Machine Learning

KTH | Royal Institute of Technology

Automated Software Testing and DevOps

MLOps: the past, the present and the future

Authors:
Ennio Rampello
Vishal Nedungadi

E-mail addresses:
ennio@kth.se
vishaln@kth.se

April 16, 2022

Contents

1 Introduction

Machine Learning Operations (MLOps), is rapidly growing in the field of software engineering. The use of MLOps has been proven effective in terms of development, deployment and monitoring of machine learning models. It sees its origins from DevOps, which was developed to effectively manage and develop software products. In this essay, we are going to talk about the past of MLOps, the present and the future of MLOps. The past is going to show that the origins of MLOps, was seen in a research paper published by Google [1] that talks about technical debt. We will see more about the paper in later sections of the essay. We will also be talking about where MLOps is at the moment.

Specifically in terms of what current tools help in effectively developing, deploying and monitoring machine learning models. We will also delve into what the future of MLOps looks like. Finally we will end with a reflection on which aspects of the essay and MLOps are the most important.

2 A brief History

The genesis of MLOps goes back to 2015, when Google published the paper "Hidden Technical Debt in Machine Learning systems" [1]. We can see the first signs of a need for a automated system to make the process of development of machine learning models much more efficient. Of course, this is related to DevOps, because MLOps is just an advancement over DevOps.

2.1 What is DevOps?

Before DevOps, there was Agile. Agile helped in making the process of development faster, with improved team collaborations, and teams were more equipped to handling errors and bug fixes faster. But agile was still far from perfect. The main issue was that all the burden and pressure was on the developers and the “scrum masters” had little to no work. Hence lot of delays were expected, and this led to slower deployment and development times. This was when DevOps emerged. DevOps had the features of Agile and included lot of integrations between various teams which led to much better development. These integrations could be manual, and many of them are also automated. With the help of DevOps, the developers could focus on the main development, testing engineers could write separate scripts to automate the whole testing process, and system admins could also write scripts to perfectly integrate all the components from various teams. With the development of DevOps, security became an add-on, and was established when DevOps practices were followed.

2.2 What is MLOps?

MLOps is the set of practices which help in deploying and maintaining various machine learning models, when they are in production. Generally they are automated, and hence the whole process becomes more efficient. It is the advancement of DevOps but for Machine Learning and AI, and just like DevOps, it helps to increase automation such that developers can continue improving the model’s accuracy, without having the burden of deployment and other factors. In addition to the basic DevOps steps i.e., Continuous Integration and Continuous Deployment/ Delivery, MLOps contains an extra step called Continuous Training. To illustrate why MLOps is necessary, Figure 1 shows the possible changes that could occur to a machine learning system.

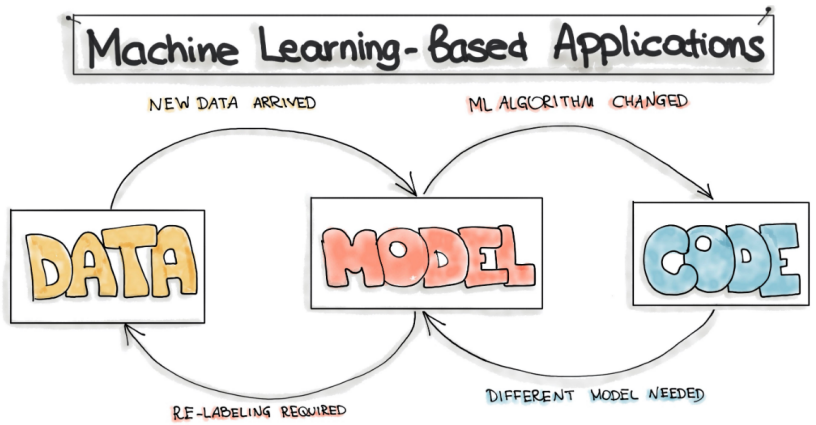

Figure 1: Various changes that can occur in a Machine Learning System [4]

There are various other steps involved within each of these categories, for example, Data can involve pre-processing, augmentation, normalizing, feature engineering, labelling etc.

2.3 How did MLOps begin?

We can see that from [1], the need for MLOps was established in the year 2015 by Google. [1] talks about "technical debt" and how it is also important for Machine Learning systems. So, to understand about this, we need to know what technical debt is.

Technical debt [11] is the concept of delaying a certain work or development phase to reach a deadline. That usually means using simpler or easier methods to solve the problem. Most times the method would have to be changed after the deadline has been met, and this additional time take to program the extra method again is also factored into the technical debt. Let's take an example: Lets say that we were building a machine learning model to show amazing results on a classification task. To meet our deadline, we need to create the model quickly and show it to our customer. But immediately after deploying we realize that this model does not scale well at all, and we need to change the whole model. This additional time in re-developing the model from scratch and performing the testing/validation is factored into the technical debt. As stated in [2] "Technical Debt is the coding you must do tomorrow because you took a shortcut in order to deliver the software today".

We can see that technical debt is also important when it comes to deploying and developing machine learning models. The paper also mentions that many problems of technical debts occur especially during the model selection phase. Hence MLOps helps us in such a way, that if we can automate every aspect of the development, and integration, then changing the model will not have much of an impact on the final deployment deadlines. One of the other reasons for using MLOps is that apart from model changes, the testing conditions of Machine Learning models can adapt over time. For example, the real-world data might change, extra biases might have to be added/removed, new tasks would have to be predicted or classified. Re-developing the model for each of these cases, every time it happens can be very strenuous. Automated MLOps integration can help make the whole process of development, deployment, and training as efficient as possible.

3 Modern technologies

3.1 Overview

Nowadays, MLOps is referred to as a "set of practices that aims to deploy and maintain machine learning models in production, reliably and efficiently" [wikipedia: MLOps]. In order to efficiently and effectively make use of Machine Learning inside a company it is not sufficient to simply rely on a model with high accuracy on historical data. Models are often deployed to deal with environments that are constantly evolving, and what worked on historical data is not ensured to be working on future data. This is the reason why most of the deployed models have to be constantly retrained and updated, in order to follow the latest standards and to be trained with the most recent data available. There are several aspects in the lifecycle of a ML model and there are several tools that handle a specific side of the complete pipeline. The most used tools nowadays are:

- *Jupyter Notebook* [5]: this is a tool that every Machine Learning engineer starts using from the early days of their career. It is an interactive development environment that is mostly used during the development phase of a model.
- *JupyterHub* [5]: this tool is closely related to the Jupyter Notebook, in which it allows several developers to contribute to the same project by providing a means to handle multiple Jupyter Notebooks on the same machine, and manage their interactions.
- *Cloud computing platforms*: these platforms include GCP, AWS, Azure and many others. They are crucial to the development of a model because of the computational resources that are needed to train the models, especially if they involve many parameters and large sets of data.
- *Tensorboard* [6]: another very important aspect of the ML pipeline is being able to monitor and manage the performance of the model to be ready to make changes in the parameters if some anomalous behaviours are detected. Tensorboard allows all this by providing the user with an easy-to-use interface to observe the advancement of the training of the model and fine-tune the hyperparamenters.
- *Spark* [7]: when developing and deploying ML models, oftentimes engineers have to deal with massive datasets and they need a way to handle them. Sparks allows the management of several petabytes of data, by distributing it across a cluster of several machines.

- *Kubernetes* [8]: this tool is very important for the deployment of a Machine Learning model. It allows us to have a scalable system that can automatically increase the amount of resources needed to handle the growing number of requests to the system.
- *Kubeflow* [9]: when having such a large number of processes to deal with in a ML pipeline, a need is created for a system that is able to orchestrate the whole pipeline end-to-end. This is exactly what Kubeflow does, by integrating many different tools together to produce the final result.

All these tools are very broadly used by Machine Learning engineers and Data Scientists around the world. However, there are many other interesting tools that are used but that we can't cover here, and we invite the reader to check them out because they might represent a better suite for specific needs.

3.2 Jupyter Notebook and JupyterHub

The Jupyter Notebook is the first tool that every ML engineer starts using when it comes to begin developing a new model. It is based on Python and it has many different features that make it a very effective and user-friendly tool.
This interface allows the user to run different pieces of code separately, while still keeping all the variables common across all the code cells. This feature is very important when building a model because it often happens that some modifications have to be made to the code without the need of retraining the model or reloading the dataset, which would otherwise take a really long time to execute.

The Jupyter Notebook is also interactive, which means that it allows the developer to visualise data interactively through different libraries that are suited for different cases. The most common one is Matplotlib, which is a general plotting tool, but there are many others for more specific needs (geographic maps, tables, etc. . .).

Since all the outputs of the cells are kept in the Jupyter Notebook, it is also a great tool to show the results and the entire process that led to them. There is also the possibility to insert Markdown cells in between code cells to increase the readability of the notebook.

JupyterHub is a software that allows the handling of multiple Jupyter Notebooks on the same machine. This feature becomes really important when multiple developers have to collaborate on the same project and there is a need for a system that manages all the different connections to the server and optimises resources utilisation.

3.3 Spark

Spark is the most popular open-source engine for distributed data processing. Efficient data management is crucial to the development of a ML pipeline, from the earliest stage when data preprocessing is carried out, all the way to the deployment phase. Spark allows you to build a cluster of several hundreds or thousands of

machines and distribute the data across all of them. Furthermore, it excels at handling data streams, which is a very important feature during the deployment phase, when the data is constantly generated and fed to the model.

It is a very flexible framework and allows the integration with many of the other mentioned tools. In particular, it can be run together with Kubernetes in order to build a complete cluster that handles both the computational resources and the storage capacity.

3.4 Kubernetes

Kubernetes, also known as K8s, was originally designed by Google and is now being maintained by the Cloud Native Computing Foundation. It is basically a system that manages containers, where each container is usually employed to run a specific microservice (that is usually part of a large set of microservices interacting together to deliver a multifunctional system) inside a specific machine.

The general workflow using containers is rather manual, that is, each container has to be manually created (e.g. with Docker) and configured to run whatever application is supposed to be inside it. However, Kubernetes allows to automate this process by creating and scaling containers, depending on the load of the system. As a result, Kubernetes has become a fundamental tool for scaling any kind of system.

Most large-scale applications require that multiple units work together towards a unique goal, that is, the product that gets delivered to several users. However, it is not an easy job to scale such a system to meet the demand that comes from user requests, especially when the system is distributed among different machines that handle different aspects of the backend (e.g. database, computation, web server, etc. . .).
So, given a set of machines, it is possible to manage all of them by creating a Kubernetes cluster, which is a management system that can be composed of either actual machines or virtual machines.

3.5 Kubeflow

Where exactly does Kubernetes become useful while dealing with the deployment of Machine Learning models? There are several reasons that make Kubernetes, and in particular Kubeflow, a good implementation choice when it comes to building a ML pipeline, here are some:

- It is an open-source and well documented framework, allowing fast deployment of models, even without much expertise with ML pipelines.
- It supports multiple cloud computing services, that are crucial for the training of ML models. Such services include AWS, Azure, GCP, etc. . .
- It offers integration with Tensorboard, which is a monitoring tool mostly used to visualise the training process of a model.
- It has an important feature (KFServing) that allows model deployment (on-premises or in the cloud), supporting the most used Machine Learning frameworks (TensorFlow, PyTorch, SciKit Learn).

The integration between Kubeflow and Kubernetes becomes particularly important in the deployment phase. This is because when a ML model has to be deployed, most of the time it has to work as a web application that constantly receives inputs from user interactions and it has to infer an output that is generated by propagating the input through the trained model.

Other times, a ML model might have to work as a backend for the decisions made by an autonomous system (e.g. autonomous driving). So, the edge architecture collects data (through cameras or sensors) and sends this information to the cloud (where the ML model is deployed) which then sends back an action that the system has to perform, based on the situation that it is currently in.

4 A glance into the future !!!

This section is our personal opinion as there are no resources on what the future of MLOps looks like. We believe that in the future, every enterprise that uses Machine Learning, will follow MLOps practices [10]. There is already a shift towards using these practices, but in the future it will become more common and certain between organizations. We also predict that the amount of automation will increase such that models could automatically adapt based on real-world conditions. We could also expect some sort of Machine Learning being applied to the whole MLOps pipeline, such that the pipeline learns the best time to perform retraining, or validation or testing. But such a level of automation is unclear at the moment.

5 Conclusions and Take-home

From the essay, the most logical conclusion is that MLOps helps improve the efficiency and reliability of a Machine Learning system. Every enterprise and organization should make use of MLOps practices in order to make the development and deployment of machine learning models more efficient. There are many tools to help integrate the platform with MLOps (like Kubeflow, MLFlow, etc.), and making use of such tools is beneficial to us.

References

[1] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. 2015. Hidden technical debt in Machine learning systems. In Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'15). MIT Press, Cambridge, MA, USA, 2503–2511.

[2] https://www.bmc.com/blogs/technical-debt-explained-the-complete-guide-to-understanding-and-dealing-with-technical-debt/

[3] https://databricks.com/glossary/mlops

[4] https://ml-ops.org/content/motivation

[5] https://jupyter.org/

[6] https://www.tensorflow.org/tensorboard

[7] https://spark.apache.org/

[8] https://kubernetes.io/

[9] https://www.kubeflow.org/

[10] https://www.iotforall.com/why-companies-need-to-think-about-mlops

[11] https://en.wikipedia.org/wiki/Technical_debt