MLOps for evolvable AI intensive software systems

Sergio Moreschini
Tampere University
sergio.moreschini@tuni.fi

Francesco Lomio
Tampere University
francesco.lomio@tuni.fi

David Hästbacka
Tampere University
david.hastbacka@tuni.fi

Davide Taibi
Tampere University
davide.taibi@tuni.fi

***Abstract*—DevOps practices are the de facto sandard when developing software. The increased adoption of machine learning (ML) to solve problems urges us to adapt all the current approaches to developing a new standard that can take full benefit from the new solution. In this work we propose a graphical representation for DevOps for ML-based applications, namely MLOps, and also outline open research challenges. The pipeline aims to get the best of both worlds by maintaining the simple and iconic pipeline of DevOps, yet improving it by adding new circular steps for ML incorporation. This aims to create an ML-based development subsystem that can be self-maintained, and is capable of evolving side-by-side with the software development.**

***Index Terms*—MLOps, AIOps, Software Engineering, DevOps**

I. Introduction

Companies are getting more and more interested in applying DevOps principles for Machine Learning (ML) based software. DevOps brings people, processes, and tools together to produce continuous values at high velocity [1], connecting Development and Operations as a continuous pipeline.

Developing agile software based on ML requires different steps compared to normal DevOps. The incorporation process of an ML pipeline when developing software needs to be carefully addressed so that the ML-based software system can be assured of long-term maintainability and evolvability.

The goal of this paper is to define a clear pipeline when practicing DevOps to ML-based software, namely MLOps. Such a pipeline will provide essential guidelines when developing ML-based software starting from DevOps.

Therefore, we propose an extension of the DevOps pipeline, adapted to MLOps. In this work we aim at highlighting the diversification yet affinity when developing both software and ML procedures essential for the creation of ML-based software, providing a more comprehensive list of steps, differentiating the steps required for the development of the ML-specific code and the whole system development. Our work differs from [2] in that we aim to give a high level overview of the process.

Multiple works faced the problem of defining a new pipeline capable of describing the process in a simple yet complete way. Zhou et al. [3] verified the feasibility of building an ML pipeline with CI/CD capabilities. In [4] and [5] multiple challenges have been addressed when adopting MLOps. Such challenges have been categorized in context and data and educating for AI operations. John et al. [6] conducted a literature review to present the state of the adoption of MLOps in practice to derive and validate a framework for the identification of activity involved when adopting MLOps. Mäkinen et al. [7] performed a survey with the goal of understanding to what extent MLOps is needed in today's ML operations. The result of the survey showed that as we are moving towards ecosystems where teams of developers are working closely in ML development, there is an urge to define a clear MLOps pipeline. The reasons behind this are the inability to simply apply *plug-and-play* DevOps tools and principles, and the lack of a definition of a full stack ML pipeline. Colantoni et al. [8] defined a theoretical model to integrate ML into DevOps, however the model does not propose a visualization of the whole pipeline, nor a complete integration of the ML steps.

Researchers investigated different aspects of MLOps, but did not provide a clear picture of the different processes adopted for the ML and for the non-ML software processes.

In the remainder of this paper, we present our proposal of MLOps representation, as well as open research challenges towards a joint research plan on MLOps practices.

II. From DevOps to MLOps

The adoption of DevOps practices in software development has become a requirement in most scenarios nowadays. The reason behind this is not only related to the improved results achieved, but also the culture created by the adoption of these practices. In a world where the new mantra is ***be faster, be more agile***, DevOps fits perfectly in, due to its dynamic nature based on continuous learning and improving.

The infinite DevOps loop aims to portray the role division of application Developers (Dev) and the IT Operations (Ops) teams in a singular team. The Dev is responsible for planning, coding, building and testing while the release, deploy and monitor are the tasks of the Ops. The rapid adoption of ML-based software created a new figure: the ML developer. Such figures exist and operate in parallel with the software developer and therefore needs to be embedded in the DevOps pipeline.

III. The evolution of Dev

In this Section, we make our proposal for a new graphical representation for MLOps (Figure 1). The introduction of ML, does not interfere with the operations. Therefore, our proposal mainly lies in the Development cycle. The Operations cycle is mostly unchanged from traditional DevOps (i.e., the ML models might need to be updated in production with new data collection [9])

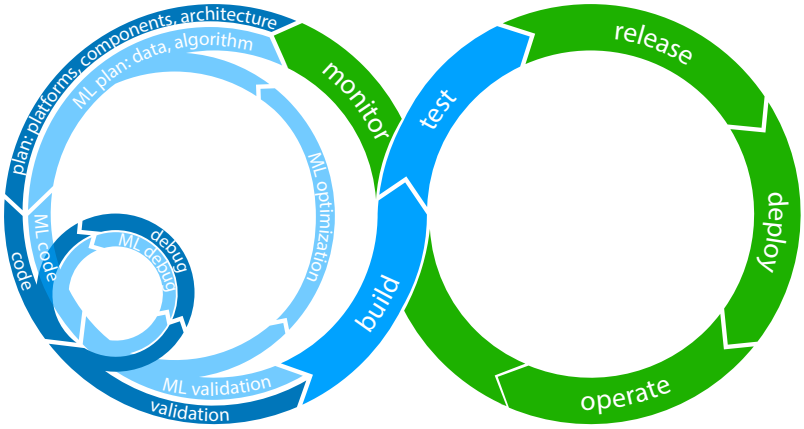

Fig. 1. MLOps pipeline.

The PLANNING phase covers everything that happens before the developers start writing code, including requirement gathering and architectural design. While considering ML, the planning refers to the identification of the problem to be solved and the available data. Based on these, the most appropriate data analysis approaches can be selected (e.g., classification or regression, supervised or unsupervised), and suitable algorithms can be selected (e.g. Random Forest, Deep Learning). In the case of supervised algorithms, a data labelling step might be also needed.

In the CODING phase, both the system and the ML code need to be implemented and then locally validated.

The VALIDATION, in the traditional DevOps, is usually performed before committing the code, running tests locally. The ML validation refers to the evaluation of the performance of the ML model, with data not previously encountered. If the validation proves that the ML-based approach is not suitable for the data or for the algorithms, it is necessary to return to the planning step to optimize the model to better tackle the problem. Once the ML optimizations and local testing of the system have been performed, the ML code needs to be integrated into the system code.

The system BUILDING and TESTING phases are then performed as in traditional DevOps.

IV. OPEN RESEARCH CHALLENGES

Our roadmap, towards an integrated MLOps process, includes several challenges that might be investigated within the software engineering and the ML community. We started our investigation [10] [5], highlighting the main software quality issues of AI systems. In the future we aim at:

- Understand how MLOps is adopted in companies.
- Delineate what kind of problems MLOps practices may be best suited for.
- Define MLOps by adapting the concepts, methods and tools used in DevOps.
- Explore other fields of research where MLOps practices can be performed (e.g. CVOps [11]).

V. CONCLUSIONS

The increased adoption of ML-based software generated a demand for ML developers who need to perform tasks in parallel to software developers.

In this paper we propose a better representation for MLOps, including ML development steps into the traditional DevOps practices. The MLOps pipeline focuses on the dualism between software and ML developers and their tasks. Compared to normal DevOps we have a new block (validation) and those related to plan and code are now field-specific. Moreover, the validation blocks produce two extra loops (optimization and debug) for the ML side and one for the software side (debug).

This work is part of a roadmap for the development of MLOps practices that will be investigated collaboratively with the ML and software engineering communities.

REFERENCES

[1] L. Bass, I. Weber, and L. Zhu. *DevOps: A software architect's perspective*. Addison-Wesley Professional, 2015.
[2] K. Salama, J. Kazmierczak, and D. Schut. Practitioners guide to mlops: A framework for continuous delivery and automation of machine learning.
[3] Y. Zhou, Y. Yu, and B. Ding. Towards MLOps: A case study of ML pipeline platform. In *ICAICE'20*, pages 494–500, 2020.
[4] D. A. Tamburri. Sustainable MLOps: Trends and challenges. In *SYNASC 2020*, pages 17–23, 2020.
[5] Li Xiaozhou, Sergio Moreschini, Aleksandra Filatova, and Davide Taibi. Knowledge management challenges for AI quality. In *1st Workshop on Software Quality Assurance for Artificial Intelligence (SQA4AI 2022). Colocated with SANER 2022*, 2022.
[6] M. M. John, H. H. Olsson, and J. Bosch. Towards MLOps: A framework and maturity model. In *Euromicro / SEAA*, 2021.
[7] S. Mäkinen, H. Skogström, E. Laaksonen, and T. Mikkonen. Who needs MLOps: What data scientists seek to accomplish and how can MLOps help? *Workshop on AI Engineering – Software Engineering for AI*, 2021.
[8] A. Colantoni, L. Berardinelli, and M. Wimmer. DevOpsML: Towards modeling DevOps processes and platforms. In *MODELS'20*, 2020.
[9] F. Khomh, Y. Guéhéneuc, S. Okuda, N. Natori, A. Seiki, and N. Shioura. Software engineering patterns for machine learning applications (sep4mla)-part 2.
[10] V. Lenarduzzi, F. Lomio, S. Moreschini, D. Taibi, and D.A. Tamburri. Software quality for ai: Where we are now? In *Int. Conf. on Software Quality*, 2021.
[11] Castelli V. How to apply MLOps to computer vision? Introducing CVOps. https://medium.com/picsellia/how-to-apply-mlops-to-computer-vision-introducing-cvops-355dcda9f949, 2021.