On the Adoption of Federated Machine Learning: Roles, Activities and Process Life Cycle

Tobias Müller[1,2][a], Milena Zahn[1,2] and Florian Matthes[1]
[1] *Technical University of Munich, Germany; TUM School of Computation, Information and Technology, Department of Computer Science, Chair of Software Engineering for Business Information Systems*
[2] *SAP SE, Dietmar-Hopp-Allee 16, 69190 Walldorf, Germany*
{*tobias1.mueller, milena.zahn, matthes*}*@tum.de*

Keywords: Federated Machine Learning (FedML), Software Engineering, Machine Learning Operations (MLOps)

Abstract: Federated Machine Learning is a promising approach for training machine learning models on decentralized data without the need for data centralization. Through a model-to-data approach, Federated Machine Learning yields huge potential from privacy by design to heavily reducing communication costs and offline usage. However, the implementation and management of Federated Machine Learning projects can be challenging, as it involves coordinating multiple parties across different stages of the project life cycle. We observed that Federated Machine Learning is missing clarity over the individual involved roles including their activities, interactions, dependencies, and responsibilities which are needed to establish governance and help practitioners operationalize Federated Machine Learning projects. We argue that a process model, which is closely aligned with established MLOps principles can provide this clarification. In this position paper, we make a case for the necessity of a role model to structure distinct roles, an activity model to understand the involvement and operations of each role, and an artifact model to demonstrate how artifacts are used and structured. Additionally, we argue, that a process model is needed to capture the dependencies and interactions between the roles, activities, and artifacts across the different stages of the life cycle. Furthermore, we describe our research approach and the current status of our ongoing research toward this goal. We believe that our proposed process model will provide a foundation for the governance of Federated Machine Learning projects, and enable practitioners to leverage the benefits of decentralized data computation.

1 PROBLEM STATEMENT

The increasing reliance on data-driven decision-making has led to an expansion of machine learning (ML) applications in various industries. However, the use of ML often requires large amounts of data, which is often collected from various sources including personal devices with sensors such as tablets, phones or other IoT devices. While this data could be valuable for training ML models to power intelligent services, it also contains sensitive personal information which must be protected. In traditional ML approaches, data is often centralized in a single data lake, which can be a privacy risk as it requires data transfer and allows a potential misuse or breach of sensitive information.

To tackle these issues McMahan et al. (2016) introduced Federated Machine Learning (FedML), a novel decentralized ML paradigm. The proposed algorithm follows a model-to-data approach, which enables training a common ML model on distributed data while the data remains on the user's devices and therefore implements data privacy by design. The FedML process is described in more detail in chapter 2. This approach yields a multitude of benefits in addition to data privacy. For example, the model can be used offline without the need of communicating with a server and the communication load is heavily reduced since only update gradients are shared between the parties. Despite these potential advantages, there are currently only a few production-level applications and most work on FedML still comprises prototypes or simulations (Lo et al., 2021).

Integrating ML systems into production-level applications requires the operationalization and incorporation of software development practices of ML systems. This is a highly challenging task since the addition of ML capabilities adds further complexity to the system design and implementation

[a] https://orcid.org/0000-0002-9088-5054

process (Serban et al., 2020; Kreuzberger et al., 2022; Wan et al., 2021). Especially the data collection, data preparation, training, monitoring as well as the deployment process of a typical ML life cycle require novel software engineering practices in comparison to traditional software engineering (Sculley et al., 2015; Serban et al., 2020). There is a persisting discrepancy between the engineering of ML-capable systems and the engineering of traditional software (Giray, 2021). To bridge this gap, the traditional software engineering ways of implementing code need to be revisited due to the uniqueness of non-deterministic ML systems.

To address these issues, a two-day long discussion of 160 practitioners and researchers on the challenges and implications of engineering ML systems at the *First Symposium on Software Engineering for Machine Learning Applications* (SEMLA) (Khomh et al., 2018) spawned two key questions:

- ”How should software development teams integrate the AI model life cycle (training, testing, deploying, evolving, and so on) into their software process?”
- ”What new roles, artifacts, and activities come into play, and how do they tie into existing agile or DevOps processes?”

According to SEMLA, researching these two key questions is essential to link software engineering and ML development processes. Currently, a growing corpus of academic literature is concerned with these problems. Chandrasekaran et al. (2021), for example, propose in their work on ML governance that an operational life cycle consists of data preparation, model development, and a model deployment phase. Accordingly, they defined principals, the involvement and interaction of these principals, and the life cycle management of ML systems (Chandrasekaran et al., 2021). Furthermore, Ritz et al. (2022) defined a comprehensive process model for ML systems to capture the dependencies between the artifacts and activities in a ML life cycle in order to bridge the gap between existing software engineering process models and ML-specific procedures (Ritz et al., 2022). Also, there has been a growing interest in defining principles, components, roles, and architectures in the context of operationalizing MLOps workflows (Ruf et al., 2021; Subramanya et al., 2022; Kreuzberger et al., 2022).

Even though the key questions posed by SEMLA have been addressed by the academic literature, FedML introduces new processes and requirements due to its decentral model-to-data approach. Due to the decentralization and local training/usage procedure, additional roles, activities, artifacts, and life cycle stages need to be introduced. Before FedML can be broadly operationalized, we argue that these specific questions posed by SEMLA need to be answered as well.

IEEE published a *Guide for Architectural Framework and Application of Federated Machine Learning* (IEEE, 2021), which defines a standard for the FedML reference architecture including user role descriptions of the FedML process. According to this IEEE standard, a participant could play the role of a data owner, model user, coordinator, and/or auditor. These roles are presented with their accompanying actions in the FedML process. This reference architecture provides generalized information as a template solution for the implementation of FedML processes including structures and respective elements with their relation. The reference architecture can be used as a basic foundation for the governance of FedML projects. However, according to this reference architecture, a single role is responsible for multiple steps of a typical MLOps life cycle. For instance, the activities of a coordinator comprise the aggregation step, model management, data management, deployment, and capabilities coordination.

We argue that the separation of roles and activities should be closely aligned with established MLOps stages and principles, such that the FedML specifics can be easily integrated into known MLOps workflows. Hence, a more structured breakdown of the activities and roles in relation to the different stages of an End-to-End FedML life cycle is needed to fully capture the dependencies and interactions between these roles. Furthermore, defining the set of actors, their roles and activities not only provides a clearer understanding of the project’s setup, but also plays a crucial part in the governance of the project (de Man and Luvison, 2019; Kujala et al., 2021; Caridà et al., 2018). To accomplish this, we aim to identify the differences between MLOps and FedML-specifics to finally derive a formal process model for a full End-to-End FedML life cycle. Through a holistic process model, we hope to enable practitioners to set up and provide a foundation for the governance of FedML projects.

In summary, the planned final contributions of our ongoing research would comprise:

- **Role Model**: To structure the individual roles including their corresponding capabilities and responsibilities.
- **Activity Model**: To understand the involvement, operations, and activities of each role.

- **Artifact Model**: To show how artifacts are used and structured in the process flow.
- **Process Model**: To capture the different life cycle stages including the interactions and dependencies between the activities, roles, and artifacts.

To achieve this goal, we follow the Design Science Research (DSR) conceptual framework by Hevner et al. (2004) through an incorporated DSR Methodology (DSRM) by Peffers et al. (2007) . The problem identification and objective description have been conducted through a focus group with three different product teams and a total of ten participants. The relevance and objectives have been confirmed through five semi-structured expert interviews. The knowledge base was built through a literature review. In an iterative manner, we developed the first versions of our models and evaluated the artifacts with a focus group every four weeks. A more detailed description of the conducted research methodology will be provided in the report of the final results.

In the following, we present our current state of research. In section 2, we will give a short introduction to the FedML process. This is followed by section 3 with a high-level overview of our state of research. This section comprises our current separation of roles including a description of the corresponding activities and the life cycle of a FedML project. Finally, we conclude with a discussion.

2 PRELIMINARIES

FedML is a novel ML paradigm that is concerned with training a joint ML model on distributed datasets over multiple iterations. In traditional ML settings, data is usually assembled on a central data lake, where the ML model is trained. Hence, data owners are obliged to share their data with a central server and therefore lose data sovereignty, which potentially poses a privacy risk. Introduced by McMahan et al. (2016), FedML counteracts this need of sharing datasets. Through a model-to-data approach, FedML enables K data owners to train a joint model $\mathcal{M}_{FED}$ iteratively without the need of disclosing their disjoint datasets $\{\mathcal{D}_{i=1}^{K}\}$. A simplified illustration of the FedML process can be seen in Figure 1.

More specifically, the FedML protocol can be divided into four distinct steps:

1. The server chooses an initial global model $\mathcal{M}$ suitable for the use case and underlying data structure. The global model $\mathcal{M}$ can be initially trained on the dataset of the server.

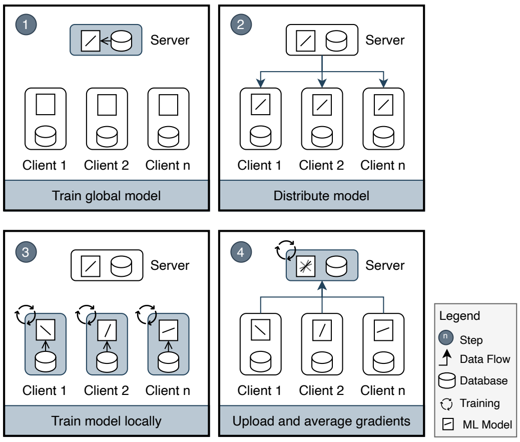

Figure 1: One iteration of the FedML process.

2. The server distributes the global model $\mathcal{M}$ amongst all clients.
3. Each client k trains the global model on its own dataset and stores the update gradient $g_k = \nabla F_k(w_t)$. After the training process, each client k owns its individually adapted ML model $\mathcal{M}_k$ based on its dataset $\mathcal{D}_k$.
4. The clients send the individually computed update gradients back to the server. The gradients are aggregated based on a pre-defined protocol and used to update the global model.

This process can be repeated over several iterations until the global model reaches a certain accuracy level or the accuracy converges. In the original *FedAVG* implementation (McMahan et al., 2016), the model is learned through stochastic gradient descent (SGD) and the aggregation scheme is as follows:

$$\begin{aligned} w_{t+1} &\leftarrow w_t - \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^{k} \\ w_{t+1}^{k} &\leftarrow w_t^k - \eta g_k, \forall k \end{aligned} \tag{1}$$

FedML commonly uses a client-server architecture consisting of a central orchestrating server and multiple clients. However, completely decentralized alternatives have been proposed, which establish a peer-to-peer network to exchange model updates without a central server (Roy et al., 2019). In this work, we solely focus on traditional client-server FedML, since fully decentral architectures are rarely used.

The distribution of features and samples across datasets may not be homogeneous. If all datasets contain different samples but share the same feature space we refer to *horizontal federated learning (HFL)*. If the same samples are present in all datasets but the feature spaces are disjoint, we refer to *vertical federated learning (VFL)* (Yin et al., 2021).

The collaborative nature of FedML implies a more sophisticated training and usage process compared to traditional, centralized ML. This may result in new roles, activities, artifacts, and processes, which need to be investigated and addressed.

3 STRUCTURING FEDERATED MACHINE LEARNING PROJECTS

To provide a clearer understanding of the process structure, we first need to analyze the different activities which are needed to implement and execute the FedML processes as well as the MLOps workflows. These activities can be grouped and assigned to different role definitions. Through examining the resulting artifacts, interactions and dependencies, the descriptions of the individual roles can be complemented with their corresponding capabilities and responsibilities. With regard to the process model, it is first needed to structure the project flow into different life cycle stages. By combining the roles, activities, artifacts, interactions and life cycle we can finally derive a holistic process model. By this, we aim to enable practitioners to set up FedML projects as well as provide a foundation for its governance.

The following presents our current proposal of the involved roles with their corresponding activities (section 3.1) and the different stages of the project (section 3.2). It is important to note, that the following descriptions are a high-level overview of our current state of work and may differ from the final results.

3.1 Participating Roles and Activities

Apart from the high-level descriptions of the IEEE standard (IEEE, 2021), no fixed roles for participants within FedML projects have been uniformly defined yet. We reviewed academic literature on ML Governance as well as reference architectures and identified different role definitions of a traditional ML life cycle. Furthermore, we performed a thorough analysis of a FedML project flow and derived critical roles from the technology specifics. Finally, we aggregated and combined our findings which resulted in a set of participating roles of the FedML process. The following describes the identified roles including their responsibilities and capabilities. It is essential to distinguish between participant and role. One participant can take on several roles, and one role can be assigned to several participants.

- **Model Manager:** The model manager has the business need of a problem to be solved with ML and represents the initiator. He takes responsibility for the context and requirements specification. The customer requirements are communicated to the model owner and deployment details to the model deployer. The primary capability is domain and business knowledge.
- **Model Owner:** The model owner holds ownership of the global ML model and is authorized to decide on technical requirements. He gets customer requirements from the model manager and communicates technical requirements to the model builder.
- **Model Builder:** The model builder defines the ML model architecture and is responsible for technical and system specifications. He acts as a mediator between the model owner, data owner, and model deployer. The primary capability is FedML expertise. He specifies and coordinates the training procedure, trains the initial model, distributes the model, aggregates gradients, evaluates the model, and provides the trained model to the model deployer.
- **Model Deployer:** The model deployer serves the ML model according to the deployment details defined by the model manager. The primary capability is deployment knowledge and infrastructure to serve the ML model.
- **Data Owner:** The data owner collects data, cleans, transforms, trains the model, and sends gradient updates to the model builder. The primary capability is a sufficient database with qualitative data to train the ML model and facilitate the FedML infrastructure.
- **End User:** The end user consumes the deployed ML model to solve a particular problem.
- **Auditor:** The auditor verifies the deployment of the ML model, adheres to technical standards, and adheres to compliance obligations.
- **Adversary:** The adversary is an entity trying to disrupt, intercept, or cause harm to the system.

The roles can be grouped according to their tasks in the FedML procedure. The Model Manager and the Model Owner are mainly responsible for the management and administration of the project. The Model Builder and the Model Deployer are carrying out the implementation and deployment of the FedML model. The Data Owner undertakes the task of data collection and local training of the model. The End User, the Auditor, and the Adversary assume a role that is not directly posted in the FedML training process. Figure 2 visualizes the grouping and interac-

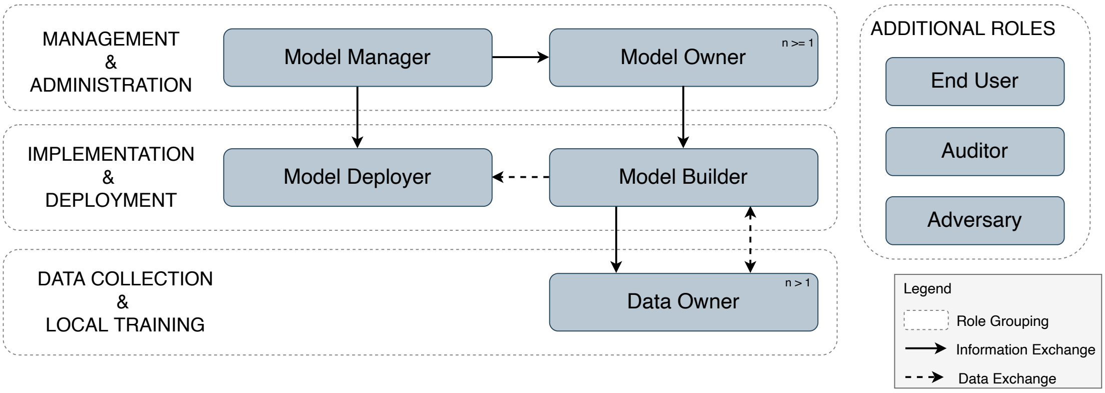

Figure 2: Overview of the roles involved in the FedML process

tions between the different roles on a high level abstraction.

3.2 Life Cycle of a Federated Machine Learning Project

To capture the involvement, interactions, and activities of each role as well as the arising dependencies, it is necessary to provide a clearly structured project process first. The decentral nature of the FedML training process allows for collaborations of different data owners. Hence, if we want to propose a holistic life cycle we need to allow and depict collaborative settings as well. According to Diirr and Cappelli (2019) collaborative projects can be divided into five main stages, from the creation of the collaboration, operational execution, and evaluation leading to evolution or dissolution. To adjust it to the FedML technology, we merged stages and included steps of operational ML activities, as well as FedML-specifics. Note, that the collaboration formation step is optional and only has to be considered in collaborative settings.

Fig. 3 shows an overview of the derived inter-organizational FedML life cycle. The illustration underlies simplifications to ease the readability of the visualized process.

The first stage, the **collaboration creation**, comprises four sub-steps: *identification*, *formation*, *viability*, and *planning* (Diirr and Cappelli, 2019). First, the business need and the resulting collaborative business opportunity are *identified* and characterized. In addition, the strategy for inter-organizational collaboration of the participants should be determined. Next, the *formation* of the network of organizations starts by identifying, evaluating, and selecting participants. After that, the *viability* of the FedML project must be assessed with the selected participants in terms of technical and legal aspects, e.g., GDPR and antitrust. Subsequently, the business project can be specified in the *planning*, and the collaboration can be institutionalized and contracts negotiated.

During this stage, iteration loops may arise, or the project may be cancelled, e.g., if the feasibility study fails.

The **FedML development** is the second stage and includes the iterative process of the FedML *setup*, *training process*, and *testing and evaluation* of the model. The *setup* compromises the installation of the infrastructure with all participants to execute the iterative FedML steps, which are performed during the *training process* (see Fig.1). Once the aggregated FedML model has been successfully *tested and evaluated* with defined quality specifications, it moves to the next stage.

The stage **FedML operation** describes the *deployment*, *monitoring*, and *operation* of the global FedML model. The model is served to the end customer by *deploying* the model into a specified environment and putting it into production to be used for its purpose. The *monitoring* includes monitoring the FedML model's performance, selecting feedback, and detecting problems as soon as possible. The *operation* comprises the application of processes to maintain the models in production environments.

The next stage, **evaluation**, assesses the FedML model and the collaboration itself to decide how to proceed with the project. The *evaluation* based on evaluation criteria should lead to decisions on how to proceed. In case of a *retirement*, the collaboration will be closed, and it must be decided whether the resulting FedML models remain to be distributed or recalled. In the other case, an evolution means either improvement or changes in the FedML development or adaptation of the collaboration of the organizations.

The simplified life cycle does not represent all it-

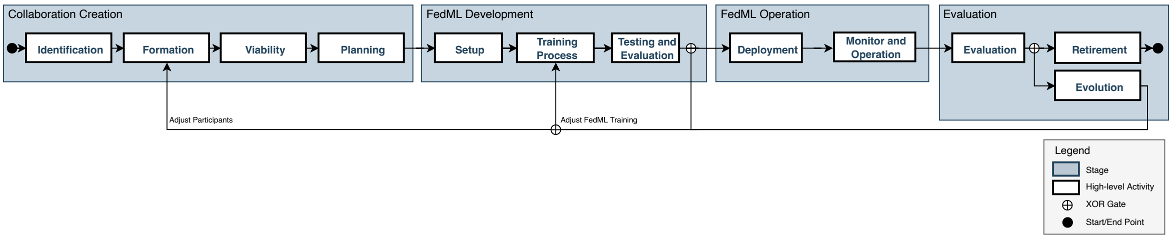

Figure 3: High-Level Overview of a FedML life cycle with exemplary iterative loops

erative possibilities and provides a rough guideline for FedML projects.

4 DISCUSSION

Despite the potentially large benefits of breaking up data silos, implementing privacy by design, and heavily reducing the communication costs of ML applications, FedML only has seen a few production-level applications and seems to be mainly in a prototype stage. We observed that FedML is missing a structured process model which gives the involved parties clarity over the structure, interactions, and responsibilities of the full FedML life cycle process. We argue that this process model should be closely aligned with established MLOps stages and principles such that the FedML specifics can be easily integrated into known workflows and therefore ease the operationalization. To provide this guidance, it is important to define role descriptions including their activities, capabilities, and responsibilities. Furthermore, the resulting artifacts throughout the life cycle need to be derived. By combining the interactions and dependencies between the roles, activities, and artifacts we can provide a formal process model of an end-to-end FedML life cycle. We argue that such a process model is needed to provide a blueprint for practitioners to establish governance and integrate FedML into their products. This position paper represents our current state of research towards a process model, which clearly structures the FedML life cycle with regard to established MLOps practices. As a first step towards this goal, we proposed an initial structuring of roles and activities which are involved in the FedML training process as well as a high-level project life cycle. Our current work focuses on capturing the artifacts, interactions, and dependencies while iteratively integrating our findings into the role and activity models. We finally aim to derive a holistic FedML process model over the full project life cycle.

ACKNOWLEDGEMENTS

The authors would like to thank SAP SE for supporting this work.

REFERENCES

Caridà, A., Colurcio, M., and Melia, M. (2018). Designing a collaborative business model for smes. *Sinergie Italian Journal of Management*, pages 233–253.

Chandrasekaran, V., Jia, H., Thudi, A., Travers, A., Yaghini, M., and Papernot, N. (2021). Sok: Machine learning governance. *ArXiv*, abs/2109.10870.

de Man, A.-P. and Luvison, D. (2019). Collaborative business models: Aligning and operationalizing alliances. *Business Horizons*, 62(4):473–482.

Diirr, B. and Cappelli, C. (2019). A systematic literature review to understand cross-organizational relationship management and collaboration. In *Anais do XV Simpósio Brasileiro de Sistemas Colaborativos*, pages 118–119, Porto Alegre, RS, Brasil. SBC.

Giray, G. (2021). A software engineering perspective on engineering machine learning systems: State of the art and challenges. *Journal of Systems and Software*, 180:111031.

Hevner, A. R., March, S. T., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, 28(1):75–105.

IEEE (2021). Ieee guide for architectural framework and application of federated machine learning. *IEEE Std 3652.1-2020*, pages 1–69.

Khomh, F., Adams, B., Cheng, J., Fokaefs, M., and Antoniol, G. (2018). Software engineering for machine-learning applications: The road ahead. *IEEE Software*, 35(5):81–84.

Kreuzberger, D., Kühl, N., and Hirschl, S. (2022). Machine learning operations (mlops): Overview, definition, and architecture.

Kujala, J., Aaltonen, K., Gotcheva, N., and Lahdenperä, P. (2021). Dimensions of governance in interorganizational project networks. *International Journal of Managing Projects in Business*, 14(3):625–651. Funding Information: This paper extends on previous research that has been presented in European Academy of Management (EURAM) conference, 1-4 June 2016, Paris, France. Publisher 2020, Emerald Publishing Limited.

Lo, S. K., Lu, Q., Wang, C., Paik, H.-Y., and Zhu, L. (2021). A systematic literature review on federated machine learning: From a software engineering perspective. *ACM Comput. Surv.*, 54(5).

McMahan, H. B., Moore, E., Ramage, D., and y Arcas, B. A. (2016). Federated learning of deep networks using model averaging. *CoRR*, abs/1602.05629.

Peffers, K., Tuunanen, T., Rothenberger, M. A., and Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, 24(3):45–77.

Ritz, F., Phan, T., Sedlmeier, A., Altmann, P., Wieghardt, J., Schmid, R., Sauer, H., Klein, C., Linnhoff-Popien, C., and Gabor, T. (2022). Capturing dependencies within machine learning via a formal process model. In Margaria, T. and Steffen, B., editors, *Leveraging Applications of Formal Methods, Verification and Validation. Adaptation and Learning*, pages 249–265, Cham. Springer Nature Switzerland.

Roy, A. G., Siddiqui, S., Pölsterl, S., Navab, N., and Wachinger, C. (2019). Braintorrent: A peer-to-peer environment for decentralized federated learning.

Ruf, P., Madan, M., Reich, C., and Ould-Abdeslam, D. (2021). Demystifying mlops and presenting a recipe for the selection of open-source tools. *Applied Sciences*, 11(19).

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.-F., and Dennison, D. (2015). Hidden technical debt in machine learning systems. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Serban, A., van der Blom, K., Hoos, H., and Visser, J. (2020). Adoption and effects of software engineering best practices in machine learning. In *Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, ESEM '20, New York, NY, USA. Association for Computing Machinery.

Subramanya, R., Sierla, S., and Vyatkin, V. (2022). From devops to mlops: Overview and application to electricity market forecasting. *Applied Sciences*, 12(19).

Wan, Z., Xia, X., Lo, D., and Murphy, G. C. (2021). How does machine learning change software development practices? *IEEE Transactions on Software Engineering*, 47(9):1857–1871.

Yin, X., Zhu, Y., and Hu, J. (2021). A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. *ACM Comput. Surv.*, 54(6).