreplaceable value of orchestration of the DevOps pipeline.

[1]https://devopedia.org/devops

B. MLOps

MLOps, i.e., Machine Learning Operations, comes originally from the concept of DevOps and is committed to combine the development and the deployment of machine learning systems seamlessly, efficiently, and continuously. MLOps is an end-to-end engineering practice covering three disciplines [19]: data engineering, machine learning, and software engineering. Therefore, MLOps requires roles of not only business analysts and software engineers but also data scientists and machine learning engineers [20].

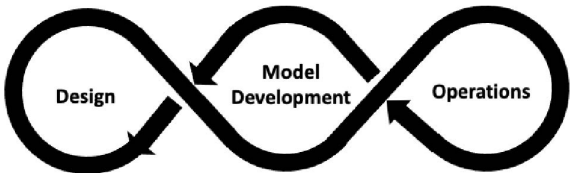

Fig. 2. Loops of MLOps[2]

MLOps, as shown in Fig. 2, has three phases included:

1) ML Model Design: In this phase, Business Analysts (BA) and Data Scientists (DS) analyse data according to the business requirements and determine the quality metrics of the model. Data Engineers (DE) then proceed with data extraction and transformation to prepare data for training, validation, and testing. ML Engineers (MLE) design the ML model structure and architecture and implement it.

2) ML Model Development: ML Engineers (MLE) load the training data to train and fine-tune the model parameters consecutively. Eventually, once the model performance meets the pre-defined quality metrics, the model is then released.

3) ML Model Operations: Once the model released, it's time to deploy the model to the production system by Software Engineers (SE). By monitoring the model performance in production, Operation Engineers (OE) can re-initiate the loop.

By running the three phases sequentially, MLOps evolves the ML system continuously and infinitely.

Kreuzberger et al. [20] studies the principles, components, roles, and architecture of MLOps. The roles for the traditional ML systems are described. Hewage et al. [21] compares data versioning among MLOps tools. Garg et al. [22] introduces an application that deploys ML models using CI/CD. But monitoring the effectiveness of ML model is still unaddressed. Granlund et al. [18] presents two real-world cases of creating multiple-organization ML systems. It concludes that data related operations can be complicated as data can't be moved across the boundaries. Paleyes et al. [8] addresses the challenges in deploying machine learning models in production systems. It describes machine learning deployment workflow with potential issues and concerns.

C. FLOps

FL focuses on distributed data as well as organizations, and develops the model in a federated way, which makes the FL

[2]https://ml-ops.org/

systems more complicated than the traditional ML systems. We define the concept of FLOps:

FLOps combines a group of processes, technologies, and tools to improve the efficiency and quality of developing and deploying cross-silo federated learning systems. FLOps is a cross-discipline software development methodology.

III. Challenges

Here we discuss a scenario where five banks will build a Credit-Card Risk Detection System(CCRD) using Federated Learning technology. Banks are located in different countries. Each bank possesses its customers' private data which is not accessible by other banks. Each bank has its own Business Analyst(BA) Team, Data Scientist(DS) Team, Machine Learning Engineer(MLE) Team, Software Engineer(SE) Team, and Operation Engineer(OE) Team, regardless of small or big. Five banks agree to (1) use all customer data without transferring it to other banks, and (2) train a shared AI model which can be deployed in each bank's production system for risk prediction.

FLOps is to tackle the following challenges for developing cross-silo FL systems.

A. Cross-Entity

1) Lack of connection: The entities in the federated learning paradigm (i.e. the banks in the example) are geographically separated, which means BA teams, DS teams and MLE teams are all distributed. Teams lack a strong association in such a federated learning system project.

2) Lack of leadership: There is a lack of influential leaders or entities that can make decisions on not only enterprise-grade architecture, data access protocol, data abstraction agreement, ML model architecture, and ML model parameters, but also infrastructures, cloud services, and databases.

3) Inconsistency of capability: The maturity and capability of the counterpart teams among the entities is inconsistent. The teams may use different skill sets and technology stack, such as database types, programming languages, and ML libraries, which even pushes the gap bigger.

B. Protected data and shared model

1) Diversity of raw data: Each entity has its own data which is not accessible by other entities. Data structure and features may vary among entities [5]. Data diversity is quite unique in a FL system but not in a traditional machine learning system where the data is central-collected.

2) Matching data with model: The global model is shared among all entities. Once the model design is finalised, the client entities take the responsibility of assuring their own data to match the model input structure. Due to both data diversity and model consistency among all entities, transforming data to fit the model is consequent in such a FL setting.

3) Responsibility of model design: The activity of the global model drafting and model designing is usually performed by one entity on behalf of all entities, then the model is reviewed, verified, and agreed by other entities before it is dispatched. Designed by one and used by all is usually true in cross-silo FL system. The design philosophy should be well stated by the designer and completely obeyed by other entities.

C. Dual-factor Versioning

1) Version of model: To acquire the traceability, model versioning is a necessary practice for a FL global model. When the global model parameter changes by running additional iterations of model training and model aggregating but model architecture or model input structure keeps unchanged, a new version of the global model is then aggregated by the server.

2) Version of data transformation: Data interface adaption (DIA) defines what and how the features will be taken into account. Once the model architecture or/and model input structure is changed, the data interface adaption may consequentially change, which then causes ML model to be retrained from scratch again. All data engineering in entities might need re-work due to such a series of changes.

IV. Lifecycle

Developing and deploying a federated learning system for a cross-silo setting is not a one-off task or a simple job. It takes multiple stages continuously and consecutively.

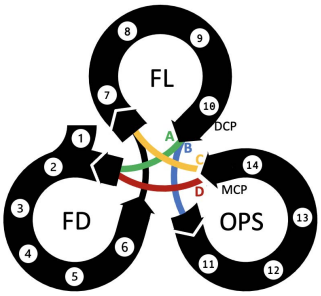

Fig. 3. Loops of FLOps with 14 activities, 3 phases, and 2 check points

A. Activities

We define 14 activities to describe the lifecycle of FLOps shown in Fig. 3.

1) Business Decision: A business decision is the initial driver of the FL system project. It is usually made by a Business Analyst who analyses system requirements. The entity where this BA works at plays a leader's role in the project. The lead entity may call for discussions, workshops and retrospectives to make decision on the project.

2) System Architecture Design: The lead entity takes responsibility of system architecture design, including the server and clients. In this step, the communication protocol between the server and the clients is defined, reviewed, and eventually published to all entities. The infrastructures that run the server and the client programs should also be considered in this step.

3) Joint Data Analyzing: Since different entities might have various data or data types, a cross-silo data scientist team, led by a chief data scientist, analyses each entity's data, and decides which features will be used for the training ML model.

4) Data Interface Abstracting: Defined by the data scientist team, Data Interface Abstraction (DIA) describes a unified form for model training or production purpose. DIA allows the single global model to adapt to various raw data owned by all entities. DIA has a version number which affects the outcome of the following steps.

5) Data Engineering: According to data interface abstraction in Activity 4, the raw data in each entity is extracted, transformed, and prepared for model training.

6) Global Model Design: The ML engineer team then designs model architecture, selects algorithms, fine-tunes hyperparameters, and ensures that the model performs well on the testing data or synthetic data. If the model performance is not satisfied, or the model is not converged correctly, *Activity 2, 3, 4, 5* could be repeated.

7) Global Model Dispatching: Once the model design is finished, it is then dispatched to each client so that both server and clients have the same model design.

8) Local Training: Local training is triggered by the server and occurs on the clients.

9) Local Model Upload: The clients then upload the local model update to the server.

10) Global Model Aggregation: The server aggregates the local model updates, then generates, evaluates, and tests the new global model. If the new global model's performance is not satisfied, the cycle will be led to *Activity 2*.

11) Global Model Release: The server publishes the global model. The artifact is marked with a version number of the global model and a version number of the Data Interface Abstraction mentioned in *Activity 4*.

12) Global Model Deployment: The operation engineer copies the model artifacts to production manually or using CI/CD tools, which depends on how the production system integrates the ML model.

13) Model Operations: The operation engineer maintains the system and infrastructure of the production system.

14) Model Monitoring: Evaluating the model performance on production data automates the whole process. Once the performance result gets lower than the predefined metrics, MLE and DS teams could work together to determine whether the model should be re-designed. The flow may fork to *Activity 2* or *Activity 7*.

B. Phases

FLOps has three phases in the whole lifecycle. Each phase is self-iterative and has at least one exit point to the next phase.

1) Federated Designed(FD): This phase covers not only system architecture design and ML model design but also the data interface design which is unique and critical in FL system. All entities in the FL settings are supposed to participate and contribute to system architecture review, data analysis, data interface verification, and model design verification.

2) Federated Learning (FL): The typical distributed federated learning activities occur in this phase. The global model is dispatched to the clients, then the clients train the local model and upload the model updates. The aggregation algorithm is run on the server to generate a new global model. The new global model will be evaluated before release. There are three exit points at the end of this phase: iterating itself, going back to FD phase, and going forward to OPS phase. This evaluation point is Design Check Point (DCP) shown in Fig. 3.

3) Operations (OPS): The global model version and the data interface adaption version are available for production purpose. Model monitoring including a performance assessment is to determine automatically whether the flow goes to FD phase or FL phase. This evaluation point is Model Check Point (MCP) shown in Fig. 3.

C. Loops

Starting from *Activity 1* in Fig. 3, three phases can be connected to make up of four types of loop combinations:

1) FD-FL-FD: Suppose the global model's evaluation doesn't meet pre-defined performance metrics, and the system architecture and the data interface adaption may require to be re-designed. In this case the flow can be redirected to the FD phase from *Activity 10* via the *Arc A* in Fig. 3.

2) FD-FL-OPS: Once the performance of the global model is satisfied for production, the model is released, deployed, and monitored.

3) FL-OPS-FD: If the production reports any low-performed alerts, and DSs and MLEs believe that Federated Design (FD) should be re-executed, then the cycle is moved to FD from *Activity 14* via *Arc D* in the Fig. 3.

4) FL-OPS-FL: In the case that the production performs low, but the system architecture and data interface abstraction are still workable and promising, then the flow goes to FL phase via *Arc C* to train the model on the client's data repeatedly.

V. Approaches

Continuously developing a federated learning system in a cross-silo setting is still a gap-filling job. The answers to How could the FLOps get started and run smoothly and automatically are not well-studied recently. FLOps can adopt the best practices of DevOps/MLOps. Besides, we propose more approaches from two perspectives.

A. Organisational Architecture Perspective

From the perspective of management, FLOps needs intensive communication and collaboration among entities. A collaborative culture should be built up in a well-organized organizational architecture. The following approaches can be project-wise for a specific FL system project development or long-term for comprehensive cooperation. Fig. 4 is an example in practice.

1) The FL Presidency:

The FL presidency in the FL setting is such an entity who is elected by all the entities and in charge of all aspects of development and deployment, such as task allocation, schedule management, risk management, and quality management. The FL presidency propels the cross-silo FL system development and deployment. The FL presidency is an organiser and

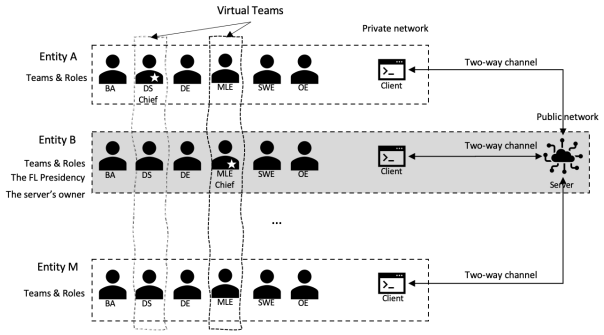

Fig. 4. A sample diagram of Organisational Architecture where Entity B is the FL Presidency and owns the server.

coordinator who plays a lead role. It calls for meetings, discussions, workshops, technical reviews, joint debugging, retrospective seminars, and concludes the entities. The FL presidency could be the initiator who raises the business requirements. It completely understands the pain points that the entities are suffering and thoroughly knows the industry where the business is running. It is passionately keen on an entire solution to deal with the pain points by working closely with other entities.

2) Ownership of the server:

FL systems have two logical roles: the server and the clients. The clients are running with no doubt in private environments that the entities own privately. The server is, however, centralized, neutral, and accessible from the clients.

The server's owner possesses the physical server or cloud infrastructures and is responsible for maintaining and managing the server. Clearly announcing the affiliation of the server ensures the server's availability, reliability, and safety. The FL presidency is a potential candidate, but any of the entities can be elected to become the server's owner. Some federated learning applications employ a server-less design or a decentralized architecture [6] and don't need a centralized server.

3) Cross-silo Dynamic Virtual Teams:

Each entity has its own functional teams, such as Business Analysts (BA), Data Scientists (DS), Data Engineers (DE), Machine Learning Engineers (MLE), Software Engineers (SE), and Operation Engineers (OE). It is essential to have teams who function the same to work closely together.

Virtual teams can be built up once necessary with the loop running according to the lifecycle shown in Fig. 3. For example, during the Federated Design phase, Data Scientists, Data Engineers, ML Engineers could have their dynamic virtual teams to analyse data, design data interface adaption, design model architecture, etc. Virtual teams are light-weighted teams who speak the same functional language and communicate efficiently. Establishing virtual teams is on demand. It is unnecessary to have all virtual teams running all the time. Virtual teams take another essential responsibility which is skill training and alignment.

4) Chief Roles:

Each virtual team has its chief role or position. For example, A virtual Data Scientist team is led by Chief Data Scientist (CDS), and the CDS comes from one of the entities and works with all data scientists remotely, dynamically, and virtually. Similarly, there are Chief BA, Chief DE, Chief MLE, etc., in different phases of the lifecycle.

The Chief Role plays the lead role in the virtual team. The primary responsibility of the chief position is to facilitate cross-organisational collaborations, ensure the correctness of the decision made by the functional teams, and align the teams' skill sets.

B. Workflow Perspective

From a more practical perspective, we propose the following pragmatic approaches that help FLOps to be managed continuously.

1) Metadata Engineering:

Metadata engineering is a series of practices aiming to abstract standard features from multiple data sources among various entities and define a unified input data form for the global model in FL settings.

Metadata engineering is carried out by the virtual teams of data scientists (DS) and data engineers (DE). Metadata engineering standardises the connection between model and data and unifies data preparation in the server and clients. The output of metadata engineering is Data Interface Adaption (DIA) which is also the definition of the model input structure. DIA is versioned at each evolution of metadata engineering. All client entities in the FL settings can retrieve the DIA from the server and prepare the data for the global model.

2) Daul Deployment:

Naming and versioning both the global model and Data interface adaption (DIA) is the fundamental capability of a FL system. The global model and DIA are traceable respectively, and associated with each other. Each version of global model links to a single version of DIA, while each version of DIA may have multiple versions of global models.

The server is supposed to store and manage the global models and the data interface adaptions (DIA) with published version numbers. By supporting dual deployment, the clients can be aware of the change in the global model and the DIA and carry out any necessary data preparation and model adaption.

3) Check Points:

The three phases in Fig. 3 are not always included in one cycle to improve efficiency. We set two check points in FLOps: Design Check Point (DCP) and Model Check Point (MCP).

DCP is the switch that routes the flow back to the FD phase. The model design finished in Phase FD is verified by all clients using private data and the server aggregating global model. DCP will decide whether the model design meets the performance requirement. An inappropriate model design followed by incorrect data preparation will lead to bad performance; therefore, Phase FD will need to be re-executed.

MCP is the place where model performance is measured. The under-performed result may blame two causes, (1) model design and data preparation and (2) model training iterations. The flow gets redirected to Phase FD and Phase FL.

VI. Conclusion

The major characteristics of a cross-silo federated learning system are cross entity and isolated raw data.

The nature of cross-entity weakens the association between functional teams among the entities, causes the hardness of collaboration and communication, and leads to inefficient decision making and project failure eventually. The management actions described in Section V improve the efficiency of collaboration and clarify the responsibility of individuals, teams, and entities in a FL setting.

The fact of isolated raw data breaks the uniformity of the global model's input structure. It consequentially results in the disconnection between the model designing and the model training. The difference can be made up by metadata engineering which abstracts data interface for all entities. Finally, check points are taking in place to re-route the cycle of FLOps at any necessity.

A. Future Research Directions

Developing and deploying cross-silo federated learning systems is a comprehensive systematic topic. It regards to not only cost, efficiency, and quality, but also management, technology, tools, and practice. Although the high-level guidelines of approaches are given in this paper, the detailed methodologies and practical solutions in FLOps for cross-silo federated learning systems can be studied further. We describe the following directions for future research on FLOps.

1) Automation: Automation of FLOps is one of the potential research topics aiming to connect lifecycle activities smoothly and seamlessly. CI/CD [10] can be re-used in FLOps but need alteration. Pipeline [10] [19] is the useful idea for automation, but there are some human-interactions in cross-silo FL systems. For instance, *Activity 4* of the lifecycle is hard to automatize if there is not a well-designed tool or method.

2) Adaption: Adaptive data engineering in FL system is critical as different data is supposed to be pumped into the same global model via the model input structure definition. Adaptive data engineering in FL is also the key technology of FLaaS [4] where FL system is re-used as a service to connect multiple data providers.

3) Metrics: Metrics system is another key point for automating FLOps continuously. A good metrics system can be integrated into the FLOps pipeline and automatically trigger the phase switching [16]. The lack of metrics is one of the obvious obstacles to FLOps automation.

4) Practical System Architecture Pattern: By using different taxonomy, federated learning systems can be classified into Cross-device FL [1], Cross-silo FL [2] [7], synchronous FL [1], asynchronous FL [23], Server-centralized FL [1], Server-free FL [6], and clustered FL [9]. Various system architecture patterns can be designed for each FL system to simplify the development.

References

[1] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. arXiv preprint arXiv:1811.03604, 2018.

[2] Chao Huang, Jianwei Huang, and Xin Liu. Cross-silo federated learning: Challenges and opportunities. arXiv preprint arXiv:2206.12949, 2022.

[3] Ahmed Imteaj, Urmish Thakker, Shiqiang Wang, Jian Li, and M Hadi Amini. A survey on federated learning for resource-constrained iot devices. IEEE Internet of Things Journal, 9(1):1–24, 2021.

[4] Nicolas Kourtellis, Kleomenis Katevas, and Diego Perino. Flaas: Federated learning as a service. In Proceedings of the 1st workshop on distributed machine learning, pages 7–13, 2020.

[5] 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.

[6] Tao Sun, Dongsheng Li, and Bao Wang. Decentralized federated averaging. arXiv preprint arXiv:2104.11375, 2021.

[7] Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. Personalized cross-silo federated learning on non-iid data. In AAAI, pages 7865–7873, 2021.

[8] Andrei Paleyes, Raoul-Gabriel Urma, and Neil D Lawrence. Challenges in deploying machine learning: a survey of case studies. ACM Computing Surveys (CSUR), 2020.

[9] Jie Ma, Guodong Long, Tianyi Zhou, Jing Jiang, and Chengqi Zhang. On the Convergence of Clustered Federated Learning. arXiv preprint arXiv:2202.06187, 2022.

[10] Liming Zhu, Len Bass, and George Champlin-Scharff. Devops and its practices. IEEE Software, 33(3):32–34, 2016.

[11] [11] G Bou Ghantous and Asif Gill. Devops: Concepts, practices, tools, benefits and challenges. PACIS2017, 2017.

[12] Leonardo Leite, Carla Rocha, Fabio Kon, Dejan Milojicic, and Paulo Meirelles. A survey of devops concepts and challenges. ACM Computing Surveys (CSUR), 52(6):1–35, 2019.

[13] Ioannis Karamitsos, Saeed Albarhami, and Charalampos Apostolopoulos. Applying devops practices of continuous automation for machine learning. Information, 11(7):363, 2020.

[14] Elisa Diel, Sabrina Marczak, and Daniela S Cruzes. Communication challenges and strategies in distributed devops. In 2016 IEEE 11th International Conference on Global Software Engineering (ICGSE), pages 24–28. IEEE, 2016.

[15] Yehia Ibrahim Alzoubi, Asif Qumer Gill, and Ahmed Alani. Empirical studies of geographically distributed agile development communication challenges: A systematic review. Information & Management, 53(1):22–37, 2016.

[16] Nicole Forsgren and Mik Kersten. Devops metrics. Communications of the ACM, 61(4):44–48, 2018.

[17] Cor-Paul Bezemer, Simon Eismann, Vincenzo Ferme, Johannes Grohmann, Robert Heinrich, Pooyan Jamshidi, Weiyi Shang, André van Hoorn, Monica Villavicencio, Jürgen Walter, et al. How is performance addressed in devops? In Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering, pages 45–50, 2019.

[18] Tuomas Granlund, Aleksi Kopponen, Vlad Stirbu, Lalli Myllyaho, and Tommi Mikkonen. Mlops challenges in multi-organization setup: Experiences from two real- world cases. 2021 IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN), May 2021.

[19] Georgios Symeonidis, Evangelos Nerantzis, Apostolos Kazakis, and George A Papakostas. Mlops-definitions, tools and challenges. In 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), pages 0453–0460. IEEE, 2022.

[20] Dominik Kreuzberger, Niklas Kühl, and Sebastian Hirschl. Machine learning operations (mlops): Overview, definition, and architecture. arXiv preprint arXiv:2205.02302. 2022.

[21] Guodong Long, Yue Tan, Jing Jiang, and Chengqi Zhang. Federated Learning for Open Banking. arXiv preprint arXiv:2108.10749, 2021.

[22] Satvik Garg, Pradyumn Pundir, Geetanjali Rathee, P. K. Gupta, Somya Garg, and Saransh Ahlawat. On continuous integration / continuous delivery for automated deployment of machine learning models using mlops, 2022.

[23] Zheng Chai, Yujing Chen, Liang Zhao, Yue Cheng, and Huzefa Rangwala. Fedat: A communication-efficient federated learning method with asynchronous tiers under non-iid data. ArXivorg, 2020.