A Systematic Review of MLOps Tools: Tool Adoption, Lifecycle Coverage, and Critical Insights

Zakkarija Micallef
Vrije Universiteit Amsterdam, The Netherlands
z.micallef@student.vu.nl

Abstract

Machine learning operations (MLOps) has become increasingly critical as more organisations move machine learning models into production. However, the growing landscape of MLOps solutions has introduced complexity for practitioners trying to select appropriate tools. To investigate how and why these tools are adopted in practice, this paper conducts a systematic review of academic literature focused on MLOps tools. The analysis maps tools to MLOps lifecycle stages to reveal their function, scope, and the challenges they are designed to address. It identifies usage trends and synthesises reported benefits and limitations based on practical implementations. The most commonly used components, according to the findings, are orchestration frameworks, data versioning, experiment tracking, and managed cloud platforms. No single tool covers the entire lifecycle, so researchers often combine multiple tools to build complete pipelines. This highlights the importance of interoperability across MLOps tools in real-world MLOps pipelines.

Keywords

Systematic Literature Review, MLOps, tool taxonomy, lifecycle mapping, tool adoption, MLflow, Kubeflow

1 Introduction

In recent years, AI has experienced a dramatic surge in popularity. As more companies deploy AI solutions, they quickly discover that specialised infrastructure is essential to support these new capabilities. However, many AI engineers and data scientists lack the software-deployment expertise of operations teams, especially when it comes to MLOps tools [14].

Traditionally, an operations team is in charge of tasks such as deployment and ongoing monitoring [16]. To reduce software time-to-value and create stronger collaboration between development and operations, software companies frequently use DevOps. DevOps is described as a culture that emphasises continuous collaboration throughout the software lifecycle. It involves practices like Continuous Integration, which involves frequent code merges, and Continuous Deployment, which automates the release process to ensure software remains deployable.

MLOps refers to the entire lifecycle of the machine-learning process, bridging the gap between data, development, and operations [22]. MLOps extends beyond applying DevOps principles to machine learning (ML). It involves continuous integration and continuous deployment automation for ML pipelines, orchestration of ML workflows, versioning of data, models, and code to ensure reproducibility, continuous training to keep models up to date, metadata tracking for experiment auditability, and continuous evaluation and performance monitoring.

Despite the growing interest in MLOps, existing reviews often remain high-level, focusing primarily on listing tools, comparing surface-level features, or distinguishing between open-source and proprietary solutions. However, there is limited synthesis of how these tools are actually used in practice [37]. Most studies fail to capture the practical experiences of teams deploying real-world ML systems. This gap in the literature limits the ability of researchers and practitioners to make informed choices based on implementation outcomes rather than tool specifications alone. In this paper, we explore the range of MLOps tools referenced in academic literature and examine their real-world applications. While other papers integrate general-purpose DevOps tools such as Docker, Kubernetes, Git, and Jenkins with ML-specific tools, our SLR (Systematic Literature Review) focuses exclusively on MLOps native tools that are developed specifically for machine learning workloads, excluding general-purpose tools. We explore how practitioners use these tools in their pipelines, why they select specific MLOps solutions, and the advantages and drawbacks they face in real-world scenarios. Through this analysis, we aim to provide a clear view of the current MLOps landscape.

2 Related Work

MLOps remains a vague umbrella term, with its scope and implications still ambiguous for both researchers and practitioners. To bring clarity, Kreuzberger et al. [22] conducted mixed-method research, where they combined a literature review, a tool review, and expert interviews to create a comprehensive description of MLOps that we adopt as our definition of MLOps in Section 1. Their work has become a *de facto* standard, widely cited by other papers [10] [45] [48]. While Kreuzberger *et al.* [22] identified principles, technical components, and roles of MLOps, we focus solely on the technical elements that recur across the literature: CI/CD pipelines for training and deployment, workflow orchestration, feature-store systems, model-training infrastructure, model registries, metadata stores, serving components, and monitoring tools. A synthesis of prior surveys shows that these components consistently form the backbone of MLOps solutions [37] [45] [48] [10] [32]:

- Data engineering
- Version control
- Hyperparameter tuning and experiment tracking
- CI/CD pipelines for training and deployment
- Workflow orchestration
- Model deployment/serving
- Automated testing and validation
- Continuous performance monitoring

Varon Maya's [26] argues that bringing DevOps principles into ML pushes organisations toward pipeline-like architectures: NVIDIA [27], Facebook [19], Spotify [1] and Google [18] each describe a

flow that runs from data collection and feature engineering through training, validation and model serving, occasionally adding feedback loops that trigger continuous retraining. This structure clarifies ownership, lets every stage use specialised tooling and aligns with CI/CD automation. However, higher-level concerns such as process orchestration and configuration management are largely unaddressed with this pipeline architecture, creating challenges for reproducibility and maintainability in complex ML systems. Our systematic literature review (SLR) adopts the same pipeline perspective, evaluating MLOps tools stage by stage.

In modern software development environments, multiple languages and libraries are combined. To determine which languages, frameworks, and runtimes each tool supports, Wazir *et al.* [45] looked at 22 open-source catalogues and research publications. Given the frequent combination of Python, Java, R, and other specialised libraries, their findings highlight the need for MLOps platforms to continue being language agnostic. Building on this, Ruf *et al.* [38] conducted interviews and real-world trials to produce a detailed feature matrix of API bindings and integration hooks offered by each candidate tool. They show that lacking support for languages like R or Java can hinder collaboration between operations and data science teams. Consequently, they recommend conducting several iterative selection rounds before beginning work, ensuring all departments agree on a unified toolchain.

In Recupito et al.'s review [37], thirteen prominent MLOps platforms were mapped to the stage of the MLOps pipeline they support: data management, model training, CI/CD, monitoring, and so on. Their comparison shows that no single product spans the entire lifecycle, so practitioners routinely assemble multi-tool pipelines. Multiple studies [37] [38] [45] highlight the importance of evaluating not just each tool's individual strengths but also how well they interoperate with other tools and the dependencies they introduce. Interestingly, nearly 50% of the sources in Recupito *et al.* [37] review are blog posts, underscoring how much MLOps expertise circulates through informal channels rather than peer-reviewed papers. While several recent papers embrace a multivocal approach that blends academic and practitioner sources, our decision to focus solely on academic papers means that some practitioner findings may be missed.

Our SLR compares tools by the benefits and limitations that paper authors explicitly report from their practical experience, in contrast to the majority of previous reviews that compare tools based on claimed features listed on the tool developer's websites or their official blogs. This method provides a more accurate understanding of the practical experience and shortcomings of MLOps tools.

3 Study Design

3.1 Research Goal

This study presents a clear landscape of MLOps tools through a SLR of academic papers. It examines the platforms and libraries adopted by practitioners and analyzes which stages each tool addresses to reveal its function, scope, and the challenges it is designed for. The review also investigates the factors influencing tool adoption, offering insights into the technologies most relevant to current MLOps practices and how they integrate into real-world workflows.

3.2 Research Questions

3.2.1 RQ1: Which tools employed in MLOps workflows are most frequently reported in academic literature? Tools with significant popularity frequently benefit from strong community support, extensive documentation, and rich integration possibilities, making them preferred candidates for further exploration. They give a great perspective on how many practitioners interact with MLOps. The goal of this research question is to determine which tools are most often used in the literature. We can identify which tools are most popular by looking at how frequently they are integrated.

3.2.2 RQ2: Which stages of the MLOps lifecycle do these tools cover in the use cases reported in the academic literature? ML pipelines are made up of several steps, ranging from data processing and model training to model deployment and continuous monitoring. MLOps tools rarely attempt to cover all of these stages comprehensively. They generally specialise in a single or subset of these stages. Analyzing which stages each tool addresses reveals its function, scope and challenges it is designed for. This understanding is crucial since, as mentioned before, engineers and researchers tend to combine multiple tools to get their final desired pipeline.

3.2.3 RQ3: What are the reported benefits and limitations of the MLOps tool? MLOps tools offer a variety of features designed to meet the diverse needs of the user. Following the analysis of existing literature and documented user experiences, we examine common trends in the advantages and limitations of various MLOps tools.

3.3 Pilot Study

Before starting the full review, we ran a pilot study with just the first ten papers from the Google Scholar results, following Kitchenham's SLR guidelines [21]. A pilot study is important to validate our methodology and ensure the consistency and reliability of selection and extraction before full deployment. We subjected the selected studies to the full review protocol. The pilot revealed that some inclusion and exclusion criteria were incorrect, so we refined the set of extraction fields by removing unnecessary items and adding missing elements. We adjusted the search string, updated the inclusion rules, and modified the extraction spreadsheet. The study design presented below is the final version after all of those tweaks were applied.

3.4 Initial search

In February 2025, we conducted our literature search on Google Scholar. We first identified relevant keywords and synonyms for the MLOps literature. The search-term selection process involved several iterations of testing various keywords in Google Scholar and assessing whether the top results aligned with MLOps tooling and lifecycle studies. The finalised search string is as follows:

Google Scholar Search String

("MLOps" OR "machine learning operations") AND ("tool" OR "application" OR "framework" OR "platform" OR "pipelines") AND ("comparison" OR "evaluation" OR "benchmark" OR "analysis" OR "empirical")

The initial query returned 9 100 records; however, to keep the review scope manageable and respect our time constraints, we screened only the first ten pages of Google Scholar results and reviewed 96 papers.

3.5 Application of Selection Criteria

For a paper to qualify as a primary study, it must satisfy all inclusion criteria and none of the exclusion criteria.

Inclusion Criteria:

- I1: Papers that either analyse/compar e MLOps tools or describe the development/implementation of MLOps tools, frameworks, or pipelines.
- I2 Papers published in the last five years to ensure relevance (i.e., 2020 and later)

Exclusion Criteria:

- E1: Papers only describing applications or projects using an MLOps tool without detailed tool analysis.
- E2: Papers discussing MLOps as a concept/architecture or highlighting the benefits of the MLOps culture.
- E3: The study must present original primary research; excluded are secondary studies such as literature reviews or surveys.
- E4: Non-English literature.
- E5: The full text of the paper is not available for our institution.

The selection criteria I2, E4, and E5 are standard criteria for SLRs. Criteria I1, E1, E2, and E3 ensure we exclude papers outside the scope of this review (those lacking original primary research or detailed analysis/development of MLOps tools) and thus focus only on studies that directly contribute to our research questions.

3.6 Snowballing

We applied both backward and forward snowballing to expand the results obtained through our initial search. Our snowballing process follows the approach suggested by Wohlin et al. [46]. To perform backward snowballing, we first reviewed the reference sections of our initial set of 96 papers by manually assessing titles to determine their relevance. If a title aligned with our research scope, we further examined the corresponding paper's abstract. Through this process, we identified four additional papers that met our inclusion criteria. On the other hand, during forward snowballing, we used Google Scholar to find studies that cite our initial set of papers. From this process, seven additional relevant papers were identified.

In both cases of forward and backward snowballing, we applied the same selection criteria and data extraction procedures (3.5, 3.7) as those used for the initial set of papers. We limited the snowballing process to a single round since further iterations yielded minimal additional relevant papers beyond those identified initially.

Figure 1 illustrates the complete selection process, from initial paper retrieval, through screening and inclusion, to snowballing.

3.7 Data Extraction

In this phase of our study, we performed a systematic analysis of the primary studies to extract data related to our research questions. As detailed in subsection 3.3, our initial pilot study helped us refine the

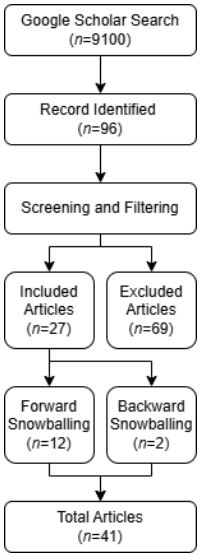

Figure 1: Flow diagram showing selection process record retrieval, screening, inclusion, and citation snowballing steps for the MLOps literature review

data extraction strategy. For each research question, we extracted a predetermined set of information from every primary research paper and recorded it in spreadsheet. With respect to RQ1, we identified tools that were actively implemented, excluding tools that were only mentioned. This distinction enabled us to evaluate the researchers' hands-on experiences with the tools. As for RQ2, we noted the tools' specific use cases mentioned in the paper. Finally, for RQ3, we gathered information on the reported benefits and limitations of each tool by analysing the justifications for their selection and the challenges encountered during their application. Table 1 lists the data extraction fields used.

Table 1: Data extraction fields

Field	Description
Name	Name of the MLOps tool
Uses	Practitioner use cases
Pipeline stage covered	Stage(s) of the MLOps lifecycle addressed
Paper	Citation of the academic study
Benefits	Reported benefits of the tool
Limitations	Reported limitations or challenges
Notes	Additional observations or context

3.8 Data Synthesis

3.8.1 RQ1: Which tools employed in MLOps workflows are most frequently reported in academic literature? In this stage of our study, we synthesise the data extracted from the selected primary research papers to address our research questions. To represent our findings for RQ1, we constructed Figure 2, a frequency graph that lists each

tool alongside the number of times it was used in the reviewed literature. This graph provides a quantitative measure of each MLOps tool's popularity, further details and interpretation are presented in the Results and Discussion sections.

3.8.2 RQ2 To address RQ2, we categorised the tools based on their respective use cases, according to the MLOps components as defined by Najafabadi *et al.* [32]. In this section, we list the selected subset of the original components whilst adapting their names and responsibilities to our context. We have also introduced a new component, Visualisations, to better capture gaps in the existing architecture.

Table 2 summarises the MLOps lifecycle categories that emerged from the primary studies discussed in this section.

3.8.3 RQ3 To answer RQ3 we applied a descriptive synthesis, which means we looked for patterns in descriptions rather combining numerical results. First, we copied every sentence that mentioned a benefit or limitation of a tool into a spreadsheet. We then grouped together statements that conveyed the same idea, allowing us to identify benefits and limitations that appeared across several studies, as well as those mentioned only once. Finally, we compiled a table for every tool that lists these combined benefits and limitations so readers can see at a glance what the literature agrees on for each tool. This analysis helped us capture the context of tool selection, including the rationale behind their adoption and the challenges associated with their use.

3.9 Study Replicability

To ensure full replicability of our review, we have made a public Zenodo repository [28] with a spreadsheet that documents every step of the study. It comprises four sheets:

(1) **Paper Selection** – the complete set of Google Scholar search results, showing whether each paper was included or excluded from our review and the exact criterion applied.
(2) **Selected Papers** – all papers that passed screening, indicating whether they came from the primary search or from forward/backward snowballing, along with key metadata (author, publication year, and so on).
(3) **Extraction** – the raw data taken from each selected paper: tool names, quoted passages, reported benefits or limitations, and possible dependencies or integrations.
(4) **Synthesis** – a consolidated view that maps every extracted tool to the relevant component of the MLOps architecture as well as summaries of the common advantages and drawbacks. Key synthesis results are presented in this paper, while the full mapping is available in the supplementary material [28].

Researchers can follow the study design and the data in the provided sheets to reproduce our search, screening, extraction, and synthesis processes or to extend the analysis.

3.10 Threats to Validity

In this section, we follow the threat classification schemes for experiment validity described by Ampatzoglou et al. [3] and outline the threats that may affect the validity of our research.

Table 2: MLOps lifecycle categories observed in the primary studies

Component	Responsibility
Orchestrator	Provides system-wide orchestration and schedules multiple models while balancing throughput and latency.
Raw Data Store	Holds raw source data; needs specialised versioning tools because datasets exceed typical Git size limits.
Data Preprocessor	Transforms, cleans, and validates data before it becomes training input.
Dataset Repository	Stores and versions datasets; relies on large-file platforms.
Feature Store	Computes, stores, and serves reusable features with low latency.
Artefact Repository	Keeps packaged or containerised ML components that include a model.
ML Metadata Repository	Tracks training metadata for experiment tracking and model performance.
Code Repository	Versions source code, configuration files, and related artefacts.
Model Repository	Versions trained models together with basic metadata such as version numbers.
ML Training Pipeline (Online)	Automates continuous model training at runtime in production.
ML Experiment Pipeline (Offline)	Supports manual experimentation and training during design time.
MLOps User Interaction Manager	Enables interaction between the MLOps team and the platform.
ML Pipeline Editor	Builds, tests, and packages pipeline code into containers or similar environments.
Model Deployer	Deploys a trained model and its dependencies to production.
Model Evaluator	Measures and assesses model performance.
Runtime Model Monitor	Continuously watches serving performance and infrastructure metrics.
Visualisation	Presents dashboards and graphs for experiments, metrics, and system status.
End-to-End	Covers the full ML workflow from data ingestion to inference, though most tools still leave gaps.
Managed End-to-End	Provides an end-to-end pipeline fully managed by the platform, automating infrastructure, execution, and monitoring; often requires auxiliary services for complete coverage.

3.10.1 External validity External validity relates to the generalisability of our systematic literature review's findings. For our primary studies to accurately represent the MLOps field and draw correct conclusions, it is essential that these studies reflect the diverse MLOps landscape. One threat to generalisability is that the limitations reported in the literature may be based on older versions of MLOps tools, thereby inaccurately representing their current

state. Such limitations might have already been addressed by the developer and thus may no longer hold.

Another threat comes from the sheer number of results returned by Google Scholar. Its proprietary relevance ranking does act as a partial mitigation by prioritizing influential papers first. Nevertheless, we reviewed only the first ten result pages, covering 96 papers, so some relevant studies may have been missed and this could introduce selection bias. A broader search by screening more Google Scholar pages would mitigate this limitation.

3.10.2 Internal validity Internal validity examines whether the study's design and execution provide a confident basis for linking causes with effects. A key threat in our review is the process of selecting appropriate papers and formulating an exhaustive set of search terms. Vital studies might be missed if the search terms are not thorough, which could introduce bias. In order to mitigate this threat, we pre-identified a number of key papers that needed to be on our final list of papers.

Another threat concerns our tool–component heatmap (Figure 3). We can map only the capabilities that authors explicitly describe, so the heatmap may miss components that a tool supports but were not used in the included studies. When an author employs only a subset of the tools capabilities, any unmentioned components are left out, which makes the mapping non-exhaustive and might under-represent the true scope of certain tools.

3.10.3 Construct validity Construct validity concerns how well our measures and constructs align with the theoretical concepts we intend to study. A threat here is inherent in the source literature: many academic papers emphasise implementing MLOps pipelines or addressing broader issues rather than critically evaluating the tools themselves. This tendency often leads to detailed reporting on the benefits while underreporting limitations, which may result in conclusions that do not accurately capture the tool's effectiveness.

3.10.4 Conclusion validity Conclusion validity focuses on the accuracy of the deductions generated from our data analysis. Our conclusions, which are based on frequency counts, categorisation mappings, and feature evaluations, are meant to be rational. However, verifying that the data analysis procedures are sound and executed correctly is crucial to confirming the validity of our findings. Finally, the extraction and synthesis process was not reviewed by a third party, which could introduce self-bias.

4 Results

In this section, we present the outcomes of our SLR structured by three research questions. Section 4.1 (RQ1) reports the most frequently mentioned MLOps tools (Figure 2); Section 4.2 (RQ2) examines tool capabilities across pipeline stages and categories (Figures 3 and 4); and Section 4.3 (RQ3) synthesizes reported benefits and limitations (Tables 3–10). Further discussion appears in Section 5.

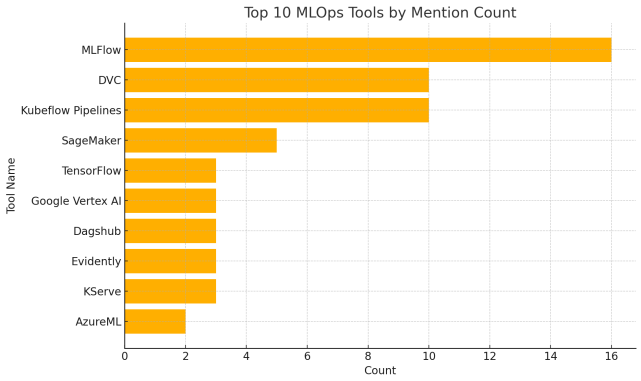

Figure 2: Top 10 MLOps tools ranked by number of mentions in primary studies

4.1 RQ1: Which tools employed in MLOps workflows are most frequently reported in academic literature?

Our first research question aims to identify the most widely used MLOps tools based on evidence from academic studies. In order to determine the prevalent tools, Figure 2 presents a bar graph showcasing tool usage based on our primary data sources. As explained previously, each tool had to be both described in the literature and practically applied by the authors for it to be included in our study. A significant portion of the literature focused on authors researching multiple tools in order to implement an MLOps pipeline. In these studies, the literature review section typically describes various MLOps tools and evaluates them based on the features offered. In our analysis, we only listed the tools that were actually implemented in their final pipeline. As shown in Figure 2, MLflow emerged as the most commonly implemented tool, appearing 16 times, followed by DVC and Kubeflow Pipelines, each appearing 10 times. Further interpretation is provided in the discussion section.

4.2 RQ2: Which stages of the MLOps lifecycle do these tools cover in the use cases reported in the academic literature?

MLOps pipelines encompass a range of stages, including data ingestion, pre-processing, model development, training, validation, deployment, and ongoing monitoring. A heatmap (Figure 3) was created to visualise what component(s) each extracted MLOps tool fulfils, along with the frequency of each tool's implementation across the reviewed studies. Analysing the stages supported by each tool revealed common areas of focus among MLOps solutions. Overall, very few tools cover the entire pipeline, as most specialise in one or more phases. Since MLOps tools can address several components, authors may discuss only a subset. As a result, the figure may omit certain stages a tool could cover, making our assessment of functionality incomplete. Throughout this paper we use the terms "stage", "phase", and "component" interchangeably to refer to one functional step in that lifecycle.

Furthermore, Figure 4 presents a bar graph of MLOps tool counts grouped by broad categories. These categories are adapted from

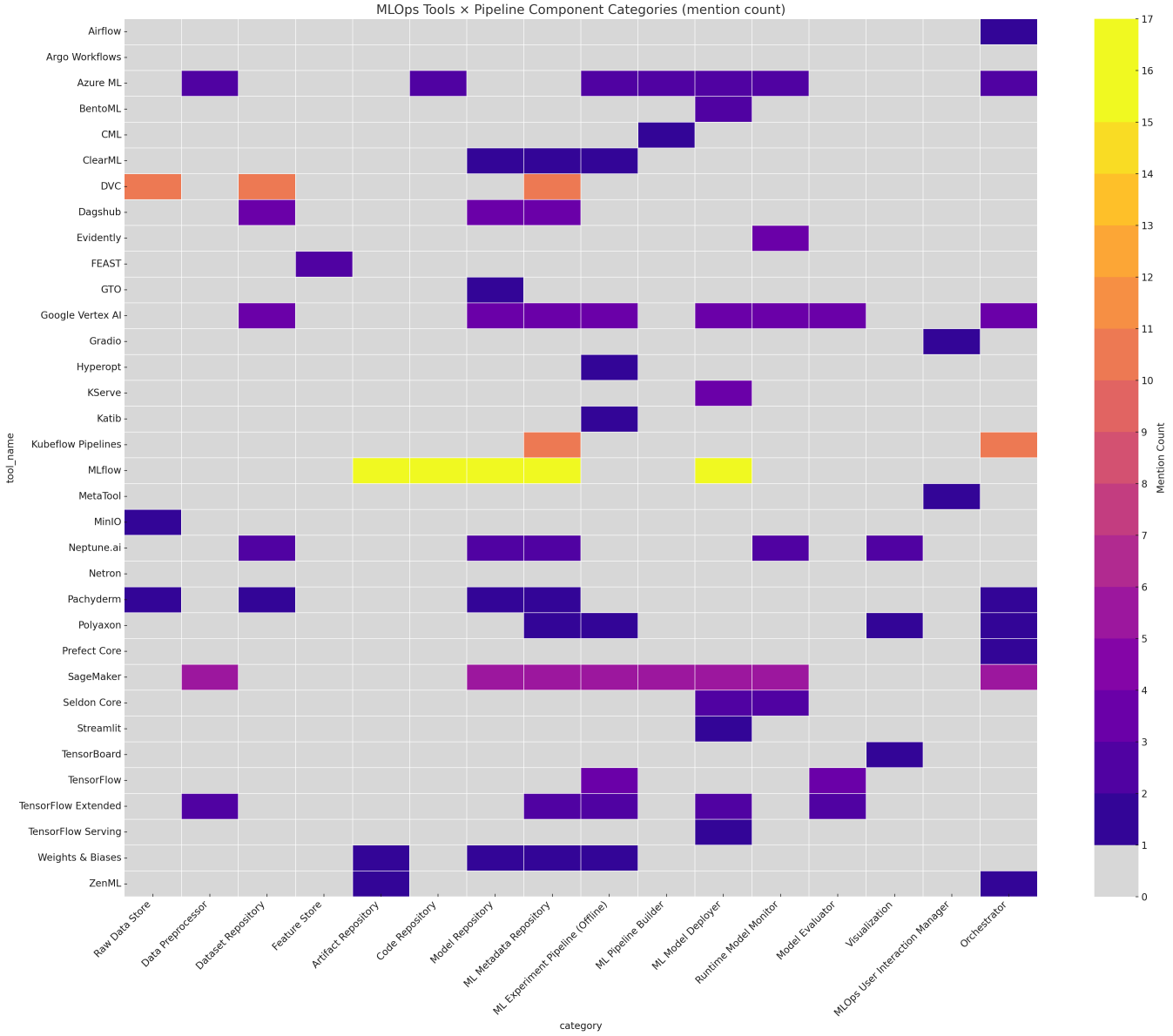

Figure 3: Heatmap illustrating MLOps tools mapped across different MLOps pipeline phases, showing which phase(s) each tool supports. Colour intensity represents the number of times each tool was implemented.

the taxonomy proposed by Najafabadi *et al.*, who originally defined six groups to classify MLOps components. To better reflect our corpus, we both added two new groups—End-to-End and Managed End-to-End—and removed unused groups such as Data Curation. The full set of categories is:

- **End-to-End**
- **Managed End-to-End**
- **Storage and Versioning**
- **Infrastructure and Supporting Services**
- **Inference**
- **ML Training**

This categorization is primarily intended to facilitate the organization and presentation of our findings in this paper. While individual tools may fall under multiple MLOps components, we assign each tool to a single category to provide a clear and high-level overview of the MLOps landscape.

4.3 RQ3: What are the reported benefits and limitations of the MLOps tools?

The following Tables 3–10 present a synthesis of the reported benefits and limitations of each MLOps tool identified in our review, thereby addressing Research Question 3.

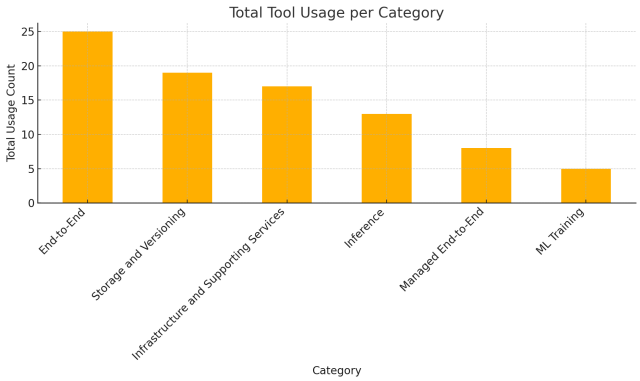

Figure 4: MLOps tool count grouped by category

These tables are grouped according to the categories defined in Section 4.2 (End-to-End, Managed End-to-End, Storage and Versioning, Infrastructure and Supporting Services, Inference, ML Training), which allows for straightforward comparison of the practical considerations involved in tool selection and implementation. Each table organises tools by one of these categories.

These summaries are drawn from direct quotations in the source literature, capturing the most common insights authors reported from academic projects or industrial deployments. The complete set of quotations, tools, and source papers is available in the accompanying supplementary sheets.

The tables are followed by Section 5, where we delve deeper into the implications of these findings.

4.3.1 *Infrastructure and Supporting Services*

Apache Airflow is an open-source platform for orchestrating production workflows and data pipelines [50].

Argo Workflow Argo Workflow is a container-native workflow engine for orchestrating parallel jobs on Kubernetes [30]. By combining Argo Events for webhook-triggered workflows and Argo Workflow for execution, developers can automate the full ML lifecycle in a reproducible, scalable, and hands-off manner. This includes training, evaluation, and deployment. While production environments can benefit from such pipelines for mature, production-ready teams, it is not recommended to incorporate such an environment in the early stages of ML services or businesses [30].

Kubeflow Pipelines handles managing and orchestrating containerised workloads [15] while taking care of model training, deployment, and coordination [24]. Since it runs on top of Kubernetes, it lets users codify preprocessing, training, and deployment steps in a single UI and execute many pipelines in parallel. Because every component ships as a Kubernetes resource, the same pipeline runs on-prem or in any managed Kubernetes service. It also has autoscaling pods, notebook sessions provided by Jupyter, experiment tracking, hyperparameter tuning, and pluggable serving (Kubeflow's own KServe or other solutions such as Seldon Core), which all sit behind the central dashboard.

Prefect Core is a pipeline orchestrator with a modern Python API that keeps code readable and flexible when workflows are highly dynamic [11].

ZenML links the orchestration layer with an artefact store so that a stack can swap orchestration, or any other component, without rewriting user code [5].

CML enables CI/CD for ML projects. It wires continuous integration to ML experiments, tracking changes and auto-generating metric reports [35].

Gradio enables fast creation of interactive web interfaces for model demos and evaluation without touching HTML, CSS, or JavaScript [15].

TensorBoard is TensorFlow's official visualisation toolkit, offering interactive exploration of computation graphs, layer structures, and distributions of weights and biases through histograms. TensorBoard also provides real-time plotting of training metrics such as loss and accuracy over time [47].

4.3.2 *Managed End-to-End*

AzureML is a managed end-to-end service; users still provision the workspace, storage, networking, and container registry either interactively or through infrastructure-as-code [29].

AWS SageMaker is Amazon's cloud platform for building, training, tuning, and deploying models in a hosted production environment [34].

ClearML provides semi-automated experimental run tracking and auditing, capturing comprehensive metadata, parameters, and metrics across popular Data Science tools. It relies on a dedicated central web server underpinned by a MongoDB database, an Elasticsearch deployment, and cloud object storage for artefact archiving, enabling end-to-end traceability and auditability of experiments [40].

4.3.3 *End-to-End*

MLflow is an end-to-end tool made of multiple modules [9] which was initially started as an open-source platform to provide comprehensive experiment tracking capabilities [25][31][5][44]. It has grown to cover most of the MLOps pipeline through the inclusion of the following components. The *MLFlow Tracking* component records parameters, metrics, and artefacts for every run [40], ensuring a complete workflow history. *MLFlow Projects* encapsulate data science code and dependencies, enabling reproducible execution across diverse environments. *MLflow pipeline* involves processing, cleaning, transforming, training models, and evaluating them, ensuring large-scale ML model deployment. *MLFlow Models* then offers a standardised structure for packaging machine learning models and facilitates seamless deployment in various serving infrastructures [9]. Finally, *MLFlow Registry* serves as a unified model store, providing versioning, annotation, stage transitions, and the promotion of specific model versions to "Production" for batch or real-time inference [31][39]. Autologging further streamlines experiments by automatically capturing parameters and metrics from supported ML libraries [5].

Neptune.ai positions itself as a complete platform that spans data versioning, testing, deployment, and monitoring [43].

Pachyderm offers GUI-based notebook services, experiment tracking, and hyperparameter tuning [23].

Polyaxon is an open-source, Kubernetes-native end-to-end MLOps platform focused on experiment tracking, visualisation, parallel executions and hyperparameter optimisation. Its main focus is parallelisable experiment tracking, where it attempts to provide computationally efficient mechanisms for running and interpreting parallel ML experiments at scale. While Polyaxon's core and experimentation tools are open source, its automation and management features are not [23].

TensorFlow Extended (TFX) offers a production-grade pipeline implementation for TensorFlow models [20].

Google Vertex AI is part of GCP, merging AutoML and AI Platform behind one API, client library, and UI [41][29]. The Workbench offers Jupyter notebooks [8] and AutoML streamlines training, experiment tracking, and metadata management. A Model Registry manages versions for online or batch prediction, while the Pipeline view visualises step dependencies, integrated logging, and monitoring track performance over time.

4.3.4 *ML Training*

Katib is part of Kubeflow. It automates hyperparameter tuning across frameworks including TensorFlow, MXNet, PyTorch, and XGBoost [17].

4.3.5 *Storage and Versioning*

Weights & Biases is cited as an ML metadata repository, yet the sources offer no elaboration [5].

Dagshub hosts data repositories and models for collaborative versioning. Some authors store all pipeline outputs remotely in it [31][2].

DVC extends Git to manage large datasets and model files, enabling version control of large data alongside code [31] [4] [25] [9] [36] [7][20][15]. Consequently, its Git-like workflow simplifies adoption for users familiar with Git [4]. DVC's pipeline feature modularizes data processing into stages with explicit input-output dependencies and parameterisation, ensuring reproducibility, automation, and reusability across projects [4]. It also supports experiment tracking and storing metadata within Git, while actual data resides in DVC storage, with metadata files orchestrating data retrieval [35].

Feast provides both offline and online feature storage, keeping historical as well as live data in sync [44]. One paper mislabels it as a performance monitoring tool [47].

GTO is Iterative's GitOps-based model registry. It eliminates the need for separate servers or databases by leveraging Git and DVC repositories and provides streamlined support for promoting models through designated repository branches and stages [35].

MinIO is an S3-compatible object store that keeps artefacts and metadata from ZenML and MLflow but can host any data [5].

4.3.6 *Inference*

Metaflow is a framework for creating and executing data science workflows in local environments and scaling them to the cloud with ease [40].

KServe originates from the Kubeflow ecosystem and wraps models as web services on Knative/Istio. This results in autoscaling, isolation, and an event-driven path to downstream monitoring [24][20][17].

BentoML is a model serving tool that bundles a trained model and its dependencies into a production-ready service [40][39].

Seldon Core is a model deployment tool that exposes models as web services [11] with robust support for Kubernetes [24].

Evidently is an open-source model monitoring library that generates predefined monitoring reports with a few lines of code. It can automatically detect drift in input data, target values, and predictions, and supports multi-model dashboards by calculating metrics across multiple models with a single monitor [2][44][24].

Streamlit provides a quick route to deploy ML applications through a simple Python interface [2].

5 Discussion

The following subsections are organised around our three research questions and draw directly on the synthesis captured in Tables 1–8. Each benefit extracted from the studies is labelled B1, B2, … and each limitation as L1, L2 and so on. Thus, a reference such as B31 sends you to the 31st benefit row, while L17 points to the 17th limitation. To examine the supporting evidence, consult the corresponding rows in Tables 1–8, where the summarised finding and the primary paper that reported it are.

5.1 RQ1 – Which tools employed in MLOps workflows are most frequently reported in academic literature?

Across the primary studies, four tools stand out: **MLflow**, **DVC**, **Kubeflow Pipelines**, and **AWS SageMaker** as shown in Figure 2. The first three are fully open-source, while AWS SageMaker is a proprietary cloud service. This divide reflects a familiar compromise in MLOps practice: practitioners prefer community-maintained tools for tasks like experiment tracking and data versioning but often turn to commercial platforms when they need infrastructure that scales quickly and comes with operational support.

Both MLflow and Kubeflow are classified as end-to-end platforms, providing experiment tracking, model packaging, and deployment in one bundle. However, their design philosophies diverge significantly. MLflow focuses on accessibility: it is easy to use, has a straightforward web UI, thorough and substantial documentation, wide storage-backend support, and a lively community (B24–B30). Kubeflow takes a different approach. It is rooted in Kubernetes and focuses more on customisability with fine-grained control, modular components, and automatic scaling (B3–B8). However, it demands a tougher setup and steeper operational know-how (L3–L5). The fact that both tools top the frequency chart suggests that while

some users prefer easy-to-use tools and others value customisable options, there is room for both approaches to succeed.

DVC is a notable outlier among the most cited tools, as it tackles only a single stage of the MLOps life cycle, data storage and versioning, and yet it appears almost as often as the full end-to-end platforms. Synchronising large data artefacts with code has been a historic pain point in ML. Authors repeatedly select DVC for its easy-to-use Git-like interface (B45) as well as pipeline caching and serverless architecture. A recurring pairing was the adoption of Git for code with DVC for large data because of Git's size limits. Git LFS was meant to alleviate this struggle, but unlike Git LFS, DVC needs no extra server, a difference reviewers flag as decisive. L29–L30) confirm that none of the other surveyed tools offer a comparable alternative.

Another reason for MLflow's popularity is that, even though it is open-source, it remains a mature, well-established platform trusted by leading companies and backed by both a strong community and Databricks (B26). Its popularity feeds a loop: more users attract more contributors, the codebase improves, and the project becomes even more appealing, which leads the community to further grow. The same holds true for Kubeflow (B4). Some projects, however, are less attractive because of their "uncertain vitality", meaning that their long-term health is harder to predict; Polyaxon, for instance, relies on a comparatively small contributor base (L27). Lower adoption means fewer contributors, slow progress, and even less visibility; a feedback loop in reverse that keeps new users away. In practice, the presence of an active maintainer community weighs heavily when choosing a tool for the long haul (B26, B4). MLflow does not just entice developers who want a self-hosted solution, since developers who prefer to skip infrastructure work can opt for a managed MLflow provided by Databricks, which sidesteps the need to run and maintain a separate tool (L16).

The widespread use of AWS SageMaker highlights the ongoing significance of managed services (Figure 2). Managed MLOps platforms such as SageMaker remain popular because they include a wide array of pre-integrated pipeline components, such as a feature store, model registry, and CI/CD templates (B17). Companies accept the subscription fee and the inherent risks of vendor lock-in because the alternatives present significant challenges and costs (L10, L12, L9, L15). A managed service hides the heavy work of provisioning and securing compute, storage, and networking, and it scales on demand (B18, B15, B16, B23). The same attraction applies to the managed stacks from Azure ML (B14) and Google Cloud Vertex AI (B22). Organisations that are unwilling to rely on a third-party cloud provider often choose to build an in-house pipeline from open-source projects such as MLflow for experiment tracking, Kubeflow for orchestration, and DVC for data versioning. However, this option demands far more integration effort and long-term maintenance and may be better suited to larger teams and more mature projects (L3, L29, L19, L16).

5.2 RQ2 – Which stages of the MLOps lifecycle do these tools cover in the reported use cases?

Mapping each tool to the taxonomy defined in Section 2 reveals that no single solution addresses the entire ML lifecycle.. However, this is more due to what the papers mention than what the product definitively offers. Kubeflow excels at orchestration (B4), MLflow at experiment tracking (B28), while DVC and Feast handle data and feature management, respectively (B45, B57).

AWS SageMaker bundles a model registry, feature store, and deployment tools, yet teams still turn to third-party services for granular security and local runs (L10–L12). The component that offers the least coverage is in feature stores and runtime monitoring; outside of Feast and Evidently, non-managed options are almost nonexistent (Tables 6 and 7). Consequently, multi-tool pipelines are the norm, underpinning the importance of effective tool integration.

While classifying tools, we found one component that did not fit cleanly into Najafabadi *et al.'s* [32] component architecture: visualisation dashboards such as TensorBoard, Weights & Biases, and Gradio. The closest existing component was Runtime Monitoring, which does not really capture their purpose, so we introduced a dedicated visualisation component instead.

Across all reviewed papers, the ML Metadata Repository (Experiment-tracking) stage is addressed the most. This dominance is caused by the ubiquity of MLflow, making it a default choice for an experiment tracking tool that is open source. Managed solutions such as SageMaker and Azure ML (B17, B16) further contribute to its widespread adoption. Reliable metadata is essential for reproducibility, auditability, and model evaluation; consequently, practitioners consistently prioritise this component.

Yet orchestration tools sit a close second: they are the "glue" that binds data prep, training, and deployment into a runnable pipeline, so virtually every study that examines end-to-end workflows also highlights orchestrators such as Kubeflow, Argo, or similar schedulers.

5.3 RQ3 – What are the claimed benefits and limitations of the MLOps tool?

5.3.1 Security and Managed Platform Trade-offs MLflow lacks automatic data versioning (L17), which explains why many studies pair it with DVC (B45–B49). It is also missing role-based access control (L19). In contrast, AWS SageMaker covers that gap through IAM integration (B19). A well-established trade-off emerges: open-source tools need extra engineering to meet enterprise-grade security, while cloud platforms shift that work to the provider at the cost of vendor lock-in. Even then, the cloud does not guarantee full coverage. For example, Google's Vertex AI still trails Azure ML and SageMaker on security, user control, and governance despite tying you to Google's stack.

5.3.2 Post-deployment Monitoring Evidently is the only tool that comes with a drift detection feature out of the box (B67–B71), and it works only for tabular data (L37). The limited findings indicate that post-deployment observability is still an emerging component of the ecosystem.

5.3.3 Prerequisite Knowledge A common limitation noted among tools is the required prerequisite knowledge. Pachyderm requires Helm and cloud-storage expertise (L23), while both Kubeflow and Argo Workflows demand solid Kubernetes and containerisation experience (L3, L7–L8). Several other services, including Feast's SDK (L34), Weights & Biases' client code (L33), Neptune's Python API

(L22) and SageMaker's SDK (L11), call for advanced programming skills. These findings show that tool comparison should not focus solely on a tool's features and limitations, but it must also consider the skills a team possesses.

5.3.4 Integration and Flexibility Easy integration and language agnosticism are among the most frequently praised benefits. Orchestrators such as Kubeflow Pipelines are applauded for both their cloud-agnostic Kubernetes foundation and their seamless hooks into TensorFlow Extended (B3, B6), while managed stacks win favour largely because of the way they slot into their parent ecosystems—AWS SageMaker's tight coupling with IAM and the rest of the AWS suite being a prime example (B19). At the experiment-tracking layer, MLflow extends this integrative spirit through container-friendly, self-hosted deployments and pluggable back-end stores that work just as well with S3, Azure Blob, on-prem NFS, or SQL-compatible databases (B25, B30). Flexibility in storage backends is also demonstrated by DVC's remote options (B49) and MLflow's broad object-store support (B30). TFX is widely appreciated for its portability, with pipelines that can run seamlessly across multiple orchestrators rather than being locked to a single workflow engine (B41). Finally, inference services such as BentoML embrace framework diversity by supporting TensorFlow, PyTorch, Keras, XGBoost, and more out of the box (B61). Taken together, these examples show that the community consistently rewards tools that can drop into existing tech stacks without forcing a wholesale rewrite in a particular language, framework, or cloud. In contrast, the lack of flexibility is listed as a drawback by reviewers, specifically in TensorBoard and TFX which tie users to the TensorFlow stack (L39, L28).

5.3.5 User Interfaces and Visualization Finally, visual dashboards and UIs are widely appreciated. Kubeflow's central UI (B7), Vertex AI's pipeline view (B22), MLflow's experiment board (B24) and Pachyderm's web console (B35) are all reported as benefits, as they improve ease of use.

6 Conclusion

This review set out to identify which MLOps-native tools appear most frequently in academic work and to understand the reasons for their uptake. A structured Google Scholar search, followed by manual screening, filtered 96 papers to 41 selected studies which were extracted and synthesised in a systematic, structured manner.

Our review confirms a clear pattern in recent MLOps practice. MLflow, DVC, Kubeflow Pipelines, and AWS SageMaker appear most frequently in MLOps pipelines because each addresses a critical pain point: MLflow simplifies experiment tracking, DVC brings Git-style version control to data, Kubeflow coordinates cloud-agnostic workflows, and SageMaker lifts the infrastructure burden through a fully managed service. Their popularity is sustained either by lively open-source communities or by the deep resources of a major cloud provider.

Yet none of these platforms covers the entire lifecycle on its own. Researchers commonly assemble a mixed tool pipeline. Future work should move beyond cataloguing tools to evaluating tool interoperability and integration effort. This includes analyzing each tool's dependencies on external platforms and how easily it can be combined with others to build coherent workflows. A useful extension would be a comparative table showing dependency relationships and open-source status, offering a clearer picture of ecosystem maturity and possible vendor lock-in. Benchmarking integration effort, metadata consistency, and runtime stability would offer the objective metrics needed to help practitioners choose interoperable end-to-end solutions.

References

[1] [n. d.]. The Winding Road to Better Machine Learning Infrastructure Through Tensorflow Extended and Kubeflow | Spotify Engineering. https://engineering.atspotify.com/2019/12/the-winding-road-to-better-machine-learning-infrastructure-through-tensorflow-extended-and-kubeflow

[2] William Inouye Almeida. 2023. *Building an Automated MLOps Pipeline and Recommending an Open-Source Stack to Deploy a Machine Learning Application.* Master's thesis. Universidade do Porto (Portugal). https://search.proquest.com/openview/3caae32a5b14a5346907376fce17b338/1?pq-origsite=gscholar&cbl=2026366&diss=y

[3] Apostolos Ampatzoglou, Stamatia Bibi, Paris Avgeriou, Marijn Verbeek, and Alexander Chatzigeorgiou. 2019. Identifying, categorizing and mitigating threats to validity in software engineering secondary studies. *Information and Software Technology* 106 (2019), 201–230. https://doi.org/10.1016/j.infsof.2018.10.006

[4] Vidushi Arora. 2024. Exploring real-world challenges in MLOps implementation: a case study approach to design effective data pipelines. (2024). https://elib.uni-stuttgart.de/items/e6f46863-465d-4e88-861c-9dcbabc746db

[5] Michal Bacigál. 2024. Design and Implementation of Machine Learning Operations. (Feb. 2024). https://dspace.cvut.cz/handle/10467/113781 Accepted: 2024-02-09T23:53:17Z Publisher: České vysoké učení technické v Praze. Vypočetní a informační centrum..

[6] Rahul Bagai, Ankit Masrani, Piyush Ranjan, Madhavi Najana, and Ankit Masrani. 2024. Implementing Continuous Integration and Deployment (CI/CD) for Machine Learning Models on AWS. *International Journal of Global Innovations and Solutions (IJGIS)* (May 2024). https://doi.org/10.21428/e90189c8.9cb39c55 Publisher: The New World Foundation.

[7] N. Bauman. 2022. Building a generalisable ML pipeline at ING. (2022). https://repository.tudelft.nl/record/uuid:35c850eb-1d03-4185-a8c5-4469b2112327

[8] Ralph Bergmann, Felix Theusch, Paul Heisterkamp, and Narek Grigoryan. 2024. Comparative Analysis of Open-Source ML Pipeline Orchestration Platforms. (2024). https://www.researchgate.net/profile/Narek-Grigoryan-3/publication/382114154_Comparative_Analysis_of_Open-Source_ML_Pipeline_Orchestration_Platforms/links/668e31d6b15ba559074d9a4b/Comparative-Analysis-of-Open-Source-ML-Pipeline-Orchestration-Platforms.pdf

[9] Anas Bodor, Meriem Hnida, and Daoudi Najima. 2023. From Development to Deployment: An Approach to MLOps Monitoring for Machine Learning Model Operationalization. In *2023 14th International Conference on Intelligent Systems: Theories and Applications (SITA)*. 1–7. https://doi.org/10.1109/SITA60746.2023.10373733

[10] Anas Bodor, Meriem Hnida, and Daoudi Najima. 2023. MLOps: Overview of Current State and Future Directions. In *Innovations in Smart Cities Applications Volume 6*. Springer, Cham, 156–165. https://doi.org/10.1007/978-3-031-26852-6_14 ISSN: 2367-3389.

[11] Antonio M. Burgueño-Romero, Cristóbal Barba-González, and José F. Aldana-Montes. 2025. Big Data-driven MLOps workflow for annual high-resolution land cover classification models. *Future Generation Computer Systems* 163 (Feb. 2025), 107499. https://doi.org/10.1016/j.future.2024.107499

[12] Ji-hyun Cha, Heung-gyun Jeong, Seung-woo Han, Dong-chul Kim, Jung-hun Oh, Seok-hee Hwang, and Byeong-ju Park. 2023. Development of MLOps Platform Based on Power Source Analysis for Considering Manufacturing Environment Changes in Real-Time Processes. In *Human-Computer Interaction*. Springer, Cham, 224–236. https://doi.org/10.1007/978-3-031-35572-1_15 ISSN: 1611-3349.

[13] Swati Choudhary. 2021. Kubernetes-Based Architecture For An On-premises Machine Learning Platform. (2021).

[14] Thomas Davenport and Katie Malone. 2021. Deployment as a Critical Business Data Science Discipline. *Harvard Data Science Review* 3, 1 (feb 10 2021). https://hdsr.mitpress.mit.edu/pub/2fu65ujf.

[15] Daniel Deutsch. 2023. *Machine learning operations – domain analysis, reference architecture, and example implementation / Author Daniel Deutsch, LL.B. (WU). LL.M. (WU).* http://epub.jku.at/obvulihs/8593075

[16] Christof Ebert, Gorka Gallardo, Josune Hernantes, and Nicolas Serrano. 2016. DevOps. *IEEE Software* 33, 3 (2016), 94–100. https://doi.org/10.1109/MS.2016.68

[17] Kanwarpartap Singh Gill, Vatsala Anand, Rahul Chauhan, Ruchira Rawat, and Pao-Ann Hsiung. 2023. Utilization of Kubeflow for Deploying Machine Learning Models Across Several Cloud Providers. In *2023 3rd International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON)*. 1–7. https://doi.org/10.1109/SMARTGENCON60755.2023.10442069

[18] Google Cloud Tech. 2020. Introduction to Kubeflow. https://www.youtube.com/watch?v=cTZArDgbIWw

[19] Kim Hazelwood, Sarah Bird, David Brooks, Soumith Chintala, Utku Diril, Dmytro Dzhulgakov, Mohamed Fawzy, Bill Jia, Yangqing Jia, Aditya Kalro, James Law, Kevin Lee, Jason Lu, Pieter Noordhuis, Misha Smelyanskiy, Liang Xiong, and Xiaodong Wang. 2018. Applied Machine Learning at Facebook: A Datacenter Infrastructure Perspective. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 620–629. https://doi.org/10.1109/HPCA.2018.00059 ISSN: 2378-203X.

[20] Hannes Jämtner and Stefan Brynielsson. 2022. An Empirical Study on AI Workflow Automation for Positioning. (2022).

[21] Barbara Ann Kitchenham and Stuart Charters. 2007. *Guidelines for Performing Systematic Literature Reviews in Software Engineering*. Technical Report EBSE 2007-001. Keele University and Durham University Joint Report, Keele, UK and Durham, UK. https://www.elsevier.com/__data/promis_misc/525444systematicreviewsguide.pdf

[22] Dominik Kreuzberger, Niklas Kühl, and Sebastian Hirschl. 2022. Machine Learning Operations (MLOps): Overview, Definition, and Architecture. https://doi.org/10.48550/arXiv.2205.02302 arXiv:2205.02302 [cs].

[23] Anders Köhler. 2022. *Evaluation of MLOps Tools for Kubernetes : A Rudimentary Comparison Between Open Source Kubeflow, Pachyderm and Polyaxon*. https://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-488601

[24] Yumo Luo. 2023. An Open-Source and Portable MLOps Pipeline for Continuous Training and Continuous Deployment. (2023).

[25] Giulio Mallardi, Fabio Calefato, Luigi Quaranta, and Filippo Lanubile. 2024. An MLOps Approach for Deploying Machine Learning Models in Healthcare Systems. In *2024 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 6832–6837. https://ieeexplore.ieee.org/abstract/document/10822603/?casa_token=5GBnjhwp4KAAAAAA:EOEBUeWoY9ZBRsrZ5ij8AoSypmOyGTqTPEraM4kOitmHiRI-0s4zW9omjkMLOt65AuexQrDa3TwVGsQ

[26] Andres Felipe Varon Maya. [n. d.]. The State of MLOps. ([n. d.]).

[27] Rick Merritt. 2020. What is MLOps? https://blogs.nvidia.com/blog/what-is-mlops/

[28] Zakkarija Micallef. 2025. A Systematic Review of MLOps Tools: Practices, Challenges, and Lessons Learned. https://doi.org/10.5281/zenodo.15459745

[29] Widad El Moutaouakal and Karim Baïna. 2023. Comparative Experimentation of MLOps Power on Microsoft Azure, Amazon Web Services, and Google Cloud Platform. In *2023 IEEE 6th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications (CloudTech)*. 1–8. https://doi.org/10.1109/CloudTech58737.2023.10366138

[30] Sasu Mäkinen. 2021. Designing an open-source cloud-native MLOps pipeline. *University of Helsinki* (2021). https://helda.helsinki.fi/server/api/core/bitstreams/d01f98ef-becf-4329-997d-3ebe70092590/content

[31] Óscar A. Méndez, Jorge Camargo, and Hector Florez. 2025. Machine Learning Operations Applied to Development and Model Provisioning. In *Applied Informatics*, Hector Florez and Hernán Astudillo (Eds.). Vol. 2236. Springer Nature Switzerland, Cham, 73–88. https://doi.org/10.1007/978-3-031-75144-8_6 Series Title: Communications in Computer and Information Science.

[32] Faezeh Amou Najafabadi, Justus Bogner, Ilias Gerostathopoulos, and Patricia Lago. 2024. An Analysis of MLOps Architectures: A Systematic Mapping Study. Vol. 14889. 69–85. https://doi.org/10.1007/978-3-031-70797-1_5 arXiv:2406.19847 [cs].

[33] Moses Openja, Forough Majidi, Foutse Khomh, Bhagya Chembakottu, and Heng Li. 2022. Studying the Practices of Deploying Machine Learning Projects on Docker. In *Proceedings of the 26th International Conference on Evaluation and Assessment in Software Engineering (EASE '22)*. Association for Computing Machinery, New York, NY, USA, 190–200. https://doi.org/10.1145/3530019.3530039

[34] Alessandro Palladini. 2022. *Streamline machine learning projects to production using cutting-edge MLOps best practices on AWS*. laurea. Politecnico di Torino. https://webthesis.biblio.polito.it/22607/

[35] Productdock d.o.o, Nataša Radaković, Ivana Šenk, University of Novi Sad, Faculty of Technical Sciences, Nina Romanić, and Productdock d.o.o. 2023. A MACHINE LEARNING PIPELINE IMPLEMENTATION USING MLOPS AND GITOPS PRINCIPLES. In *19th International Scientific Conference on Industrial Systems*. Faculty of Technical Sciences, 94–99. https://doi.org/10.24867/IS-2023-T2.1-6_08141

[36] Katja-Mari Ratilainen. 2023. Adopting Machine Learning Pipeline in Existing Environment. (2023).

[37] Gilberto Recupito, Fabiano Pecorelli, Gemma Catolino, Sergio Moreschini, Dario Di Nucci, Fabio Palomba, and Damian A. Tamburri. 2022. A Multivocal Literature Review of MLOps Tools and Features. In *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. 84–91. https://doi.org/10.1109/SEAA56994.2022.00021

[38] Philipp Ruf, Manav Madan, Christoph Reich, and Djaffar Ould-Abdeslam. 2021. Demystifying MLOps and Presenting a Recipe for the Selection of Open-Source Tools. *Applied Sciences* 11, 19 (Jan. 2021), 8861. https://doi.org/10.3390/app11198861 Number: 19 Publisher: Multidisciplinary Digital Publishing Institute.

[39] Enrico Salvucci. 2021. MLOps-Standardizing the Machine Learning Workflow. (2021). https://amslaurea.unibo.it/id/eprint/23645/

[40] Luca Scotton. 2021. Engineering framework for scalable machine learning operations. (2021). https://aaltodoc.aalto.fi/items/a1497a44-1c3a-46bf-b76a-c7cba635462c

[41] Ladson Gomes Silva. 2022. A Review on How Machine Learning Operations (MLOps) are Changing the Landscape of Machine Learning Development for Production. (2022).

[42] Afonso Rafael Carvalho Sousa. 2022. Orchestrator selection process for cloud-native machine learning experimentation. (2022).

[43] Matteo Testi. 2024. Machine Learning Operations (MLOps) in Healthcare. (2024). https://www.iris.unicampus.it/handle/20.500.12610/83683 Publisher: Università Campus Bio-Medico.

[44] T Vishwambari and Sonali Agrawal. 2023. Integration of Open-Source Machine Learning Operations Tools into a Single Framework. In *2023 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*. 335–340. https://doi.org/10.1109/ICCCIS60361.2023.10425558

[45] Samar Wazir, Gautam Siddharth Kashyap, and Parag Saxena. 2023. MLOps: A Review. https://doi.org/10.48550/arXiv.2308.10908 arXiv:2308.10908 [cs].

[46] Claes Wohlin. 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*. ACM, London England United Kingdom, 1–10. https://doi.org/10.1145/2601248.2601268

[47] Ting Chun Yau. 2023. *Investigate the challenges and opportunities of MLOps*. https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-324011

[48] Mohammad Zarour, Hamza Alzabut, and Khalid T. Al-Sarayreh. 2025. MLOps best practices, challenges and maturity models: A systematic literature review. *Information and Software Technology* 183 (July 2025), 107733. https://doi.org/10.1016/j.infsof.2025.107733

[49] Yue Zhou, Yue Yu, and Bo Ding. 2020. Towards mlops: A case study of ml pipeline platform. In *2020 International conference on artificial intelligence and computer engineering (ICAICE)*. IEEE, 494–500. https://ieeexplore.ieee.org/abstract/document/9361315/?casa_token=gVu7NAHT9ekAAAAA:dRRrkR3bXuCbJBAJNB3M95gMr9rn64LekVjhlXhl2E8M5hgbYSzM5HaF7ysjw0VLIO3K-ilzXVPW6M8

[50] Iago Águila Cifuentes. 2023. *Design and Development of an MLOps Framework*. Master's thesis. Universitat Politècnica de Catalunya. https://upcommons.upc.edu/handle/2117/395348 Accepted: 2023-10-25T10:32:47Z.

Table 3: Summary of MLOps Tools: Infrastructure and Supporting Service (Orchestrators)

Tool	Benefits	Limitations
Apache Airflow		
	B1 Open source software licensed under Apache License 2.0 [50]	L1 Installation and configuration can be complex in real-world environments [50]
	B2 Provides an intuitive web interface for visualising and monitoring workflows [50]	L2 The recommended installation method is complicated in real-world situations [50]
Kubeflow Pipelines		
	B3 Cloud-agnostic architecture enabling it to be executed on any cloud provider that supports Kubernetes [17] [24] [20] with distributions dedicated for major cloud providers [23].	L3 Difficult to set up with a non-trivial learning curve. [23][42].
	B4 More mature and widespread than its competitors and more specialised for ML when compared with Flyte, Apache Airflow, and so on [20].	L4 "Everything in one package" and hence can feel bloated compared with small, focused solutions aimed at solving specific pain points [42].
	B5 Abstracts away the complexity of dynamically scaling workloads up or down via the Kubernetes engine. scale [13].	L5 Out-of-date documentation for many Kubeflow features [23].
	B6 Full end-to-end MLOps solution (Kubeflow Notebooks, Kubeflow Pipelines, Katib) that is highly customisable, through either KServe or Seldon Core [23] and with seamless TFX integration [49].	
	B7 Offers an easily accessible and configurable Kubeflow UI dashboard [23] [12].	
	B8 Strong security (multi-user isolation) [23].	
ZenML		
	B9 Simple local development (configured in three simple commands) [5].	L6 Incomplete user management features [5].
	B10 Automatic deployment of a local MLflow tracking server [5].	
	B11 Easily expandable to support different orchestration components [5].	
Argo Workflow		
	B12 Workflow definitions enable straightforward artefact storage and transfer between tasks.	L7 Requires developers to manage containerisation complexity.
	B13 Supports fully automated, reproducible, and scalable end-to-end model training, evaluation, and deployment without manual intervention.	L8 Pipeline development and management demand in-depth knowledge of Kubernetes and associated tooling.

Table 4: Summary of MLOps Tools: Managed End-to-End Platforms

Tool	Benefits	Limitations
AzureML	B14 Simple data imports and code-free drag-and-drop tools for data cleaning and transformation [29]. B15 Quick and easy to set up for testing, orchestration, and robust security (network protection, RBAC) [29]. B16 Fully complete security features. Only Microsoft Azure ML provides the ability for users to set up network and data protection policies as well as built-in RBAC features [29].	L9 Commercial product requiring an Azure subscription [29].
SageMaker	B17 Robust and intuitive, offering feature parity with Azure ML [29]. Includes a good set of tools with its own model registry, a feature store and lineage tracking logic. Handles step creations and management as well as automates model deployment with CI/CD [34]. B18 Facilitates easy scalable training and deployment that reduces operation overhead [6]. B19 Seamless integration within the AWS ecosystem [34] such as the ability to integrate with AWS Single Sign-On (SSO) to manage identities and access [29]. B20 Provides abstracted tools for model tuning [34].	L10 Requires a commercial licence [47]. L11 Requires programming skills and has less comprehensive documentation [47]. L12 If an organisation does not already use AWS IAM or SSO, additional adoption and migration work is required [29].
Google Vertex AI	B21 Offers AutoML, integrates Jupyter Notebook in Vertex AI Workbench and provides its own model registry and pipelines. Integrates monitoring and logging capabilities [41]. B22 Easy to use central UI dashboard with visualised workflows where users can monitor the progress, review the execution history, and understand the dependencies between different components of the pipeline [8]. B23 Flexible scalability and easy to use Kubernetes cluster with a choice of machine hardware [8].	L13 Dataset preparation tasks are considered the most "painful", with large multi-step coding tasks without a visual drag-and-drop interface when compared to Azure and AWS Sagemaker [29]. L14 Lacks robust security control measures (IAM and workspace separation) [29]. L15 Operates on a commercial, pay-as-you-go model.

Table 5: Summary of MLOps Tools: End-to-End Platforms

Tool	Benefits	Limitations
MLflow	B24 User-friendly with a UI providing visualisation capabilities for easy data interpretation and analysis, along with an intuitive web-based dashboard [31][36][25]. B25 Completely open-source and self-hosting capabilities while natively supporting container deployments [5][40]. B26 Strong vitality, since it is a well-established tool used by many world-renowned companies, backed by strong community support [5][11]. B27 Offers comprehensive and extensive documentation [11]. B28 Comprehensive Experiment Tracking with robust logging and visualisation of metrics and artefacts, which allows users to trace models to their training rounds via MLflow APIs/UI and to track algorithms, hyperparameters, dataset versions, and feature selections [31][4][49]. It is MLFow's key feature and module. B29 Simplifies performance evaluation with enhanced model performance monitoring. Moreover, it features autologging and seamlessly integrates with other monitoring tools for efficient model improvement [5][9][15][31]. B30 Flexible data storage options with support for popular cloud storage services (Amazon S3, Azure Blob Storage, Google Cloud Storage) as well as on-premise or hybrid options (SFTP, NFS), local files, SQLAlchemy-compatible databases, or remote tracking servers [36].	L16 Requires a dedicated server or additional web service for collaboration, which complicates initial setup and maintenance [4][40]. L17 Does not automatically reproduce data versions, necessitating manual intervention for consistency [36][4]. Relies on the user using other tools. L18 Lacks in-built alerting based on monitoring for insufficient resources [40]. L19 Does not offer role-based access or user isolation, allowing unrestricted changes to experiments by any user. [5][36]. L20 Lacks native collaboration features and automated deployment tools [36]. L21 Certain advanced features or enterprise use cases require a commercial licence [47].
Neptune	B31 Easy setup with a clear guide [47] and integrates with tools like Google Colab, Git, and Docker [43]. B32 Complete MLOps platform: monitoring, data versioning, and testing in one solution [43]. B33 Open-source [47]. B34 Can be used for monitoring [43].	L22 Requires basic programming skills [43].
Pachyderm	B35 Provides GUI-accessible services for notebook experimentation, experiment tracking, and hyperparameter optimisation [23]. B36 Simple to parallelise experiments [23].	L23 Requires knowledge of, and manual configuration for, cloud storage and Helm deployments [23]. L24 Entails a computationally heavy file system overhead compared with Kubeflow and Polyaxon [23]. L25 Lacks "out-of-the-box" solutions.
Polyaxon	B37 Requires only basic Helm and Kubernetes operator knowledge [23]. B38 Provides a meticulous audit trail that enhances reproducibility [23].	L26 Does not provide out-of-the-box scaling [24]. L27 Uncertain open-source vitality due to a weaker community compared to Pachyderm and Kubeflow [23].
TFX	B39 Automates regular retraining, evaluation, and deployment [8]. B40 Supports distributed computing for large workloads [8]. B41 Portable and can be run on various orchestration platforms such as Apache Airflow [8].	L28 Requires TensorFlow, which might be limiting for non-TF use cases [20].

Table 6: Summary of MLOps Tools: Storage & Versioning

Tool	Benefits	Limitations
Dagshub	B42 Facilitates collaboration and versioning [31]. B43 Circumvents GitHub's size limitations and integrates with DVC and MLflow [31][15]. B44 Offers a free integrated MLflow server and unified storage for data and metadata [2][15].	- -
DVC	B45 Easy to use as its DVC workflow is similar to Git's. This helps ease its adoption among users familiar with Git [4]. B46 Modularises workflows into stages, ensuring reproducibility and reducing manual errors [4]. B47 Pipeline data can be automatically pulled from a DagsHub repository so that the entire process can be run using only a single command, thus simplifying workflow execution [33]. B48 Caching for unchanged stages saves time and minimises overhead [35]. B49 Flexible storage options (e.g., Google Drive, HTTP, S3) without requiring a dedicated server or GitLFServer [4].	L29 Requires manual updates for externally stored data [4]. L30 File-level versioning can lead to extensive storage use in environments with frequent file changes and is not suited for versioning SQL databases [4].
GTO	B50 Functions as a GitOps-based model registry, removing the need for separate databases or servers by leveraging Git and DVC [35]. B51 Supports promotion of models to specific stages, facilitating deployment across designated environments [35]. B52 Seamlessly integrates with Iterative products such as DVC and CML [35].	- -
MinIO	B53 S3-compatible storage solution optimised for AI workloads [5]. B54 Used for storing artefacts and metadata generated by ZenML [43] and MLflow [4], and suitable for general data.	- -
Weights & Biases	B55 Real-time experiment tracking with built-in hardware usage monitoring [5]. B56 Free tier offers unlimited experiment runs and 100 GB of artefact storage. In addition, it includes dataset/model versioning, hyperparameter optimisation, report generation, and a model registry [5].	L31 Requires registration and a licence key for initial setup. The free plan is limited to personal projects [5]. L32 Only the client application is open-source. The server-side infrastructure is proprietary, limiting self-hosting options [5]. L33 Demands advanced programming knowledge for integration and usage [43][5].
Feast	B57 Manages historical and live data with offline/online feature storage [44]. B58 Allows reuse of features across different ML projects [47]. B59 Integrates with MLflow [44]. B60 Open-source [47].	L34 Requires advanced programming skills [47]. L35 Integration process is relatively complicated [47].

Table 7: Summary of MLOps Tools: Inference (Model Serving)

Tool	Benefits	Limitations
BentoML	B61 Open-source [40] and supports multiple frameworks (TF, PyTorch, Keras, XGBoost) [39]. B62 Features automated micro-batching for better API performance and cloud-native deployment [40].	L36 Provides inference-only [40] and does not offer an automatic deployment of models, as KServe and Seldon Core do [24][20].
KServe	B63 Automatically wraps models as web services and is easily integrated with Kubeflow [24]. B64 Provides scalable and isolated deployments [17] with support for HTTP/gRPC APIs [24].	- -
Seldon Core	B65 Offers pre-packaged inference servers with robust Kubernetes support [11]. B66 Supports advanced metrics tracking via Prometheus [11].	- -
Evidently	B67 Provides model monitoring for data, target, and prediction drifts [2][44]. B68 Predefined monitoring reports can be generated with few lines of code [2]. B69 Enables a single monitor to calculate metrics across multiple models [24]. B70 Integrates with Prometheus and Grafana for interactive reports, scheduled tests, and result logging [44]. B71 Automatically logs results as artefacts in MLflow [44].	L37 Supports tabular data only. For image or text data, consider Alibi Detect [24].

Table 8: Summary of MLOps Tools: Inference (Model Deployment)

Tool	Benefits	Limitations
Metaflow	B72 Provides a framework for creating and executing data science workflows locally and scaling to the cloud with ease [40]. B73 Rigorous checkpoint system enables great tracking and logging [40]. B74 Fully integrated with AWS for automatic resource management and native parallelisation via AWS Batch [40].	- -
Streamlit	B75 Simple interface for deploying ML applications to the cloud [2]. B76 Convenient GitHub integration streamlines deployment workflows [2].	L38 1 GB limit for public application deployments [2].
CML	B77 Manages ML experiments and tracks modifications automatically [35]. B78 Generates comprehensive reports with essential metrics and plots [35]. B79 Reports are created and displayed directly in pull request comments, enhancing collaboration and review efficiency [35].	- -

Table 9: Summary of MLOps Tools: Infrastructure and Supporting Services

Tool	Benefits	Limitations
Gradio	B80 Enables creation of interactive web interfaces for model evaluation, demonstration, and deployment without HTML/CSS/JS knowledge [15].	- -
TensorBoard	B81 Visualisation toolkit for model graphs, weight/bias histograms, and training metrics [47]. B82 Provides full exploration and visualisation functionality [47]. B83 Open-source and integrates with multiple tools and applications [47].	L39 Requires familiarity with TensorBoard tooling and community support for effective use [47].

Table 10: Summary of MLOps Tools: ML Training

Tool	Benefits	Limitations
Katib	B84 Optimises hyperparameters for frameworks such as TensorFlow, MXNet, PyTorch, and XGBoost [17]. B85 Seamlessly integrates with Kubeflow [17].	- -