Automating the training and deployment of models in MLOps by integrating systems with machine learning

Penghao Liang[1a,*],Bo Song[1b] ,Xiaoan Zhan[1c], Zhou Chen[2], Jiaqiang Yuan[3]

[1a]Information Systems, Northeastern University, San Jose, CA
[1b]Computer Science, Northeastern University, Boston, MA, USA
[1c]Electrical Engineering, New York University, NY, USA
[2]Software Engineering, Zheiiang University, Hangzhou, China
[3]Information Studies, Trine University, Phoenix, AZ, USA

*Corresponding author: liang.p@northeastern.edu

Abstract. This article introduces the importance of machine learning in real-world applications and explores the rise of MLOps (Machine Learning Operations) and its importance for solving challenges such as model deployment and performance monitoring. By reviewing the evolution of MLOps and its relationship to traditional software development methods, the paper proposes ways to integrate the system into machine learning to solve the problems faced by existing MLOps and improve productivity. This paper focuses on the importance of automated model training, and the method to ensure the transparency and repeatability of the training process through version control system. In addition, the challenges of integrating machine learning components into traditional CI/CD pipelines are discussed, and solutions such as versioning environments and containerization are proposed. Finally, the paper emphasizes the importance of continuous monitoring and feedback loops after model deployment to maintain model performance and reliability. Using case studies and best practices from Netflix, the article presents key strategies and lessons learned for successful implementation of MLOps practices, providing valuable references for other organizations to build and optimize their own MLOps practices.

Keywords: Machine learning, MLOps, Automated deployment, CI/CD pipeline, Supervisory control.

1. Introduction

Machine learning has revolutionized the way people use and interact with data, driving business efficiency, fundamentally changing the advertising landscape, and revolutionizing healthcare technology. Over the past decade, machine learning (ML) has become an essential part of countless applications and services in a variety of fields. Thanks to the rapid development of machine learning, there have been profound changes in many fields, from health care to autonomous driving. However, the increasing importance of machine learning in practical applications also brings new challenges and problems, especially when it comes to moving models from a laboratory environment to a production environment. Traditional software development and operations methods often fail to meet the specific needs of machine learning models in production, resulting in challenges such as the complexity of model

deployment, difficulties in performance monitoring, and the absence of continuous integration and continuous deployment [1](CI/CD) processes.

To address these issues, attention is being paid to an emerging field called Machine Learning System Operations (MLOps). MLOps is a relatively new term that has gradually gained traction over the past few years. It closely links computer systems and machine learning and considers new challenges in machine learning from the perspective of traditional systems research. [2]MLOps is not just a tool or process, it is a philosophy and methodology that aims to achieve continuous delivery and reliable operation of machine learning models. Against this background, this article will explore ways to automate model training and deployment by integrating systems with machine learning. First, we will review the challenges and problems in existing MLOps, and then lead to the topic of this article, which is how the integration of systems with machine learning can solve these challenges and improve productivity.

2. Related Work

2.1. Review on the development of MLOps

In the past, different software development process models and development methods have appeared in the field of software engineering. Prominent examples include the waterfall model and the Agile Manifesto. These approaches have a similar goal of delivering a production-ready software product. In 2008/2009, a concept called "DevOps" [3] emerged to reduce problems in software development. DevOps is not just a pure approach, but represents a paradigm for solving social and technical problems in organizations engaged in software development. It aims to close the gap between development and operations and emphasizes collaboration, communication, and knowledge sharing. It ensures automation through continuous integration, continuous delivery, and continuous deployment (CI/CD) for fast, frequent, and reliable releases. [4] In addition, it is designed to ensure continuous testing, quality assurance, continuous monitoring, logging, and feedback loops.

The evolution of DevOps to MLOps[5] is a process of extensibility that includes the adaptation of traditional software development processes and the introduction of new technologies. In this process, organizations need to combine the principles of DevOps with the unique requirements of machine learning, redesigning and extending existing continuous integration, continuous delivery, and continuous deployment pipelines to accommodate the development, training, and deployment processes of machine learning models. [6]At the same time, specific tools and techniques for machine learning need to be introduced, such as model warehouses for model versioning, trial tracking systems for automated experiment management, and specific tools for model monitoring and logging. This evolution allows MLOps to better meet the needs of machine learning projects and enable efficient development, deployment, and management of models.

2.2. Traditional system integration deployment

The traditional concept of system integration refers to the integration of computer systems, including the integration of computer hardware platform, network system, system software, tool software and application software, and the corresponding consultation, service and technical support around these systems. It is based on computer related technology reserves, with reliable products as tools, to achieve a specific combination of computer system functions of the engineering behavior. The content of system integration includes integration of technology environment, integration of data environment and integration of application program.

In order to connect these heterogeneous systems and port applications from one system to another, existing proprietary systems must adapt to standard interfaces and transition to open systems. What users want is interoperability between multi-vendor platforms. Therefore, the work of system integration is very important in the construction of information system projects. It integrates all kinds of resources organically and efficiently through hardware platform, network communication platform, database platform, tool platform and application software platform to form a complete workbench. However, the

quality of system integration has a great impact on system development and maintenance. The basic principles to be followed in technology include: openness, structure, advancement and mainstreaming.

2.3. Automation benefits of machine learning systems

MLOps is an ML engineering culture and practice that seeks to unify ML System development (Dev) and ML System Operations [7](Ops). Practicing MLOps means advocating automation and monitoring in all steps of ML system building, including integration, testing, release, deployment, and infrastructure management.

Machine learning systems differ from other software systems in the following ways:

- Development: Machine learning is experimental in nature. You should try different features, algorithms, modeling techniques, and parameter configurations to find the best fit for the problem as soon as possible. The challenge is keeping track of what works and what doesn't, and maintaining repeatability while maximizing code reusability.

Testing: Testing machine learning systems is more complex than testing other software systems. In addition to typical unit and integration testing, you need data validation, trained model quality assessment, and model validation.

Deployment: In an ML system, deployment is not as simple as deploying an offline trained ML model as a predictive service. ML systems may require you to deploy multi-step pipelines to automatically retrain and deploy models.

Through continuous monitoring and feedback mechanisms, MLOps emphasizes constant attention to model performance and data changes to optimize model stability and accuracy. However, the lack of software engineering experience of machine learning team members, as well as the complexity of model deployment and sensitivity to data changes, also pose certain challenges to the practice of MLOps that require further research and resolution.

3. Methodology

3.1. Automation in Model Training

Automating the model training process is essential to simplify development and ensure repeatability. First, automation can dramatically reduce human intervention, improve efficiency, and save time and resources. By using version control systems and automated workflows, teams can track and manage all changes during model training, including data sets, parameter Settings, and algorithm selection. In real-world deployments, large neural networks (DNNS) face challenges due to their huge demands on resources. [8]Traditional DNNS may face hardware limitations, insufficient computing resources, and latency in actual deployments, especially in environments where edge devices or resources are limited. This can be achieved by pruning, quantization, low rank approximation and other techniques. Therefore, the compressed neural network can not only reduce the deployment cost and delay, but also expand the application range of the model in different devices and scenarios, and promote the process of DNN productization.
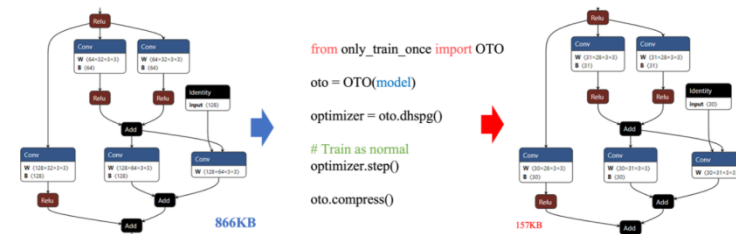

Figure 1. OTOv2 framework based on DNN automated deployment

Pruning is one of the most common DNN compression methods, which aims to reduce redundant structures, slim down DNN models while maintaining model performance. However, existing pruning

methods often point to specific models, specific tasks, and require AI engineers to invest a lot of engineering and time effort to apply these methods to their own tasks. Therefore, the compressed neural network can not only reduce the deployment cost and delay, but also expand the application range of the model in different devices and scenarios, and promote the process of DNN productization.This tracking and management mechanism not only facilitates collaboration and communication among team members, but also ensures the consistent performance of the model in different environments, improving the reliability and maintainability of the model.

3.2. Integration with CI/CD Pipelines

DevSecOps is a cultural approach in which every team and person working on an application considers security throughout its life cycle. It ensures that security is implemented at every stage of the application software development life cycle (SDLC) by embedding the required security checks into CI/CD[9] automation using appropriate tools. But in actual deployments, with vulnerabilities emerging faster than ever before, integrating Dynamic Application security testing (DAST) into continuous integration/continuous deployment (CI/CD) pipelines is a game-changer, helping you consider security at an early stage, find and address security vulnerabilities as early as possible. Rather than wait until they seriously affect users before taking action.
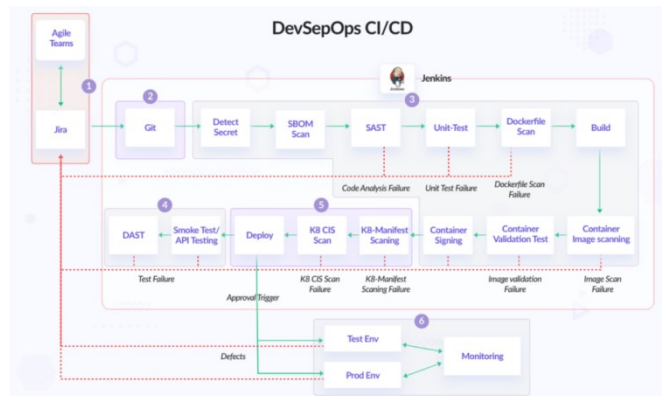

Figure 2. DevSecOps CI/CD pipeline architecture

CI/CD is a way to frequently deliver applications to customers by introducing automation in the application development phase. CI/CD's core concepts are continuous integration, continuous delivery, and continuous deployment.

Specifically (Figure 2), CI/CD enables continuous automation and continuous monitoring throughout the entire life cycle of the application, from the integration and testing phase to delivery and deployment. By using containerization, machine learning workflows can be packaged into containers and deployed in different environments, simplifying the process of deployment and management, and improving portability and reliability.

3.3. Model Deployment and Monitoring

In a large cluster, even routine operations can become variable, including operating system upgrades, security patch application, software package management, and custom configuration of kubelet or containerd. During O&M[10] operations, if the node status is inconsistent due to errors, that is, the configurations of some nodes are inconsistent with the expectations, or even multiple versions of nodes exist at the same time, the next O&M operation will fail and unexpected behaviors of the same service copies may occur on some nodes, resulting in service stability risks.
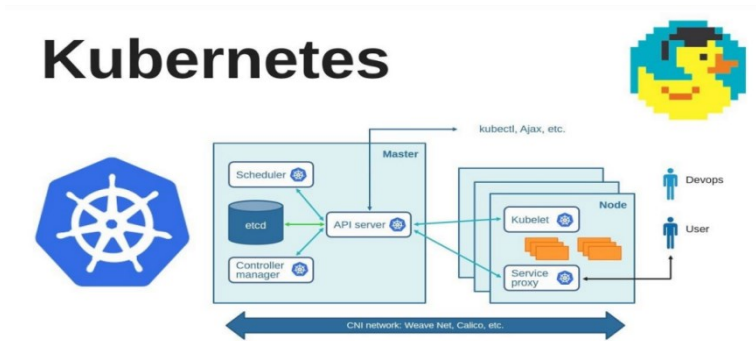
Figure 3. Kubernetes (K8s) automated deployment environment architecture

Large AI models are typically deployed on cloud-native environments such as Kubernetes (K8s)figure 3. With the popularity and maturity of cloud-native technologies, more and more enterprises and developers are choosing Kubernetes as the infrastructure to deploy and manage AI large model services based on the following points:

1. Resource management and Scheduling: [11]Kubernetes provides flexible and efficient resource management and dynamic scheduling capabilities, which are critical for large-scale machine learning models that rely on high-performance computing resources, especially Gpus.

2. Elastic scaling: According to the changes in the demand of the model service, K8s can automatically expand and shrink the number of Pods, so as to better use resources and ensure the high availability of services.

3. Containerized deployment: Packaging the model and its runtime environment through container technologies such as Docker, so that the model can be quickly and consistently deployed in any Kubernetes-enabled cluster.

4. Service orchestration: K8s provides a complete set of service discovery, load balancing, and service governance mechanisms to help build complex microservice architectures, especially for AI applications where multiple services may be required to work together.

5. Model version management and update: With K8s rolling update, canary release and other functions, model versions can be smoothly upgraded or rolled back to reduce operation and maintenance risks.

Cloud-native architectures and Kubernetes have become one of the standard choices for large-scale AI deployment and management, especially when dealing with large-scale, high-performance demanding scenarios. From model training to inference services, K8s provides powerful support.

4. Case Studies and Best Practices

At Netflix, there are hundreds of thousands of workflows and millions of jobs running on multiple layers of the big data platform every day. Given the broad scope and intricate complexity inherent in such distributed large-scale systems, diagnosing and fixing job failures can create a considerable operational burden, even if the failed jobs represent only a small percentage of the total workload. To handle errors efficiently, Netflix has had great success using machine learning to improve the user experience, recommend content, optimize video coding, and make content distribution more efficient. Their MLOps practice is one of the keys to their success.

First, the classification of Netflix's rules-based classifier, which uses machine learning services to predict retry success probability and retry cost, and select the best candidate configuration as a recommendation; Netflix's autofix system integrates rules-based classifiers with machine learning services to optimize retry success probability and retry cost, automatically selecting and applying the best configuration for error resolution. Its key advantages lie in its integrated intelligence, combining deterministic classification from rules-based classifiers with ML-powered recommendations, full automation of the repair process, and multi-objective optimization considering both performance and

cost efficiency.Automatic repair generates recommendations by considering performance (that is, the probability of retry success) and calculating cost efficiency (that is, the monetary cost of running a job) to avoid blindly recommending a configuration that consumes too many resources. For example, for memory configuration errors, it searches for multiple parameters related to memory usage for job execution and recommends a combination of linear combinations that minimize the probability of failure and computational costs.

4.1. Netflix has accumulated many best practices and lessons learned in the practice of MLOps.
These include:

1. Automation and Continuous integration: Netflix emphasizes automation and continuous integration, leveraging the CI/CD pipeline to automate model training, evaluation, and deployment. This automated process increases efficiency, reduces human error, and ensures rapid iteration and updating of models.

2. Containerized deployment: Netflix containerizes models and applications and leverages Kubernetes for deployment and management. Through containerization, they are able to achieve rapid deployment, elastic scaling, and high availability of models, while ensuring consistency and portability of the environment.

3. Real-time monitoring and feedback: Netflix has established a real-time monitoring and feedback mechanism to detect and resolve problems in a timely manner by monitoring model performance, user feedback, and system logs. This continuous monitoring and feedback loop helps to improve model stability and reliability, and to adjust models and services in a timely manner.

4. Refined experiment management: Netflix values experiment management and version control to ensure that each model has clear traceability and repeatability. They utilize advanced experiment management tools and processes to manage model versions, parameters, and results for effective comparison and selection.

Through these best practices and lessons learned, Netflix has not only overcome common challenges in MLOps, but also improved the efficiency and quality of workflows, laying a strong foundation for innovation and success. The successful application of these strategies provides a valuable reference for other organizations to build and optimize their own MLOps practices.

5. Conclusion

The content of the article reveals the challenges faced by machine learning in practical applications and introduces the importance of MLOps as a solution. By automating model training and deployment and integrating into traditional CI/CD pipelines, the complexity and challenges of deploying machine learning models in production environments can be effectively addressed. In addition, the paper emphasizes the importance of continuous monitoring and feedback loops in maintaining model performance and reliability. These methods and tools provide effective solutions for the development, deployment, and management of machine learning models, thus accelerating the model development cycle and improving the quality and performance of the models.

Looking ahead, as machine learning technologies continue to evolve and the range of applications expands, we can expect more innovation and progress. Machine learning not only plays an important role in improving business efficiency, promoting innovation in the advertising industry, and improving medical technology, but also brings great potential and benefits to human society. By applying machine learning technology more widely, we can enable smarter and more efficient decisions and services, contributing to the sustainable development of society. The development of artificial intelligence will bring more convenience and well-being to mankind, and we should continue to be committed to promoting the innovation of machine learning technology to better meet human needs and achieve social progress and development.

References

[1] Kreuzberger, Dominik, Niklas Kühl, and Sebastian Hirschl. "Machine learning operations (mlops): Overview, definition, and architecture." IEEE access (2023).

[2] Ruf, P., Madan, M., Reich, C., & Ould-Abdeslam, D. (2021). Demystifying mlops and presenting a recipe for the selection of open-source tools. Applied Sciences, 11(19), 8861.

[3] Choudhury, M., Li, G., Li, J., Zhao, K., Dong, M., & Harfoush, K. (2021, September). Power Efficiency in Communication Networks with Power-Proportional Devices. In 2021 IEEE Symposium on Computers and Communications (ISCC) (pp. 1-6). IEEE.

[4] Srivastava, S., Huang, C., Fan, W., & Yao, Z. (2023). Instance Needs More Care: Rewriting Prompts for Instances Yields Better Zero-Shot Performance. arXiv preprint arXiv:2310.02107.

[5] Luksa, Marko. Kubernetes in action. Simon and Schuster, 2017.

[6] Ma, Haowei. "Automatic positioning system of medical service robot based on binocular vision." 2021 3rd International Symposium on Robotics & Intelligent Manufacturing Technology (ISRIMT). IEEE, 2021.

[7] Sun, Y., Cui, Y., Hu, J., & Jia, W. (2018). Relation classification using coarse and fine-grained networks with SDP supervised key words selection. In Knowledge Science, Engineering and Management: 11th International Conference, KSEM 2018, Changchun, China, August 17–19, 2018, Proceedings, Part I 11 (pp. 514-522). Springer International Publishing.

[8] Mahesh, Batta. "Machine learning algorithms-a review." International Journal of Science and Research (IJSR).[Internet] 9.1 (2020): 381-386.

[9] Wurster, Michael, et al. "The essential deployment metamodel: a systematic review of deployment automation technologies." SICS Software-Intensive Cyber-Physical Systems 35 (2020): 63-75.

[10] Lu, Q., Xie, X., Parlikad, A. K., & Schooling, J. M. (2020). Digital twin-enabled anomaly detection for built asset monitoring in operation and maintenance. Automation in Construction, 118, 103277.

[11] Dearle, Alan. "Software deployment, past, present and future." Future of Software Engineering (FOSE'07). IEEE, 2007.