SCHOOL OF COMPUTATION,
INFORMATION AND TECHNOLOGY —
INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

FLOps: Practical Federated Learning via Automated Orchestration (on the Edge)

Alexander Malyuk

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

FLOps: Practical Federated Learning via Automated Orchestration (on the Edge)

TODO

Author:	Alexander Malyuk
Supervisor:	Prof. Dr-Ing. Jörg Ott
Advisor:	Dr. Nitinder Mohan, Giovanni Bartolomeo
Submission Date:	15.09.2024

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, 15.09.2024 Alexander Malyuk

Acknowledgments

Abstract

Kurzfassung

Contents

1 Introduction

The number of smart devices has been rapidly growing in the last several years. Improvements in connectivity (Cloud Computing & Internet of Things—IoT), connection speeds (5G), and computing power enable this increasing fleet of edge/mobile devices to generate enormous amounts of data (BigData). Combined with the expansion of AI/ML, this data is a driving factor for current successful workflows and future advancements. This complementing union of technologies plays a key role in elevating various domains, from agriculture and healthcare to education and the security sector, to Industry 4.0 and beyond (Industry 5.0). Diverse and complex challenges arise from this swiftly evolving landscape. [1]

1.1 Problem Statement

With great access to data comes great responsibility that can be easily exploited. Many of the aforementioned machines are personal user devices or belong to companies and organizations that handle customer or internal resources. These devices store and handle sensitive private data.

In classical (large-scale) Machine Learning, data gets sent from client devices to a centralized server, which usually resides in the cloud. The collected data is used on the server to train ML models or perform inference serving. This approach provides direct access to this sensitive data and the power to trace back its origin, creating a breach of privacy. [4]

Governments and organizations have established laws and regulations to prohibit potential abuse of sensitive data. Examples include the European Parliament regulation to protect personal data [6] or the California Consumer Privacy Act (CCPA) [3]. These measures aim to support cooperation between organizations and nations while protecting trade secrets. However, some laws and regulations prohibit sharing or moving data to other countries or even off-premises. [4]

Ignoring and no longer using this large amount of data would heavily limit current workflows and further developments for many data-dependent and data-hungry technologies. In 2017, a team of Google researchers introduced Federated Learning (FL) as one possible solution to utilize sensitive data while keeping it private. Instead of collecting the data on a server and training ML models centralized, in FL, the model training occurs directly on the client devices. Afterward, the individually trained models get sent to the server, which combines the collected models into a single shared one. This so-called global model can then be distributed to the clients again for further training cycles. Therefore, FL enables training a shared model on sensitive data while keeping that data secure on the local client devices. [5]

While many researchers are actively engaged in the field of FL, the majority of them are focused on enhancing existing FL components, strategies, and algorithms or devising better ways of doing FL. However, there is a noticeable scarcity of work that concentrates on the crucial aspects of the initial setup, deployment, and usability of FL. We delve into this issue in greater detail in the dedicated background section (2.1.3).

Because FL is a relatively modern technique, it lacks a sophisticated production-grade eco-system that includes frameworks and libraries that improve ease of use by automating its setup and execution. As a result, contributing to the field of FL or reproducing findings is a task ranging from non-trivial to improbable due to the lack of documented steps regarding setup, deployment, management, and execution. Instead of using a shared set of tools for bootstrapping to make progress on novel work more efficiently, one needs to set up and manage FL from the ground up. Note that a small set of emerging libraries and frameworks exists for FL. Instead of orchestrating FL on real distributed devices, they focus on executing FL algorithms and processes, often via virtual simulations. Not to mention utilizing more advanced techniques to increase productivity that other domains have already been using for several years, such as modern DevOps practices like continuous deployment. We review existing FL tools in detail in the dedicated section (2.1.5).

1.2 Motivation

Let's imagine the mentioned problems could be resolved. What difference would it make? If we invision a universally trusted and used standard tool or set of tools for doing FL, similar to what Docker or Kubernetes have become. This tool would allow researchers, developers, and end-users to easily set up, perform, reproduce, experiment everything FL related. This would not only reduce the necessary level of expert knowledge in this field and related ones but also accelerate existing workflows. FL could be adapted and used by more people in more areas. By the power of automating tedies and error-prone tasks the current and future progress of FL connected work could be accelerated.

1.3 Contribution

In this thesis we introduce FLOps, a novel open-source foundational work and proof-of-concept that enables real (not simulated) FL to be used, developed and evaluated. FLOps enriches FL with best SOTA practices from the realms of automation, DevOps/MLOps, and orchestration. FLOps is intended to be easy to use for people without previous expert knowlege in FL, MLOps or orchestration. Users can simply provide their ML code in form of a git repository (e.g. in GitHub) as long as this code satisfies some simple structural prerequisites. This repo code gets automatically augmented by FLOps to support FL. FLOps creates a containerized Image with all necessary dependencies to do FL training. These images get automatically build and follow best practices to be as fast and light-weight as possible. These images can be build for multiple different target platforms. Thus enabling FL component images to run e.g. on arm devices like PIs or Nvidia Jetsons. With the help of Oakestra deployes and orchestrates these FL components. FLOps automatically performs FL training. This training process can be observed during runtime via the use of SOTA MLOps tools like MLflow, which offers a sophisticated GUI where users can observe, compare, store, export, share, and organize training runs, metrics, and trained models.

FLOps uses Flower as a FL framework which so far does not support Hierarchical FL. As far as we know, FLOps is the first work that combines Flower and MLflow and allows HFL.

As far as we know, the term FLOps (besides the unit of how powerful something can compute things) has not been used, thus this work is called FLOps but should also open the door into future exiting developments for dedicated ML/Dev-Ops practices specifically for FL.

FLOps also allows to automatically build an inference server/service based on the trained model. This image can then be pulled like any other image by the user and used for arbitrary purposes. If the user requests it FLOps can also directly deploy this trained-model image as an inference service in Oakestra.

Besides the end-user perspective FLOps is intended to be a foundational piece of software that should be easily modifiable and extendable for developers and researchers. We put a lot of effort into writing high quality code, using SOTA libraries and frameworks. We also added many developent friendly features into FLOps. E.g. enforced proper styling and typing via formatters and linters, including CI. We createad ready-made static images and services that can be used and extended to automate development and evaluation workflows. E.g. we have a Mock-Data-Provider image/service that can act as a data provider to populate the data used for FL training. Or the Inference Service that can be deployed with the base case FLOps project to verify that the trained model works as expected and is able to properly do inference serving. All these images support arm and amd architectures.

We also add base-images with optinal development flags to speed up build and execution times of FLOps so that developers can verify and check their changes more rapidly.

On top of that we also implemented a new CLI tool for Oakestra from scratch that is used to interact with Oakestra's and FLOps API. Besides that this configurable CLI tool also is capable of visualizing current proceses in a human friendly way in real time as well as trigger evaluation runs and other automated tasks like installing necessary dependencies onto the local machine.

1.4 Thesis Structure

2 Background

2.1 Federated Learning

2.1.1 Basics

2.1.2 Architectures

2.1.3 Research

2.1.4 Industry

2.1.5 Frameworks & Libraries

2.1.6 Flower

2.2 Machine Learning Operations

2.2.1 DevOps

2.2.2 MLOps

2.2.3 MLflow

2.3 Orchestration

2.3.1 Fundamentals

2.3.2 ML Containerization & Orchestration

2.3.3 Oakestra

2.4 Related Work

3 Requirements Analysis

4 System Design

5 Object Design

6 Evaluation

6.1 Rationale

6.1.1 Chosen Experiments

6.2 Experimental Setup

6.2.1 Monolith

6.2.2 Multi-Cluster

6.2.3 Evaluation Procedure

6.3 Results

6.3.1 Basics

6.3.2 Image Builder

6.3.3 Different ML Frameworks/Libraries & Datasets

6.3.4 Multi-cluster & HFL

7 Conclusion

7.1 Limitations & Future Work

7.1.1 Federated Learning via FLOps

7.1.2 Complementary Components & Integrations

8 DELME tum example

8.1 Section

Citation test [2].

Acronyms must be added in `main.tex` and are referenced using macros. The first occurrence is automatically replaced with the long version of the acronym, while all subsequent usages use the abbreviation.

E.g. `\ac{TUM}, \ac{TUM}` ⇒ Technical University of Munich (TUM), TUM

For more details, see the documentation of the `acronym` package[1].

8.1.1 Subsection

See Table 8.1, Figure 8.1, Figure 8.2, Figure 8.3.

Table 8.1: An example for a simple table.

A	B	C	D
1	2	1	2
2	3	2	3

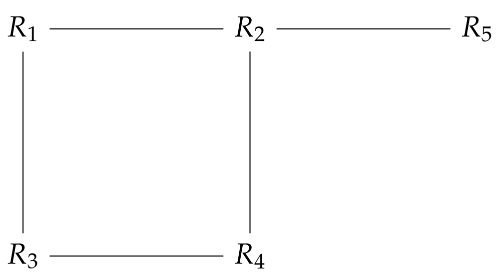

Figure 8.1: An example for a simple drawing.

[1] `https://ctan.org/pkg/acronym`

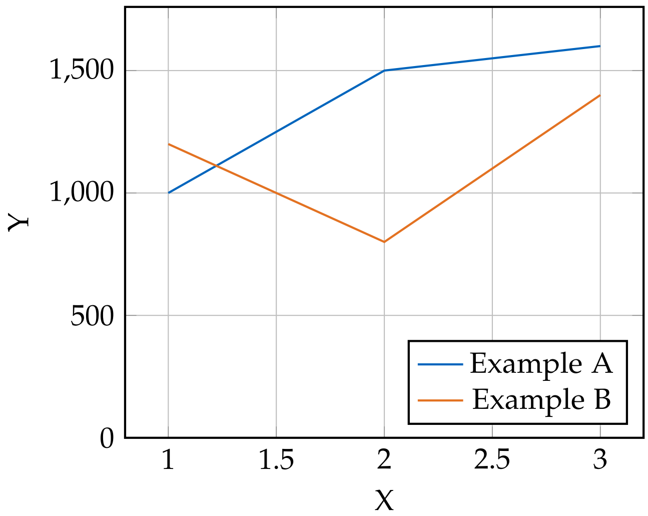

Figure 8.2: An example for a simple plot.

```
SELECT * FROM tbl WHERE tbl.str = "str"
```

Figure 8.3: An example for a source code listing.

Abbreviations

TUM Technical University of Munich

List of Figures

List of Tables

Bibliography

[1] A. Bourechak, O. Zedadra, M. N. Kouahla, A. Guerrieri, H. Seridi, and G. Fortino. "At the Confluence of Artificial Intelligence and Edge Computing in IoT-Based Applications: A Review and New Perspectives." In: *Sensors* 23.3 (2023). ISSN: 1424-8220. DOI: 10.3390/s23031639.

[2] L. Lamport. *LaTeX : A Documentation Preparation System User's Guide and Reference Manual*. Addison-Wesley Professional, 1994.

[3] C. Legislature. *California Consumer Privacy Act (CCPA)*. Online; accessed August 11, 2024. 2018.

[4] H. Ludwig and N. Baracaldo, eds. *Federated Learning - A Comprehensive Overview of Methods and Applications*. Springer, 2022. ISBN: 978-3-030-96896-0. DOI: 10.1007/978-3-030-96896-0.

[5] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas. "Communication-Efficient Learning of Deep Networks from Decentralized Data." In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Ed. by A. Singh and J. Zhu. Vol. 54. Proceedings of Machine Learning Research. PMLR, 20–22 Apr 2017, pp. 1273–1282.

[6] T. E. Parliament and Council. *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)*. Online; accessed August 11, 2024. 2016.