MLOps

Challenges and Opportunities

Authors

Gabriel Christensson

August Tengland

We certify that generative AI, incl. ChatGPT, has not been used to write this essay.
Using generative AI without permission is considered academic misconduct.

1 Introduction

The rapid adoption of DevOps (Development and Operations) principles in the world of software engineering has not gone unrecognized. These principles practically constitute a new IT paradigm spearheaded by a rising DevOps community to which organizations are progressively shifting. The reach of DevOps now spans multiple fields of computer science and companies in all lines of business are following the trend of automation. One of the fields in which automation is currently researched is Machine Learning (ML). ML systems can be challenging to maintain in production and may therefore benefit from increased automation and monitoring [1]. To operate such systems more efficiently both in production and development, MLOps (Machine Learning and Operations) was introduced as the practice of applying DevOps principles to ML systems. However, introducing new technologies and manners of working is not straightforward. It is therefore of interest to analyze and discuss the different challenges and opportunities associated with this transition.

2 Background

With the rise of ML-based applications such as speech recognition, product recommendations, and language translation the interest has surged for mechanisms that help make such systems more maintainable and operational. Some of the mechanisms may be classified as principles, others as tools. For this reason, terms such as MLOps can be ambiguous as what they refer to. In this section, a definition of MLOps and parts of the research, technologies, and principles related to it are presented.

2.1 Defining MLOps

Mark Treveil et al. defined MLOps as: "...the standardization and streamlining of machine learning life cycle management." [2, p. 4]. This definition explains the core idea while maintaining a comfortable generality. For the purpose of this essay, it is adopted here as well.

2.2 Demystifying MLOps

Before diving into the meaning of MLOps it is necessary to understand the principles of DevOps. Given the scope of this essay, it is assumed that the reader is familiar with them, and thus only the core DevOps concepts are revisited.

Practicing DevOps means having teams for development and operations (Dev and Ops) work together to build and deliver software products rapidly. Essentially, DevOps is all about automating the software development life cycle. Tasks that can be accomplished without human intervention such as testing, building, and deploying are automated to avoid introducing human errors and to give the Dev team more time to develop and the Ops team more time to monitor and maintain. Concurrently, the teams should collaborate and exchange feedback to improve future releases.

Practicing MLOps is practically the same as practicing DevOps but in the setting of developing/operating machine learning models. Of course, it differs somewhat from the traditional DevOps practices since there must be features specific to ML added to the mix. The idea of applying DevOps principles to ML systems emerged as these systems scaled and when a clear need for further automation was expressed.

2.3 The Need for MLOps

In 2014, data scientists at Google authored a publication in which they summarize multiple pitfalls related to operating ML systems [3], which may serve as a set of arguments for the transition to MLOps. One of the points made in the publication is that ML systems have a sizeable system-level complexity. The code for the ML model occupies only a small portion of the entire system and there are large components at play that make up the overall system operations. Thus, the interplay between all components is essential [4].

2.4 Bringing an ML Model To Production

To sufficiently grasp the challenges associated with MLOps one must understand the steps involved in bringing a machine-learning model to production. Below is a road map of that process as originally described in [1] and figure 1 conceptually illustrated the MLOps life cycle.

1. Define use cases, business requirements, and success criteria.
2. **Data extraction:** Select data from relevant sources.
3. **Data preparation:** Clean the data, split data into training/validation/training sets, and transform data if necessary.
4. **Model training:** Use the prepared data to train different models and tune hyperparameters, and select the best-performing model.
5. **Model evaluation:** Evaluate the model quality by testing it on a holdout test set (data that has never been used in training).
6. **Model validation:** Validate that the model reaches established success criteria and is ample for deployment.
7. **Model serving:** Deploying the model.
8. **Model monitoring:** Continually monitor the performance of the model to eventually schedule a new iteration in the ML process.

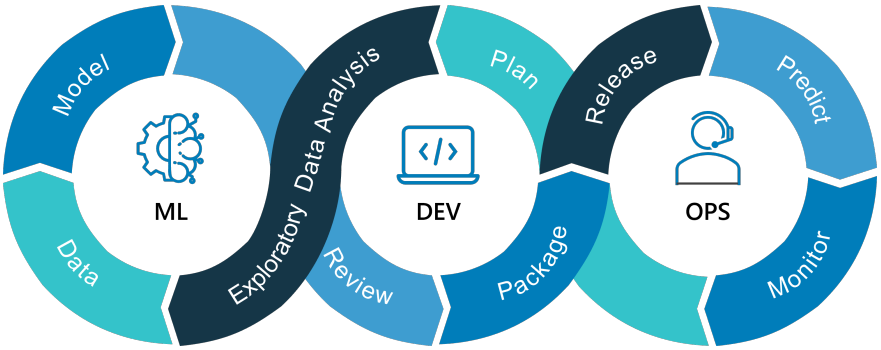

Figure 1: Conceptual life cycle of MLOps [5]

3 Benefits of MLOps

There are many benefits of adopting MLOps practices to ML systems and they mostly resonate with those of using DevOps. Naturally, having automated processes will speed up production and delivery, but there is also greater resource allocation. Indeed, with ML systems being introduced in applications that have high safety requirements such as autonomous vehicles and medical diagnosis, developers and operators must not waste time on tasks that can be automated [6]. Another advantage of using MLOps is scalability. Having successfully incorporated MLOps practices with efficient ML life cycle management means that it is easier, faster, and safer to bring new models into production or expand existing ones.

In practice, introducing an MLOps pipeline not only allows for continuous integration and automatic testing, but it can also significantly improve the *data handling*, *retraining*, and *continuous validation* of the system. Creating an automation pipeline not only allows for easier initial deployment but also makes the process of *iteratively improving* models a lot faster, since this procedure is in many ways formulaic and lends itself well to automation. This can be considered a difference between MLOps when compared to DevOps applied for traditional systems, where the steps required to improve performance can be harder to predict and require manual labor and creativity.

4 Current Challenges in MLOps

While MLOps has undeniably advanced the development of adaptable models and could be considered a *de facto* standard for organizations creating ML systems, there is still room for growth. Many MLOps pipelines, if not all, still require humans to a larger extent than typical DevOps systems. This is not surprising, as the nature of machine learning poses *unique* challenges when applying DevOps principles. Some of these differences include:

- *Data Handling:* MLOps pipelines need to manage not only system code but also data. The data might need to be cleaned, transformed, validated, and logged [1] [7].
- *Automatic Testing:* Testing models require data and can be both difficult and costly [1] [8].
- *Concept Drift:* As the data distribution of an ML model is trained on changes, the model will start to decay. Re-training is often necessary to maintain performance over time [1] [8].

Tackling these issues requires DevOps tooling specific to the domain of MLOps. The rest of the section dives deeper into these concepts and discusses some of the challenges currently being tackled to advance MLOps:

4.1 Data Management and Quality

As data is the foundation for any machine learning system, it is important that the data collection and manipulation process is well integrated into the MLOps pipeline. A frequently occurring issue in MLOps systems is managing data in accordance with the different constraints and restrictions. The pipeline might, for example, be subject to local data regulations which require that data is contained within the organization [7]. This makes DevOps practices such as automation and cloud services challenging to apply in the data management stage as they must conform to these regulations (which might differ based on where the system is deployed). As such, there is a need for more sophisticated tooling which allows managing data in a "lawful" way throughout a distributed environment.

In addition to this, the quality of a machine learning model is generally decided, in large, by the quality of data that has been supplied to it [8]. Consequently, introducing data optimization techniques into the MLOps pipeline, such as data cleaning, is a promising way to increase model performance. Cleaning is however expensive and requires a degree of manual labor, creating a desire for systems that can help optimize this process. Solutions have been proposed that can analyze how noisy data *propagates* into the final model performance, providing a decision basis on how to conduct cleaning [8] [9]. However, we have yet to see systems expanded to a general setting where they could be implemented in an existing pipeline with arbitrary ML models, giving way for further research.

4.2 Automatic Testing

As previously stated, testing machine learning systems is rarely a simple procedure. One problem in automatic testing for ML is that test results from models are inherently random, creating a need for test cases that are both meaningful and can account for this variance. [10] propose a CI system for ML that handles this issue by allowing test conditions based on *probability*, meaning that we can evaluate whether the difference between models is *statistically significant*. This solution is however dependent upon a human-in-the-loop that can manage these conditions, meaning that it is not trivial to implement it into an existing pipeline. As such, there is a need for more sophisticated testing solutions that allow for easier adoption.

Another problem in testing, which separates MLOps from traditional DevOps, is that re-using the same test cases risks *overfitting*, where the models become over-trained on the test cases and cannot generalize beyond them. It is generally agreed that a model should not be evaluated against the same test set numerous times [8], however, data is expensive, and discarding test sets after each trial is seldom feasible. This issue is also addressed in the system proposed by [10], by introducing an optimized *estimator* for the number of test set re-uses that are possible. The fundamental problem does however remain, and the ephemeral validity of test sets will likely remain an issue in MLOps.

4.3 Continuous Training and Validation

One of the biggest things separating MLOps from traditional DevOps is the emphasis on *feedback loops* that are present in ML systems. For example, a model deployed to production will collect data that can be used to train new iterations of said model. This process is of course not exclusive to ML, in fact many *agile* systems operate in this manner, however this feedback loop is generally not *automated* for traditional systems but rather part of a manual design process. A large part of the necessity for feedback loops in ML is contributed to *data drift* and *concept drift*, where the underlying data distribution or problem might change over time (consider, for example, how a model predicting *shopping* behaviour might be affected by seasonal changes) [7].

Automating the *retraining* of a model poses a number of issues. For example, how do we know that the model is performing worse when it is evaluating unlabeled data? Manually labeling new data is, as stated in the previous section, expensive and requires manual labor (when *automatic labeling* cannot be applied). For this reason, many MLOps pipelines still rely on manual evaluation for when to trigger the retraining process [7]. Another solution, employed by some ML pipelines, is to have *scheduled* retraining times at fixed intervals. This solution does, however, imply that the system is not flexible to sudden changes in data or performance, defining the appropriate retraining interval is also subject to manual interference.

One might argue that the issue of unlabeled data is trivial, as using data for training requires it to be

labeled regardless (when applying *supervised learning*). It is, however, possible, and not uncommon, that a system is composed of multiple models and wants to know which is most applicable at a given time, without doing any retraining. This is a process that benefits greatly from automation and while there are systems that lower the need for manual labor, such as [8], the goal should be to automate the process completely, while still ensuring reliability.

5 Reflection

While new technologies and concepts are born from opportunities and needs they also come with challenges that must be critically explored to enable advancement and employment. The increasing number of applications that rely on ML expedites the need for sustainable, scalable, and resilient systems; qualities that are offered by MLOps. This essay presents a selection of challenges that are of interest to researchers working to advance the field of MLOps. Clearing some of these bottlenecks would lower the cost of supporting big models and would likely cause not only increased performance but also a larger adoption of ML solutions across the industry. With this in consideration, it must be acknowledged that some of the problems highlighted are connected to the fundamental nature of machine learning, such as the need for manual labeling, which can indicate that they might never be resolved completely. Research has however shown that for these issues, which create a need for manual interference, we are able to optimize the process to minimize this workload. [8] is a good example of such solutions, and we are likely to see further improvements in all of the proposed research areas. MLOps is still evolving at a rapid pace and, as it has clearly demonstrated its worth, should continue to receive attention.

References

[1] Google, *Mlops: Continuous delivery and automation pipelines in machine learning*. [Online]. Available: https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning.

[2] M. Treveil, N. Omont, C. Stenac, *et al.*, *Introducing MLOps*. O'Reilly Media, 2020.

[3] D. Sculley, G. Holt, D. Golovin, *et al.*, "Machine learning: The high interest credit card of technical debt," in *SE4ML: Software Engineering for Machine Learning (NIPS 2014 Workshop)*, 2014.

[4] S. Mäkinen, H. Skogström, E. Laaksonen, and T. Mikkonen, "Who needs mlops: What data scientists seek to accomplish and how can mlops help?" In *2021 IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN)*, 2021, pp. 109–112. DOI: 10.1109/WAIN52551.2021.00024.

[5] N. Analytics, *Machine learning operations (mlops)*. [Online]. Available: https://nealanalytics.com/expertise/mlops/.

[6] Y. Zhou, Y. Yu, and B. Ding, "Towards mlops: A case study of ml pipeline platform," in *2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE)*, 2020, pp. 494–500. DOI: 10.1109/ICAICE51518.2020.00102.

[7] M. Steidl, M. Felderer, and R. Ramler, "The pipeline for the continuous development of artificial intelligence models—current state of research and practice," *Journal of Systems and Software*, vol. 199, p. 111 615, May 2023. DOI: 10.1016/j.jss.2023.111615. [Online]. Available: https://doi.org/10.1016%5C%2Fj.jss.2023.111615.

[8] C. Renggli, L. Rimanic, N. M. Gürel, B. Karlaš, W. Wu, and C. Zhang, *A data quality-driven view of mlops*, 2021. arXiv: 2102.07750 [cs.LG].

[9] S. Krishnan, J. Wang, E. Wu, M. J. Franklin, and K. Goldberg, "Activeclean: Interactive data cleaning for statistical modeling," *Proc. VLDB Endow.*, vol. 9, no. 12, pp. 948–959, Aug. 2016, ISSN: 2150-8097. DOI: 10.14778/2994509.2994514. [Online]. Available: https://doi.org/10.14778/2994509.2994514.

[10] B. Karlaš, M. Interlandi, C. Renggli, *et al.*, "Building continuous integration services for machine learning," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '20, Virtual Event, CA, USA: Association for Computing Machinery, 2020, pp. 2407–2415, ISBN: 9781450379984. DOI: 10.1145/3394486.3403290. [Online]. Available: https://doi.org/10.1145/3394486.3403290.