Deep Science Publishing, 2025
https://doi.org/10.70593/978-93-7185-753-6_4

Chapter 4: MLOps and Lifecycle Management

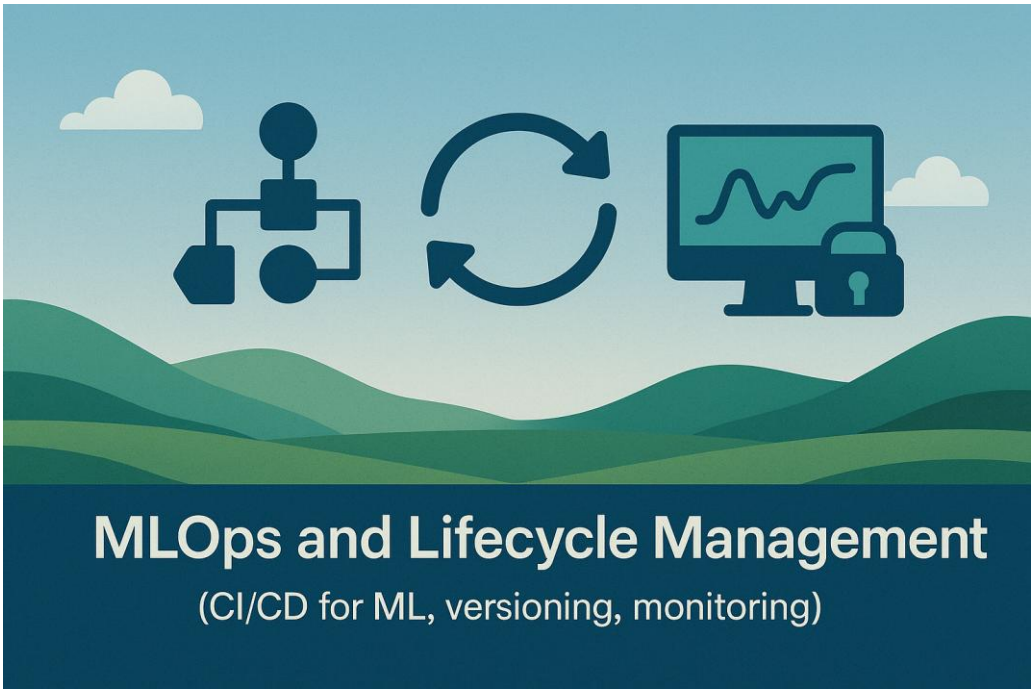
Swarup Panda
SRM Institute of Science and Technology, Kattankulathur, Tamil Nadu, India

1. Introduction to MLOps

Companies today use increasingly more machine learning in their products and services. From recommendation engines to computer vision to natural language processing, the usage of machine learning is exploding. But while the adoption of machine learning continues to grow, deploying and maintaining machine learning systems in production remains painfully challenging [1-3]. Respondents to industry surveys report that deploying machine learning is harder than any other part of the software development lifecycle. The gap between niche research and core business functions threatens to hollow out investment in new ideas: a significant amount is spent on machine learning annually, but a large percentage is not delivering any value [2,4,5]. Firms are spending huge sums on machine learning, but the vast majority of projects are failing. Organizations that successfully implement strong systems to support the MLOps function should be rewarded with vectors of highly leveraged development teams, and a reliable return on their investment in large scale machine learning studies.

The challenges lie on the operations side of the equation. Operationalizing machine learning is different from operationalizing regular software in several profound ways. A machine learning model will need retraining based on changing data distributions — a problem exacerbated by the pace of change of many of the world's most data-dependent businesses, such as finance, retail, and travel. Machine learning is often user-facing, which introduces concerns from design and performance perspectives that require continuous commitment from

the engineering organization. Manufacturing and deploying a piece of machine learning code is not simply an engineering task. Doing this well requires collaboration with data science, engineering, and design — in particular, mobilizing data scientists to work on real products while still allowing them the autonomy they need to find new models and techniques. Why use one framework over another?



2. Understanding Machine Learning Lifecycle

New technology is constantly being developed to offer automatic assistance for machine learning model development. The Evergreen tool allows enabling or managing different stages of the machine learning lifecycle. It's important to clarify what stages these are and what needs to be done to manage machine learning workloads effectively. There's no such data science project which passes through all proposed stages and approaches, and certainly not with the same importance [6-8]. Some workflows pass through the phases lightly and some workflows delegate the majority of work to specific tools. The goal is not to have a general guide detailing the best possible design for all projects, but to discuss machine learning lifecycle management in the hope of designing better tools.

One of the first points to elaborate on in ML system management concerns the organization of project data. Several machine learning tools manage projects by an explicit data organization based on file system structure or database record links. Other tools have no explicit project structure, managing different types of external files in the same workspace. Tools defining a project structure organize physically or virtually the data but doing little to additional restrictive modeling or resource control. Projects can be folders which only allow a specific data structure and types for files or database collections which have obligations or additional access and control features [9,10]. Data organization affects how users retrieve and understand the underlying machine learning process. It also facilitates or makes more difficult the application of data science best practices, increasing or decreasing the risk of unpleasant surprises.

3. Continuous Integration and Continuous Deployment (CI/CD) for ML

3. Continuous Integration and Continuous Deployment (CI/CD) for ML

3.1. Overview of CI/CD in Machine Learning

Continuous Integration and Continuous Deployment (CI/CD) pipelines have become essential for the success of Machine Learning (ML) systems. By automating not only testing but the risky process of deployment, well-designed CI/CD pipelines allow companies to publish AI products in a controlled and systematic manner while keeping the risk associated with executing a ML model under constraints [11-13]. This is the case, for example, when the trained model makes use of an online learning system, re-training every time some new data becomes available or at specified periods. Ideally, CI/CD pipelines reduce the time interval between when a model is trained and when it makes decisions and infers on users.

However, ML is a special kind of software being that the software consists of a model that makes predictions on a dataset. The inherent variability present in AI models complicates significantly the design and implementation of CI/CD pipelines for ML. Indeed, datasets form probability distributions capturing the underlying generative process of the data. These distributions are changed naturally over time, in coordination and reaction to economic, social, and

informational events [2,14-17]. Frequent changes in the underlying distribution cause the variable behavior of AI products and services. Models that work well on the training dataset may become obsolete and perform poorly on the deployment dataset, also referred to as drift and, in practice, deteriorate the predictions made to users or clients. These events, however, are unpredictable. As a result, keeping the underlying distributions in check is an important and difficult aspect of operating successful AI deployments.

3.1. Overview of CI/CD in Machine Learning

Continuous integration (CI) and continuous deployment (CD) are common practices in modern software engineering that enable more rapid delivery of features and fixes, while reducing errors in production software [9,18-21]. A CI/CD pipeline automates the process of deploying updates to code, as well as any necessary actions associated with those changes. CI/CD works very well for many common software use cases, but the basic principles of these tools, and the experience and confidence they give developers about the quality and performance of new code changes, means that it's natural to try to extend these principles into the lifecycle management of other types of code-intensive systems. In particular, the rapid innovation and experimental research direction inherent in machine learning makes it attractive to see if CI/CD can apply to ML. However, current ML systems can be more complex, with a larger variety of interhandled processing steps communicated for diverse heterogeneous data types, and a broader class of model evaluation metrics used to judge and monitor the outcome of the various steps. Early attempts to directly adopt CI/CD for ML have resulted in tears.

The disparate, heterogeneous nature of ML systems means that implementing and using a CI/CD pipeline requires experience, understanding, and effort [22,23]. There is no single standard for building a CI/CD system for ML; there are many choices, and the core CI/CD system must be customized for each target ML application. Thus, the system is specific to an organization, and the people in the organization building and deploying the ML application. So, while the principles of CI/CD still apply to ML, it is both more complex to do a CI/CD rollout for ML than for standard software, and the results of the effort have a larger payback, with reduced deployment times along with increased application accuracy and reliability.

3.2. Setting Up CI/CD Pipelines

Creating CI/CD pipelines for ML requires some specific considerations. We will illustrate a simple example where a few steps are outlined [24-26]. First, version the base environment for your ML project: languages and packages, as well as configuration files, datasets, and code are the typical artifacts that change. By using these artifacts, you'll be able to replay any experiment later. Use your ML code repositories with ML code incorporated in them. Next, containerize your ML environment: images that combine the base environment, configuration files, and installation scripts for datasets and external libraries should be defined. These images should be tagged to reflect their version for any particular experiment. Then host your containers: publish these images to your internal enterprise repository to improve security. Finally, trigger the model training: machine controllers, event-based triggers, orchestrators are often used for that. Automated tests should validate data, environment, code, hyperparameters, and model quality.

These components allow you to schedule training workflows to execute these training and testing phases during your project lifecycle. You might have to tune the completion criteria for these jobs to accurately reflect the model training and testing costs. Centralizing your computation resources is an important cost reduction practice. This area is evolving rapidly.

3.3. Tools and Technologies for CI/CD

Many of the MLOps frameworks and tools that are being adopted already were implemented for CI/CD of software in general. The main difference is the unique requirements of machine learning systems and their lifecycle. These specialized tools encapsulate best practices for CI/CD of ML. The following sections briefly summarize the key tools and capabilities related to CI/CD.

Continuous integration (CI) automates the processes of testing changes and ensuring that a new model meets both functional and non-functional requirements. CI enhances testing infrastructure as well as its capabilities. Non-functional testing of models and data has received less attention than functional testing in CI for ML, although it is addressed to some extent by various tools. However, tools for non-functional testing of ML models in production are less mature and tested than the vast software industry experience of functional testing CI tools.

Continuous Delivery (CD) automates the creation of ML deployment artifacts. Specialized workflows manage these pipelines and allow for configuration of multiple target environments [27,28]. CD provides iterative and versioned deployment of ML. These capabilities are embodied in various tools. These aspects are also addressed by general-purpose pipeline frameworks. The workflow orchestration capabilities have evolved from general-purpose workflow orchestration frameworks to serve machine learning users' specific needs. These services derive their capabilities from previous work on dedicated frameworks.

3.4. Challenges in CI/CD for ML

The CI/CD practices in MLOps go beyond just creating a continuous integration or delivery pipeline. While it is relatively simple to trigger a build and a set of tests to run on each commit, this doesn't mean the code is ready for production usage. In ML, a model provides useful predictions only if it is trained correctly and used to make inference on data which is similar in distribution to data the model was trained on. Therefore, well-defined interfaces are needed between various components for an ML pipeline. This ensures that all teams involved in building an ML solution have a mutual understanding of how to interact with each other's collectively built artifacts. Additionally, it's important to version ML models and training surroundings and keep track of data distributions while iterating on model improvements. While all these requirements are met in non-ML software development via various processes and policies defined within organizations, the same policies often do not get translated to MLOps CI/CD pipelines [19,29-31].

Besides, MLOps and ML CI/CD pipelines still do not have a universal set of tools widely used and accepted by the community for solving pitfalls uniquely present in ML development. As a result, experiments in different organizations are different due to the diversity of domain, resources, and use cases. Custom tools and policies therefore still need to play a part in operations pipelines, but preferring an open-source model that encourages collaboration between organizations may be a better approach. Moreover, the constant exploration-exploitation trade-off present in ML can lead to unpredictability in performance if enough safeguards are not used judiciously [32,33]. Non-ML pipelines also struggle with certain facets of software maintenance. In particular, these are managing code quality, technical debt, and testing that fraudulently steals user feedback or is dangerous to engage with actors when the software is predicting.

4. Versioning in Machine Learning

Versioning in machine learning is a vital aspect of the MLOps lifecycle that ensures the audibility, availability and an easy rollback to the previous state of a model. Versioning here does not only mean versioning a model artifact, but also other model-related assets such as the data, commands and the environment needed to reproduce the trained model [34-36]. In a machine learning project, a model is considered the source of truth as it is supposed to increase the trustworthiness of the overall ML system. A model's utility changes with time and space, i.e., the input domain and the environment that the ML pipeline operates under changes. For all these cases, versioning is a vital aspect of the MLOps lifecycle.

Versioning a model is important considering the various stages of the Model Lifecycle Management, such as Data Management, Model Development, Model Evaluation, Model Serving, and Model Deployment. Each stage requires artifacts, models, and configurations to be tracked. While solutions exist for tracking these other aspects, the model itself simply gets dropped on a centralized file store with a name reflecting the date and time of when it was produced. This practice is dangerous as the unscripted model is a black box and it is necessary to know how to interrogate it to find additional information. Keeping this process manual is completely against the basic premise of MLOps, which suggests automation to make it CI/CD enabled.

Some existing Model Version Control implementations allow the user to create a model and upload it to a versioning system. These systems then keep track of a user's interactions on the model – whether a model has been replaced, deployed in a production environment, and so on. Model version control allows organizations to more rapidly develop ML solutions, track the evolution of models, and deploy models that are more explainable. Version control and source code management systems have similar functions – script, parameterized and versioned model training [37-40]. However, version control serves a different purpose for ML engineers and teams. While source code management serves as a standard library of model development, version control seeks to maintain interactions with a model such as deployment status for explainability. Both version control paradigms integrate easily with CI/CD pipelines.

4.1. Importance of Versioning Models

Understanding data versioning helps contextualize model versioning's requirements; specifically, the peculiarities of machine learning as an infrastructure that requires exotic digital assets as input – models are static files and are digital assets. Models learn from data and thus are supposed to be improving over time [41-43]. In practice, however, many reasons can warrant model versioning: from self-serving necessity of promotional aspect of creating better models; to collaboration in teams of practitioners, either to serve the same use case or techniques like ensembling; to considerations of attribution and oversight: such as regulatory obligations from financial services or medical domains; forensic investigations on malicious uses; or any scenario demanding explanation of outputs based on model behavior.

In addition to flattening diversity and complexity of models into specific files, model versioning has to tackle some extra challenges. First, these digital assets are very different from static data files or other software files. They are outputs of proprietary procedures unique to machine learning, typically leveraging third party frameworks or libraries with varying quality and transparency; they require tuning flexible hyper-parameters with hard-to-model effects; and they learn not immediately, but once during production. This makes practitioners reliant on third-parties, with unknown use cases or horizons: particularly with respect to transfer learning tasks, supplying models that serve as starting point for other use cases.

4.2. Techniques for Versioning

Given their vast variety and differences, there are different types of models and ways to create them, which results in many kinds of model versioning approaches. We categorize these ways into two groups: backward-compatible versioning and non-backward-compatible versioning. The first category is basically the same as how software applications are versioned, where every version released is backward-compatible. The second category depends on model types and usage at deployment time. Some models require special usage considerations to ensure that different versions of that model can be called at the same time.

Versioning can be applied at different levels in a pipeline. At the core of a model, it is usually a custom operation that embeds how to create predictions. This is the concept known as the “predictor.” This predictor can be called in many ways in

a pipeline, be it handmade code in a micro-service architecture or hidden inside commercial products. Developers spend effort creating special software libraries to guarantee that any code calling functions from this library can run prediction without knowing how to interact with the actual code implementing this predictor. Since any change in that implementation can generate different micro-level predictions, it is expected that any change to the model would trigger a library change and possibly a new key.

4.3. Best Practices for Model Version Control

Below, I outline several best practices for model version control to help make it easier and simpler for development teams to manage machine learning model development and deployment pipelines. Most of these best practices come directly from industry experience managing machine learning models in production.

1. Track the Metadata that Matters: Maintain proper metadata when versioning models. At a minimum, you should consider storing the following artifacts in association with the model: the model code details, hyperparameter settings, the version of the libraries with which the model was built and trained, the training dataset, the evaluation metrics, and model performance against those metrics along with a minimum threshold that the model must pass to successfully be validated, and the person that trained the model. Otherwise, operations teams will be at risk of not "knowing what they don't know," and a mysterious model could easily slip into production without any proper auditing, ethical, or legal guidelines being followed.

2. Model Metadata as Code: If possible, place all model metadata in code. This way, it can be tracked, versioned, and managed the same way any other codefile in the project is managed.

3. Track the Journey of a Model: There are different types of model versioning - each version can be anything from a modification in a single hyperparameter to an entirely differently designed model; a model can even have no code changes but simply a re-training on a different dataset and, thus, the model architecture may not change much. The key point here is to understand the journey of a model at all stages in its life cycle.

4. Use a Centralized Repository for your Models: Model versioning must occur in a centralized location accessible to all engineering teams responsible for

particular models. It should be part of a larger tooling ecosystem that supports other aspects of model lifecycle management as well as tooling for the development and deployment pipelines. This is also a guideline for practice as it mandates automated workflows such as policy-based auto archiving, access control, comparison, and lineage visualization for models and their artifacts in addition to simpler storage and tracking capabilities that are important.

5. Monitoring Machine Learning Models

In traditional software development lifecycle, production systems are closely monitored for any performance degradation. Typical monitoring workflow detects anomalies in the execution, logs, and results, and triggers alerts that can go to the application team to fix any possible issue. Since Machine Learning based applications are different and introduce a black box model instead of rule-based software logic, the need for monitoring is even higher for ML based applications due to the following reasons:

Data Drift: Streaming data into Model inference can change the distribution of input and output data. The Model predictions are as good as the data fed to the Model. Hence, data drift can lead to bad model performance over time.

Concept Drift: Changing relationships between input and output data can lead to concept drift. For example, in fraud detection, increase in fraud for a payment mode can lead to relationship where the Fraud = Payment Mode. If an organization introduces more payment modes, then the relationship is different, and concept drift occurs.

Different Failure Mode: A traditional software application may fail with exception scenarios. Machine Learning Models have different types of failure modes, wherein the Model is not throwing errors or exceptions, however it is acting on unverified inputs or predicting outputs that are not realistic. For example, a customer churn Model predicting previously loyal customers to be frauds would be a cause for concern.

Monitoring Machine learning Models is not straightforward and involves different type of metrics based on the Business domain, Model usage patterns and Infrastructure. Let's discuss the different types of metrics in more detail.

5.1. Need for Monitoring in ML

Monitoring is an important part of any machine learning life cycle. You cannot simply deploy a model and forget about it. If you want your deployed model to run optimally, you need to continuously monitor its drift. Why is monitoring important? A model may be highly accurate at deployment and guarantee good performance. However, over time, the underlying distribution of data can change, which can impact model performance. This type of degradation is commonly referred to as drift.

Why does drift happen? The data used to train the model is a snapshot of data taken at a single point in time. An assumption of the modeling process is that that snapshot is representative of the real world over time. For high-stakes use cases such as fraud detection and clinical diagnosis, this assumption does not hold beyond a short time and models typically do not perform well over time. In high-stakes use cases, data changes constantly, and predictions degrade. Therefore, monitoring the model as data drifts is crucial. But drift can occur in multiple ways. A change in the data over time is termed as data drift or covariate shift. A change in the relationship between covariates and outcomes is termed as concept drift. This can happen even if there is no data shift.

5.2. Key Metrics for Model Performance

After going through the need for monitoring ML models, the next important question is about the metrics that can be used to monitor the models. Although several ML lifecycle management tools offer default alerts and checks, it is left to the data scientists and ML ops engineers to select the specific features and thresholds that need to be checked for monitoring. These are model specific and will depend on the model architecture being used in addition to the business problem at hand. There are several factors affecting the ML performance: the intended usage of the model, available external checks, degree of dependency needed with input data, and distribution of its output values. Below we describe some of the most common metrics that should be checked.

Model Accuracy. Usually, measuring the model accuracy is the most used method for comparing model predictions with actual results or labels. It can be defined for the case of regression as 1 – MAE or 1 – MSE, where MAE or MSE compares the predicted values with actual values. Since this similarity computation is usually made for the entire test set, it can be compared to input data at a later point in time to check for thresholds for drift detection. A note of

caution here is to ensure the timing of the prediction and actual label assignment is matched perfectly to avoid data leakage, as prediction labels can usually be stringently held via derivation rules over the dependent variable.

Data Drift Detection. Model performance drift or prediction error drift is an indicator of the performance of a model exposed to real-world data. Model monitoring should track the data stream over time to understand the underlying pattern and distribution of the data to capture the expected underlying pattern. Data drift is an indicator for ML practitioners, where prediction quality monitoring should be the most focus. The performance of traditional ML or statistical models can be monitored with their intended purpose, but modern supervised ML driven by neural networks have little model interpretability or explainability. An increasing prediction error drift information should lead to conducting correlation tests to understand the relationship between input data and performance drift. In case their correlation be significant, then the ML model should be retrained, repurposed, or revisited to achieve explainability.

5.3. Tools for Monitoring ML Models

We have discussed the importance of monitoring the performance of the ML models and how to measure their performance. Now, we will touch upon the tools that are available for monitoring the ML models. These tools are used for different types of monitoring, including data, model performance, data and model observability. Data Observability tools monitor the data pipelines, detect the anomalies and help alert in case of any issues in the data, including missing data, change in distribution of input features or target variable etc. The Model Performance Monitoring tools help check the performance metrics of a deployed ML model against the business defined thresholds and alert in case of any performance degradation. The Data and Model Observability tools are a combination of the above two capabilities. They provide a single view to monitor the data pipelines and deployed models.

Here are some of the data observability tools that are available: Dagshub, Datakin, Databand, Datafold, Datastreams, GitHub, InfluxDB, Monte Carlo, PipeRite, PipelineWise, Puffin, Soda, Unpuzzle, etlworks, etleap, futureX, FiveTran, Grouparoo, Hevo Data, Keboola, Keen.io, Merge, Matillion. Some of the tools that help monitor model performance are: Albacore, Aporia, App Insight, Arize, Bespoke Metrics, Evidently, Fiddler, Giskard, Great Expectations, Human Loop, Mistral, NannyML, Neurala, MLflow, Npml, Onto, Pymetrics,

Shadow Mode, Superwise, Tecton, Verta, Weights and Biases, and ClearML. Here are some of the Data and Model observability tools that can be used: Arize, Clips Analytics, Datadog, DoiT, Explorium, Immuta, Metaplane, Observability, Ponder Policy, Sisu, Sumo Logic, and StatisticalML.

5.4. Handling Model Drift

In high-stakes environments, drift detection is necessary to ensure that models maintain their desired level of performance. Drift mitigation methods can be applied when drift is detected or can be added into the pipeline before a drift is detected, such as training based on labeled data streams or retraining periodic batch processing. In urgent situations, a fully retrained model may need to be tapped immediately, but for the majority of non-real-time predictions, drift correction via retraining on the most relevant data is warranted. Several solutions are available to build and monitor such adaptive frameworks to help mitigate model drift and data integrity problems.

The first choice to handle model drift and data integrity issues is to create a fully automated retraining pipeline where models are retrained incrementally, or periodically as data becomes available without human intervention. In real-time systems, the models can be updated incrementally. In such solutions, solutions based on online learning are the preferred approach to updating existing models via an additional small optimal incremental update.

Adaptive semi-supervised models that incrementally retrain library models in case modeling performance is below a drift detection threshold have been proposed. Decision trees and hierarchical relevance tree models have also been proposed for incrementally learning and adapting for different classes without retraining on previously seen data or the need for a re-initialization effort. The periodic retraining setup is more common in periodic batch processing pipelines. In the majority of these solutions, the monitored models are retrained on partial input data that fall within a recency threshold of the current time.

6. Integration of MLOps with DevOps

MLOps can be seen as extending the principles of DevOps into the realm of machine learning and AI development. Specifically, the integration of MLOps with DevOps can be understood as a natural progression of a collaborative culture

where developers and IT operations teams continuously work together interleaving multiple steps involved in the continuous delivery of applications. In this regard, this section seeks to first elaborate on the similarities and differences between MLOps and DevOps and then present the advantages of integrating both.

MLOps extends the DevOps principles of agile, collaboration, automation, monitoring, and continuous improving, into the realms of ML development and deployment. MLOps provides a set of AI lifecycle management capabilities to address the additional complexity and challenges of moving ML systems into production and relying upon them for supporting business operations and decisions. An initial important difference between MLOps and DevOps is that in DevOps application releases are discrete and require code level changes, while in MLOps, all models need to be continuously monitored across a number of evaluation metrics, in order to determine drift or model degradation performance thresholds for automated retraining or redevelopment execution in any form of cycle and frequency.

On the other hand, both worlds also look to use common technologies. In particular, in the first step of the CI pipeline, MLOps aims to integrate model training, testing, and validation, at the code and model level into the CI process. AutoML capabilities such as hyperparameter scaling, and bias and fairness checking can further promote and facilitate this integration. Furthermore, integration pipelines can then be spawned for the CD step executions using containers or with artifact storage enabling monitoring observability and auditability controls.

6.1. Comparing MLOps and DevOps

Probably the first question regarding MLOps is why do we need this concept instead of using our already established DevOps processes? Or better, what is the difference between MLOps and DevOps? To answer this question we need to go once more to the conceptual basis that defines MLOps. The main goal of both concepts is to automate the development pipeline of the products: for DevOps the products are primarily software, while for MLOps the products are ML models, but also the services or products that use these models. There are many differences that change the way how we can provision tools and techniques for production and post-implementation of products, which lead us to understand that

MLOps cannot yet be totally addressed within the notion of DevOps in the same way that we understand it for software projects.

To be more precise, these two activities focus on the same inputs and business objectives: having delivers continuously done is necessary for both tasks. But in the case of MLOps the inputs are models and not just code. Before being put into production, ML models go through pre-production activities such as data pipeline design, data collection, exploitation, and exploratory analysis, feature engineering, hyper-parameters optimization, and analysis, selection, and training of the model itself. Through these activities, a model can go through numerous versions, which are selected based on statistical and business metrics, in order to be deployed in the production environment. And in the production phase, MLOps activities are not just confined to ensuring that ML code is continually supplied for updated. In fact, the ML models are not static entities while in the production environment. They demand a ceaseless cycle of evaluation against data defined metrics in order to understand when the model's concept in production has drifted, indicating the need for a model update.

6.2. Benefits of Integration

Although both DevOps and MLOps are intended to enhance productivity, minimize time to market, and increase system reliability, to date integration has only occurred at the technological level, not at the organizational level. In other words, a company may have IT and AI teams that both use compatible tools for CI/CD, but are still independently managed. A cultural or leadership agreement or bond between the departments is not yet visible in most organizations. If this gap were bridged, the benefits of technological integration could be amplified. Businesses must learn that the problems DevOps teams face in delivering AI assets and adapting them to production environments differ from those that the teams involved in ML share. For one thing, the frequency of changes is different – IT applications are regularly modified, whereas, in ML applications, collation is much rarer. AI team efforts are more disparate than software team responsibilities, even if the teams work within the same framework. AI systems are called on to solve more complex tasks and support a larger range of use cases than other technological tools. IT microservices are more interconnected. What is more, while the only major system failure that may occur with DevOps is that the service is not working, AI systems can create irrevocable damage if, for instance, they recommend giving an overdose of fentanyl for a patient in need of morphine to relieve pain for terminal illness.

7. Case Studies in MLOps Implementation

In the previous sections of this paper, we have outlined the need for ML teams to take a software engineering approach and implement a cycle of continuous monitoring, experimentation, and learning while serving ML systems in production. In this section, we provide more detail about these processes through a set of case studies gathered from multiple domains and teams at multiple levels of maturity. Sharing the stories of others helps your MLOps team both to share the benefits you hope to provide, as well as the challenges other companies have faced on their MLOps journeys. While brief, these case studies describe many different MLOps systems and processes, including:

- How various teams have adopted best practices of DevOps, DataOps, and ModelOps and rolled those practices into their daily workflow. - The importance of human collaboration and communication when deploying ML into production. - Examples of successful MLOps deployments and ended failures along the way. - Various levels of MLOps expertise, including those teams who build complete frameworks and MLOps services utilized by other teams, as well as those who are just beginning their MLOps journey and offering MLOps related support to other teams in their organization.

Collectively, we hope these case studies give you a taste of what other teams have dealt with successfully, and the benefits and lessons they have gleaned from those journeys. The efforts outlined are reflections of what we have seen worked at scale and want to help teams beginning the MLOps journey avoid the pitfalls along the way.

7.1. Successful MLOps Deployments

MLOps is actually just a natural extension of decades' worth of software development and services work, and most of the internals are built on top of the decades' worth of learning and investing done in enterprise-grade software. The lifecycle management aspects of machine learning may be new in some ways, but the code-centric DevOps and SRE automation patterns are very mature. Even the AI-specific parts of MLOps are derivative of existing work in DevOps, Chaos Engineering, safety, security, and the other aspects of code-centric tech.

At least for production systems in the world today, much of what many companies think of as "MLOps" are capabilities already deployed in the real world at scale, thanks to the ongoing work on the part of both developers as well

as tooling developers. Core tenets of MLOps, such as the automation of model evaluation, validation, verification and monitoring for performance and degradation is the main focus of software in production today, often supplemented with tried and true code-centric production practices. We already have feedback loops.

This is not to say that MLOps doesn't need more work. Absolutely – as any developer in this space will attest to, there's far more space for capabilities that aren't fully fleshed out, or at least have not been productized and made accessible for enterprise use. True cost, performance and quality, especially in high stake/cost industries, are nascent right now, especially with capabilities such as drift detection and monitoring that tie natural variance detection approaches to active governance models of machine learning pipelines.

7.2. Lessons Learned from Failures

Both Kwan and MLOps.org focus on the lessons learned from predictive analytics projects which had failed to provide value due to the lack of formal operationalization. They stimulate projects in which value creation data products have been deployed but are not adequately managed in production. The lessons learned are reported as guidelines which are considered necessary to make analytics deployments more efficient and effective in practice. These guidelines are widely echoed within current literature as important factors to avoid failure in CS 2. By illustrating the history of common projects in which an algorithm was deployed for internal stakeholders in the company, the guidelines describe principles of operation and product thinking to be followed closely.

The lessons highly relate to best practices for professional software products, oriented to internal users. The PO values from the product owner perspective, strategic alignment with the organizational business goals from the organizational development perspective, the PO and POA from the product strategy perspective, the PDP from the stakeholder perspective, and the DMP and EDP from the decision model perspective. The importance of the POC, or prototype opportunity cycle, is to highlight that development periodicity should be tempered with learning orientation. Additionally, exercises and examples are used from software practice. Using these practice and theory combinations, operational principles are introduced, specifically for decision products, to improve practice.

Learning is very important in business decision making. However, it also introduces complexities in the lesson learning. The partners may use different words, or even language, for operations, tactics, and strategical issues. The recommendation structure is also different. In MLOps, the company is solely responsible. In the data engineering decision model triplet, the company is ridiculed if the systems fail. For practitioner-oriented decision products focused on adding value to an organization, the product is never considered finished. The ownership of a product is important until it erupts as business as usual, triggering more formal production management.

8. Future Trends in MLOps and Lifecycle Management

Machine Learning is thriving. Tasks that were considered prohibitive a few years ago, or that were not yet feasible, have now been solved. Emerging technologies such as large language models, diffusion models, foundation models, and autonomy have shown what is possible with Machine Learning. As these technologies grow into new disciplines, such as AI for Drug Discovery, they will proliferate and grow in sophistication, and new use cases and business models will emerge in the next decade. MLOps software will need to evolve and grow up with them. As the MLOps toolchain matures, it takes more of its responsibility for the ML team, which becomes smaller. At the same time, MLOps tools must align with the business strategy, thereby enabling organizations to accomplish their broader AI objectives. Yet, while the MLOps journey can accelerate an organization's AI strategy execution, it cannot replace the capabilities of human ingenuity and expertise.

Success in MLOps requires a plan. Defining the right strategy at the outset can eliminate many headaches down the road and can mean the difference between becoming an AI winner or a laggard. In the next decade, an increasing number of organizations will find themselves in the AI "winners" group, out-competing their peers, internalizing a cycle of continuous improvement through investment in AI and MLOps infrastructure and a larger AI talent pool through the expansion of educational infrastructures. MLOps will become embedded in investment banking, transportation, logistics, or manufacturing, to name just a few sectors,

and function as drivers of the return on investment from their AI-related expenditures.

Strategically, MLOps represents a major milestone. The boundaries of MLOps are the boundaries of the business strategy. What happens outside of these boundaries can have a huge influence on what happens inside the boundaries, but is outside of the influence of the MLOps function per se. Thus, MLOps will have fewer dependencies on other functions in the organization, as the latter increasingly pick up the activities that are required for responsible AI and AI governance and hence provide impetus for the development of the associated solutions.

8.1. Emerging Technologies

In this chapter, we present what we consider the most important aspects for future MLOps and Lifecycle Management (LCM) trends among many open technologies and current hot topics in industry and academia [28,44-47]. This is, of course, a subjective view that will necessarily miss important considerations, but it can serve as a starting point for various companies and research institutions that plan to work in the MLOps and LCM areas. Some of these considerations previously appeared in a conference.

Machine Learning Operations, or MLOps for short, is the new DevOps. It makes sense to take the DevOps concept as a metaphor because it shares many principles that were successfully applied to software development companies, and should work for ML and AI. That being said, AI is not just another software product; there are important other dimensions to be considered [6,48]. Basically, there are internal and external changes that companies and researchers should take into consideration. We point out some of these dimensions.

The world is constantly changing. The same old strategy of collecting a set of data, building a model around this set of data, and deploying it to prediction for months or years is still successful to many verticals -- but it is becoming increasingly risky. In some situations, businessmen are even wiser to build a monitor or an analyzer of prediction problems to verify the actual behavior of data and models used to product predictions. If the actual predictions go to a certain “too predictable” area – or say, similar to hazard prediction, the understandable best strategy is to remove it from the deployed model, and from the deployed service.

8.2. Predictions for the Next Decade

Machine learning is now a central part of much of tech innovation. Today's LLMs are vehicles for science and technology, not just an amusement or a passable product. The next decade will see machine learning increasingly embedded within both existing and new products and services, with a focus on new advanced capabilities. It will be informing decisions across every industry and every company. It will, of course, help create the new software that runs on our phones and computers. It will help us with the hard parts of writing — inspiration, structure, and clarity — even as we write less directionally and leave much of the work to the model. At first, tools will augment writers. Over a decade, we will write less text while communicating more, better, refined by near-constant model assistance. In addition to helping us work, machine learning will also help us move. Our cars will be constantly telling us where someone is doing an ad-hoc test drive or how much people are willing to pay. For other tasks, we'll be corrected actively by the models in our smartphones and watches.

The LLMs of the near future will help us process text at rates unimaginable today. The fun text games of today will be on the outer edge of what they can do, much as simple adders are on the edge of usefulness — by a long shot — of a silicon chip, or how routing an aircraft is on the edge of what, in the late 1940s, you could expect from vacuum tubes. The LLMs of the next decade will have much-improved inner products, and will be accompanied by a wave of technology designed to shrink them as much as possible. They will be ubiquitous, overwhelming in performance, and eventually easy to use, with technology designed to help ordinary people choose the right words to express what they want to say.

9. Conclusion

Organizations are eager to leverage state-of-the-art ML and Advanced Analytics technologies to enhance decision-making, enable innovation and increase business value. However, productionizing and scaling ML is still hindered by many challenges on the way from prototype to large-scale business impact, including technology infrastructure and orchestration, operationalization process, ecosystem capabilities and roles, as well as governance, monitoring and sustenance mechanisms. There is a need for consulting frameworks to help

organizations assess the current state, challenges and opportunities to enhance the MLOps and lifecycle management process.

Through a MLOps Lifecycle Management framework that emphasizes on 6 key pillars, namely Development, Data, Governance, Model Lifecycle Management, Operations and Teaming alongside core activities, such as Build, Monitor, Governance and Automate, we discuss key activities and processes that organizations would need to consider from a consulting standpoint to effectively innovate leveraging ML. To boost time-to-value of ML, organizations need to invest in MLOps core capabilities, such as technology infrastructure and tooling, organizational capabilities and roles, ML solutions packaging, development process and governance, ML strategy and innovation agenda, as well as model health monitoring and sustenance, along with associated processes and practices.

References

[1] Liang P, Song B, Zhan X, Chen Z, Yuan J. Automating the training and deployment of models in MLOps by integrating systems with machine learning. arXiv preprint arXiv:2405.09819. 2024 May 16.

[2] Kreuzberger D, Kühl N, Hirschl S. Machine learning operations (mlops): Overview, definition, and architecture. IEEE access. 2023 Mar 27;11:31866-79.

[3] Liang P, Song B, Zhan X, Chen Z, Yuan J. Automating the training and deployment of models in MLOps by integrating systems with machine learning. arXiv preprint arXiv:2405.09819. 2024 May 16.

[4] Bano S, Tonellotto N, Cassarà P, Gotta A. Artificial intelligence of things at the edge: Scalable and efficient distributed learning for massive scenarios. Computer Communications. 2023 May 1;205:45-57.

[5] Mishra A. Scalable AI and Design Patterns: Design, Develop, and Deploy Scalable AI Solutions. Springer Nature; 2024 Mar 11.

[6] Panda SP. Augmented and Virtual Reality in Intelligent Systems. Available at SSRN. 2021 Apr 16.

[7] Abisoye A, Akerele JI. A scalable and impactful model for harnessing artificial intelligence and cybersecurity to revolutionize workforce development and empower marginalized youth. International Journal of Multidisciplinary Research and Growth Evaluation. 2022 Jan;3(1):714-9.

[8] Raman R, Buddhi D, Lakhera G, Gupta Z, Joshi A, Saini D. An investigation on the role of artificial intelligence in scalable visual data analytics. In2023 International Conference on Artificial Intelligence and Smart Communication (AISC) 2023 Jan 27 (pp. 666-670). IEEE.

[9] Panda SP. The Evolution and Defense Against Social Engineering and Phishing Attacks. International Journal of Science and Research (IJSR). 2025 Jan 1.

[10] Newton C, Singleton J, Copland C, Kitchen S, Hudack J. Scalability in modeling and simulation systems for multi-agent, AI, and machine learning applications. InArtificial Intelligence and Machine Learning for Multi-Domain Operations Applications III 2021 Apr 12 (Vol. 11746, pp. 534-552). SPIE.

[11] Bestelmeyer BT, Marcillo G, McCord SE, Mirsky S, Moglen G, Neven LG, Peters D, Sohoulande C, Wakie T. Scaling up agricultural research with artificial intelligence. IT Professional. 2020 May 21;22(3):33-8.

[12] Meir Y, Sardi S, Hodassman S, Kisos K, Ben-Noam I, Goldental A, Kanter I. Power-law scaling to assist with key challenges in artificial intelligence. Scientific reports. 2020 Nov 12;10(1):19628.

[13] Shivadekar S, Kataria DB, Hundekar S, Wanjale K, Balpande VP, Suryawanshi R. Deep learning based image classification of lungs radiography for detecting covid-19 using a deep cnn and resnet 50. International Journal of Intelligent Systems and Applications in Engineering. 2023;11:241-50.

[14] Panda SP. Relational, NoSQL, and Artificial Intelligence-Integrated Database Architectures: Foundations, Cloud Platforms, and Regulatory-Compliant Systems. Deep Science Publishing; 2025 Jun 22.

[15] Shlezinger N, Ma M, Lavi O, Nguyen NT, Eldar YC, Juntti M. Artificial intelligence-empowered hybrid multiple-input/multiple-output beamforming: Learning to optimize for high-throughput scalable MIMO. IEEE Vehicular Technology Magazine. 2024 May 20;19(3):58-67.

[16] Samuel O, Javaid N, Alghamdi TA, Kumar N. Towards sustainable smart cities: A secure and scalable trading system for residential homes using blockchain and artificial intelligence. Sustainable Cities and Society. 2022 Jan 1;76:103371.

[17] Villegas-Ch W, Govea J, Gurierrez R, Mera-Navarrete A. Optimizing security in IoT ecosystems using hybrid artificial intelligence and blockchain models: a scalable and efficient approach for threat detection. IEEE Access. 2025 Jan 22.

[18] Mungoli N. Scalable, distributed AI frameworks: leveraging cloud computing for enhanced deep learning performance and efficiency. arXiv preprint arXiv:2304.13738. 2023 Apr 26.

[19] Panda SP. Artificial Intelligence Across Borders: Transforming Industries Through Intelligent Innovation. Deep Science Publishing; 2025 Jun 6.

[20] Cheetham AK, Seshadri R. Artificial intelligence driving materials discovery? perspective on the article: Scaling deep learning for materials discovery. Chemistry of Materials. 2024 Apr 8;36(8):3490-5.

[21] Panda SP, Muppala M, Koneti SB. The Contribution of AI in Climate Modeling and Sustainable Decision-Making. Available at SSRN 5283619. 2025 Jun 1.

[22] Shivadekar S. Artificial Intelligence for Cognitive Systems: Deep Learning, Neuro-symbolic Integration, and Human-Centric Intelligence. Deep Science Publishing; 2025 Jun 30.

[23] DeCost BL, Hattrick-Simpers JR, Trautt Z, Kusne AG, Campo E, Green ML. Scientific AI in materials science: a path to a sustainable and scalable paradigm. Machine learning: science and technology. 2020 Jul 14;1(3):033001.

[24] Klamma R, de Lange P, Neumann AT, Hensen B, Kravcik M, Wang X, Kuzilek J. Scaling mentoring support with distributed artificial intelligence. InInternational Conference on

Intelligent Tutoring Systems 2020 Jun 3 (pp. 38-44). Cham: Springer International Publishing.

[25] Otaigbe I. Scaling up artificial intelligence to curb infectious diseases in Africa. Frontiers in Digital Health. 2022 Oct 21;4:1030427.

[26] Dasawat SS, Sharma S. Cyber security integration with smart new age sustainable startup business, risk management, automation and scaling system for entrepreneurs: An artificial intelligence approach. In2023 7th international conference on intelligent computing and control systems (ICICCS) 2023 May 17 (pp. 1357-1363). IEEE.

[27] Peteiro-Barral D, Guijarro-Berdiñas B. A study on the scalability of artificial neural networks training algorithms using multiple-criteria decision-making methods. InInternational Conference on Artificial Intelligence and Soft Computing 2013 Jun 9 (pp. 162-173). Berlin, Heidelberg: Springer Berlin Heidelberg.

[28] Kuguoglu BK, van der Voort H, Janssen M. The giant leap for smart cities: Scaling up smart city artificial intelligence of things (AIoT) initiatives. Sustainability. 2021 Nov 7;13(21):12295.

[29] Gowda D, Chaithra SM, Gujar SS, Shaikh SF, Ingole BS, Reddy NS. Scalable ai solutions for iot-based healthcare systems using cloud platforms. In2024 8th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) 2024 Oct 3 (pp. 156-162). IEEE.

[30] Awan MZ, Jadoon KK, Masood A. Scalable and effective artificial intelligence for multivariate radar environment. Engineering Applications of Artificial Intelligence. 2023 Oct 1;125:106680.

[31] Landin M. Artificial intelligence tools for scaling up of high shear wet granulation process. Journal of Pharmaceutical Sciences. 2017 Jan 1;106(1):273-7.

[32] Panda SP. Securing 5G Critical Interfaces: A Zero Trust Approach for Next-Generation Network Resilience. In2025 12th International Conference on Information Technology (ICIT) 2025 May 27 (pp. 141-146). IEEE.

[33] Mocanu DC, Mocanu E, Stone P, Nguyen PH, Gibescu M, Liotta A. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. Nature communications. 2018 Jun 19;9(1):2383.

[34] Blanco L, Kukliński S, Zeydan E, Rezazadeh F, Chawla A, Zanzi L, Devoti F, Kolakowski R, Vlahodimitropoulou V, Chochliouros I, Bosneag AM. Ai-driven framework for scalable management of network slices. IEEE Communications Magazine. 2023 Nov 23;61(11):216-22.

[35] Sadek AH, Mostafa MK. Preparation of nano zero-valent aluminum for one-step removal of methylene blue from aqueous solutions: cost analysis for scaling-up and artificial intelligence. Applied Water Science. 2023 Feb;13(2):34.

[36] Cohen RY, Kovacheva VP. A methodology for a scalable, collaborative, and resource-efficient platform, MERLIN, to facilitate healthcare AI research. IEEE journal of biomedical and health informatics. 2023 Mar 20;27(6):3014-25.

[37] Adelodun AB, Ogundokun RO, Yekini AO, Awotunde JB, Timothy CC. Explainable artificial intelligence with scaling techniques to classify breast cancer images. InExplainable Machine Learning for Multimedia Based Healthcare Applications 2023 Sep 9 (pp. 99-137). Cham: Springer International Publishing.

[38] Sanz JL, Zhu Y. Toward scalable artificial intelligence in finance. In2021 IEEE International Conference on Services Computing (SCC) 2021 Sep 5 (pp. 460-469). IEEE.
[39] Haefner N, Parida V, Gassmann O, Wincent J. Implementing and scaling artificial intelligence: A review, framework, and research agenda. Technological Forecasting and Social Change. 2023 Dec 1;197:122878.
[40] Sai S, Chamola V, Choo KK, Sikdar B, Rodrigues JJ. Confluence of blockchain and artificial intelligence technologies for secure and scalable healthcare solutions: A review. IEEE Internet of Things Journal. 2022 Dec 29;10(7):5873-97.
[41] Moro-Visconti R. Artificial Intelligence-Driven Digital Scalability and Growth Options. InArtificial Intelligence Valuation: The Impact on Automation, BioTech, ChatBots, FinTech, B2B2C, and Other Industries 2024 Jun 2 (pp. 131-204). Cham: Springer Nature Switzerland.
[42] Oikonomou EK, Khera R. Designing medical artificial intelligence systems for global use: focus on interoperability, scalability, and accessibility. Hellenic Journal of Cardiology. 2025 Jan 1;81:9-17.
[43] Sayed-Mouchaweh M, Sayed-Mouchaweh, James. Artificial Intelligence Techniques for a Scalable Energy Transition. Springer International Publishing; 2020.
[44] Govea J, Ocampo Edye E, Revelo-Tapia S, Villegas-Ch W. Optimization and scalability of educational platforms: Integration of artificial intelligence and cloud computing. Computers. 2023 Nov 1;12(11):223.
[45] Hammad A, Abu-Zaid R. Applications of AI in decentralized computing systems: harnessing artificial intelligence for enhanced scalability, efficiency, and autonomous decision-making in distributed architectures. Applied Research in Artificial Intelligence and Cloud Computing. 2024;7(6):161-87.
[46] Pazho AD, Neff C, Noghre GA, Ardabili BR, Yao S, Baharani M, Tabkhi H. Ancilia: Scalable intelligent video surveillance for the artificial intelligence of things. IEEE Internet of Things Journal. 2023 Mar 31;10(17):14940-51.
[47] Sakly H, Guetari R, Kraiem N, editors. Scalable Artificial Intelligence for Healthcare: Advancing AI Solutions for Global Health Challenges. CRC Press; 2025 May 6.
[48] Shivadekar S, Halem M, Yeah Y, Vibhute S. Edge AI cosmos blockchain distributed network for precise ablh detection. Multimedia tools and applications. 2024 Aug;83(27):69083-109.