Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 10, October - 2025

MLOps: Changes and Tools - Adapting Machine Learning Workflows for Enhanced Efficiency and Scalability

Mysaa Khaled Alrahal
Syrian Virtual University

Dr. Sira Astour
Arab International University

***Abstract* - This study presents an analytical and practical investigation of Machine Learning Operations (MLOps) processes, with a particular focus on experiment tracking and reproducibility. An open-source, end-to-end MLOps pipeline was proposed, encompassing data preparation, training, tracking, deployment, and preliminary post-production monitoring. The research utilized DVC, MLflow, and Weights & Biases (W&B) to ensure proper versioning and logging of configurations and metrics and compared their performance to conventional non-automated workflows. A CI/CD pipeline was implemented using GitHub Actions, with automated deployment through Docker and FastAPI.**

Experimental results demonstrated that integrating tracking and automation tools significantly improved model accuracy, reduced training and deployment time, and minimized operational errors — ultimately enhancing model reliability in both academic and production environments.

Keywords: MLOps, MLOps Pipeline, Experiment Tracking, CI/CD, DevOps, Versioning, Machine Learning, Artificial Intelligence

1. INTRODUCTION

In the era of rapid digital transformation and the growing reliance on Artificial Intelligence (AI) and Machine Learning (ML) technologies in both research and industrial applications, ensuring the efficiency and sustainability of model life cycles has become increasingly critical. [1], [2] Despite remarkable progress in algorithmic development and accuracy enhancement, significant challenges persist in data management, experiment tracking, and deployment in production environments. [3], [4]

To address these issues, Machine Learning Operations (MLOps) has emerged as an extension of DevOps practices, integrating data management, automation, and continuous model monitoring to achieve transparency, reproducibility, and scalability. [2], [3] Nevertheless, as reported in recent studies, several persistent gaps and implementation challenges remain — particularly in aligning tools, workflows, and team collaboration — which necessitate deeper analysis and applied research. [1], [5], [6]

Although AI and ML have advanced considerably, most traditional research and application environments still face fundamental limitations. These include difficulties in managing datasets and models, the absence of effective version control mechanisms, low operational efficiency due to manual training and deployment processes, and poor scalability when models are transferred across multiple servers or cloud environments. [2], [3], [4] Furthermore, experiment reproducibility often suffers; when code or data change, prior results become irreproducible, undermining scientific reliability. [3], [7] Another recurring issue is the lack of integration between development and data analysis teams, leading to the common “works on my machine” problem. [2], [3]

Manual model deployment also remains prevalent in academic projects — training and evaluation are typically completed, but deployment is either manual or neglected entirely, resulting in high error rates and delayed updates. [8] In production environments, the gap between development and operational contexts often leads to failures when models are transitioned from one environment to another. [3] Moreover, post-deployment monitoring is frequently overlooked, causing gradual degradation in model performance due to data drift and changing operational conditions. [9]

These challenges have motivated numerous researchers to enhance MLOps practices and improve interoperability between tools that often differ in design and configuration. [8], [6] This study contributes to these efforts by focusing specifically on experiment tracking — a central yet underexplored component of the MLOps workflow — and by analyzing its role in ensuring transparency, reproducibility, and scientific reliability. [10], [11]

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 10, October - 2025

The main objective of this research is to evaluate the impact of MLOps techniques on improving the efficiency of ML model life cycles by proposing and implementing a fully integrated and open-source pipeline that supports reproducibility, traceability, and automated deployment. The study compares open-source tools such as DVC, MLflow, and Weights & Biases (W&B) in terms of experiment tracking and model performance analysis, benchmarking them against conventional non-automated workflows.

This research fills an important gap by highlighting the relationship between experiment tracking and subsequent stages in the MLOps life cycle. It provides a practical, modular framework that can be leveraged in both academic and industrial contexts to enhance model efficiency, scalability, and transparency.

Empirically, the study is among the first to conduct a comparative evaluation of two open-source tracking tools — MLflow and W&B — across textual (TF-IDF + Logistic Regression) and image-based (CNN–CIFAR10) experiments. Results revealed comparable statistical performance (Accuracy and F1-score) across both environments, with MLflow demonstrating faster execution (≈11 s for text, ≈144 s for images) due to local storage efficiency, while W&B offered superior collaborative and analytical capabilities despite higher synchronization latency (≈26–42 s for text, ≈216 s for images).

The integration of these tools with CI/CD pipelines using GitHub Actions, Docker, and FastAPI successfully enabled an end-to-end MLOps workflow — from data preparation and training to automated deployment and post-deployment monitoring. Consequently, the proposed framework addresses a key research gap by emphasizing the operational value of experiment tracking and providing a reproducible, extensible MLOps model suitable for real-world use.

2. RELATED WORK

MLOps practices are rapidly evolving from isolated laboratory prototypes into production-grade, socio-technical ecosystems that intersect with data tooling, cloud platforms, governance, and regulatory compliance. Early work foregrounded sustainability and long-term maintainability as first-class concerns in AI software, warning that unchecked platform complexity and technical debt can undermine both social and organizational sustainability. [12] Subsequent systematic mappings consolidated best practices, recurring challenges, and maturity models, revealing the need for standardization across the MLOps lifecycle and for reliable mechanisms that ensure end-to-end traceability and reproducibility. [9]

To bridge academic and industrial viewpoints, Steidl et al. provided a unifying lens over terms such as AI-DevOps, CI/CD for AI, MLOps, model lifecycle management, and CD4ML, and organized triggers and pipeline stages into a coherent hierarchy (data processing, model learning, software development, and system operations). [13] Complementing process-centric views, Faubel and Schmid systematically analyzed MLOps platforms and architectural tactics using attribute-driven design to map capabilities to platform choices. [14] A broader landscape study further compared 16 widely used platforms with a three-step evaluation, yielding decision guidance for end-to-end versus specialized stacks and highlighting gaps in production monitoring and drift detection. [15] In healthcare, a scoping review distilled thematic areas that stress continuous monitoring, re-training, ethics, clinical integration, infrastructure, compliance, and feasibility. [16] Large-scale geospatial pipelines demonstrated how container orchestration (Kubernetes) enables scalable, repeatable updates over massive satellite imagery. [17]

Experiment tracking remains pivotal: Budras et al. surveyed 20 tools and analyzed four in depth. [10] Matthew highlighted versioning and reproducibility at scale. [7] Safri, Papadimitriou, and Deelman argued for dynamic tracking integrated with workflow systems. [11] Applied templates that codify provenance, logging, configuration management, and CI/CD offer pragmatic on-ramps. [18] Collectively, prior work establishes the need for standardization and sustainability, shared vocabulary and staged pipelines, architecture-level guidance, domain constraints, and the centrality of tracking. Yet a quantitative link between tracking choices and operational outcomes within a single, reproducible pipeline is still scarce.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 10, October - 2025

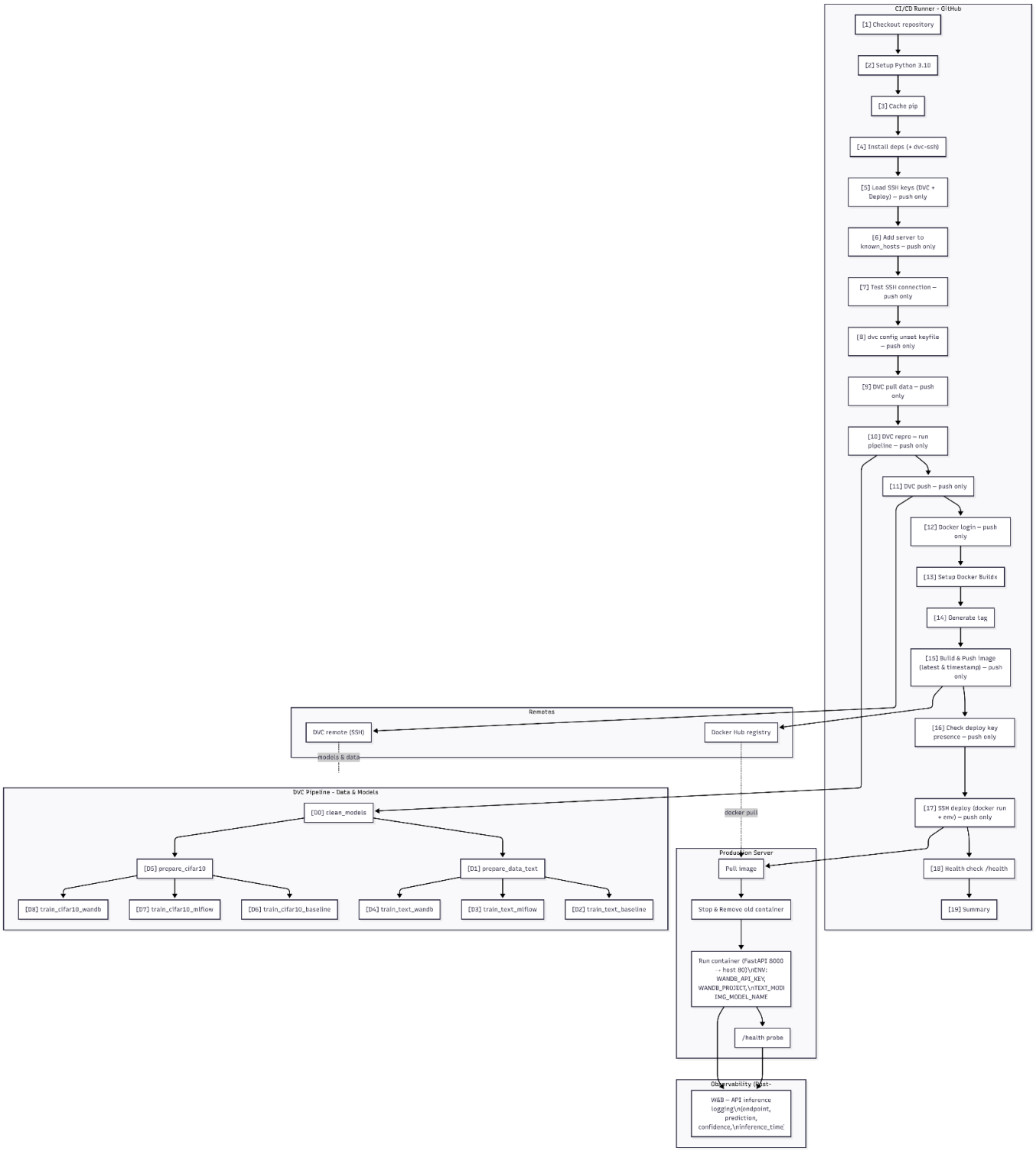
Table 1. Comparative positioning against Matthew (2025) [7]

Dimension	Matthew (2025) [7]	This Study
Methodology	Descriptive/theoretical framing of versioning & reproducibility with high-level examples.	Field implementation of a fully reproducible, open-source pipeline (DVC, MLflow, W&B, GitHub Actions, Docker, FastAPI).
Application Scope	Broad focus on large-scale industrial projects; no concrete experimental pipeline.	Dual case study (text + images) to demonstrate generality across data modalities.
Focus	Version control & reproducibility as foundational concerns.	Practical impact of experiment tracking within an integrated lifecycle covering versioning, deployment, and monitoring.
Tooling	Discusses tools primarily as exemplars.	Quantitative head-to-head analysis of MLflow vs. W&B.
Technical Stack	No implemented CI/CD architecture.	Concrete CI/CD with GitHub Actions and Docker integrated with tracking and serving.
Scientific Value	Organizational guidance for the reproducibility crisis.	Applicable, benchmarked pipeline and measurement framework (operational trade-offs).

3. PROPOSED MLOPS PIPELINE

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 10, October - 2025

Figure 1: pipeline architecture



3.1 Architectural Overview

We propose an end-to-end, open-source MLOps pipeline spanning five phases—design & planning, experimentation & development, deployment, monitoring & maintenance, and continuous improvement. The pipeline orchestrates DVC stages (text and image), experiment tracking via MLflow and W&B, automated builds/releases with Docker and GitHub Actions, and online serving through FastAPI.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 10, October - 2025

Key assets:
• DVC stages (text + vision): clean_models, prepare_data_text, train_text_baseline, train_text_mlflow, train_text_wandb, prepare_cifar10, train_cifar10_baseline, train_cifar10_mlflow, train_cifar10_wandb.
• Serving: FastAPI container on port 80.
• Tracking: MLflow (local artifact store) and W&B (hosted dashboards).
• CI/CD: GitHub Actions; Docker Buildx for caching and multi-arch builds.
• Remote storage: DVC-Remote (SSH) for versioned datasets/models.

3.2 Five-Phase Workflow (19 Steps)

P1 — Design & Planning

Step 1. Repository checkout. Retrieve the repository and pipeline descriptors (YAML/DVC) as the single source of truth.

P2 — Experimentation & Development

Step 2. Set up Python 3.10. Interpreter pinning for deterministic runs.

Step 3. Cache pip. Speeds up repeated CI executions.

Step 4. Install dependencies (+ dvc-ssh). Ensure ML/data/DVC tooling; dvc-ssh enables secure DVC remote access.

Step 8. DVC config unset keyfile. Remove stale local keyfile paths to enforce CI SSH credentials.

Step 9. DVC pull. Fetch the exact versioned datasets/models.

Step 10. DVC repro. Incrementally reproduce affected stages:

– Prepare text & image data;

– Train models under Baseline, MLflow, and W&B backends.

Step 11. DVC push. Persist newly produced artifacts/models to the remote.

Outcome: Trained, versioned models under models/, ready for packaging.

P3 — Deployment

Step 5. Load SSH keys. Secure access to DVC remote and production hosts via GitHub Secrets.

Step 6. Add server to known_hosts. Enforce host authenticity; avoid interactive prompts.

Step 7. Test SSH connectivity. Early failure detection.

Step 12. Docker login. Authenticate to the container registry.

Step 13. Setup Docker Buildx. Advanced caching / multi-arch builds.

Step 14. Generate timestamped tag. Publish :latest and a precise time-based tag (e.g., :2025YYYYMMDDhhmmss).

Step 15. Build & push image. Bundle code, deps, and models; push to the registry.

Step 16. Check deploy key. Sanity-check presence/permissions before release.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 10, October - 2025

Step 17. SSH deploy (docker run + env vars). Pull & run the new image with required env (incl. W&B); expose FastAPI on port 80.

P4 — Monitoring & Maintenance

Step 18. Health check (/health). Verify readiness/liveness post-rollout.

Online telemetry to W&B. Each API inference logs endpoint, prediction, confidence, inference_time for lightweight post-deployment monitoring.

P5 — Continuous Improvement

Step 19. Analyze W&B metrics → optimize → retrain via DVC. If accuracy/latency degrades, adjust and re-run (10) dvc repro to produce a new versioned model—closing the learning loop.

3.3 DVC Pipeline Stages and Artifacts

Text: prepare_data_text → train_text_baseline | train_text_mlflow | train_text_wandb; Vision: prepare_cifar10 → train_cifar10_baseline | train_cifar10_mlflow | train_cifar10_wandb; Housekeeping: clean_models. Each stage declares inputs/outputs and params enabling selective recomputation and exact provenance.

3.4 CI/CD and Release Engineering

A single GitHub Actions workflow implements build–test–package–deploy with Python 3.10, dependency caching, and DVC remote integration; Docker release with dual tagging (latest + timestamp); zero-touch rollout via SSH with health verification.

3.5 Production Telemetry and Early-Warning Signals

FastAPI logs minimal inference metadata to W&B (endpoint, predicted label, confidence, latency) supporting quality monitoring, SLOs, and auditability.

3.6 Continuous-Learning Loop

Monitoring insights drive targeted retraining. Using DVC versioning, we re-run repro with updated data/params; new artifacts receive fresh hashes and image tags.

3.7 Reproducibility, Security, and Compliance Notes

• Reproducibility: pinned interpreter, locked deps, DVC-tracked data/models, immutable images, dual tags.

• Security: SSH keys in GitHub Secrets; known_hosts pinning; runtime-injected env vars.
• Governance: verifiable lineage from commit → params → data snapshot → model hash → image tag → production metrics.

4. EXPERIMENTS AND RESULTS

4.1 Text Classification (TF-IDF + Logistic Regression)

A labeled text dataset (text, label) was processed with TF-IDF; stratified 80/20 split. Logistic Regression served as the baseline classifier.

Table 2. Text model performance

Tracking	Accuracy	F1-macro	F1-weighted
None	0.855	0.853	0.854
MLflow	0.855	0.853	0.854
W&B	0.855	0.853	0.854

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 10, October - 2025

Table 3. Text model timing (train/predict)

Tracking	Train Time (s)	Predict Time (s)
None	~0.51	~0.014
MLflow	~0.64	~0.015
W&B	~0.51	~0.020

Table 4. Text model total runtime

Tracking	Total Runtime (code)	Runtime (tool UI)
None	~0.55 s	—
MLflow	~11.5 s	~10.6 s
W&B	~26.8 s	~42 s

4.2 Image Classification (CNN on CIFAR-10)

CIFAR-10 (60k images, 10 classes) was normalized to [0,1] and stored as NPZ for faster loads. A small CNN was trained under three tracking conditions.

Table 5. Vision model performance

Tracking	Accuracy	Test Loss
None	0.6767	0.9366
MLflow	0.6804	0.9288
W&B	0.6704	0.9528

Table 6. Vision model timing (train/predict)

Tracking	Train Time (s)	Predict Time (s)
None	~124	~2.62
MLflow	~133	~2.82
W&B	~146	~2.77

Table 7. Vision model total runtime

Tracking	Total Runtime (code)	Runtime (tool UI)
None	~127 s	—
MLflow	~144 s	≈ 2.7 min (162 s)
W&B	~149 s	≈ 3 min 36 s (216 s)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 10, October - 2025

4.3 Summary

Predictive metrics were identical across tracking tools, indicating that tracking does not affect model quality when model/data are fixed. MLflow was faster locally with disk-based artifacts; W&B introduced upload latency yet enabled rich, collaborative dashboards.

5. CONCLUSION AND FUTURE WORK

5.1 Summary of Findings

This study delivered a reproducible, automated MLOps pipeline using DVC, MLflow, W&B, Docker, FastAPI, and GitHub Actions. Outcomes include reproducible runs, reduced operational burden, collaborative sharing, and CI/CD-integrated deployment. On text and vision tasks, statistical metrics matched across tracking tools; operational trade-offs favored MLflow locally and W&B for collaboration.

5.2 Future Research Directions

- Post-deployment drift and bias detection (Fairlearn, AIF360).
- Explainability integration (SHAP, LIME) linked to tracking runs.
- Unified tracking middleware (merge MLflow & W&B metadata).
- Self-adaptive MLOps (auto re-training on monitored degradation).
- Temporal performance auditing (standardize runtime/latency logs).
- Multi-modal pipelines (text+image) with dual-tracking assessment.

5.3 Concluding Remark

Well-structured MLOps practices measurably improve reliability, transparency, and deployment velocity. Choosing the right tool mix early—balancing speed (MLflow) and collaboration (W&B)—is decisive for long-term success.

REFERENCES :

[1] J. Díaz-de-Arcaya, A. I. Torre-Bastida, G. Zárate, R. Miñón, and A. Almeida, “A Joint Study of the Challenges, Opportunities, and Roadmap of MLOps and AIOps: A Systematic Survey,” ACM Computing Surveys, vol. 56, no. 4, pp. 1–30, 2023.
[2] A. Singla, “Machine Learning Operations (MLOps): Challenges and Strategies,” Journal of Knowledge Learning and Science Technology, vol. 2, no. 3, pp. 334–342, 2023.
[3] D. Kreuzberger, N. Kühl, and S. Hirschl, “Machine Learning Operations (MLOps): Overview, Definition, and Architecture,” IEEE Access, vol. 11, pp. 31866–31879, 2023.
[4] S. Wazir, G. S. Kashyap, and P. Saxena, “MLOps: A Review,” arXiv preprint arXiv:2308.10908, 2023. [Online]. Available: https://arxiv.org/abs/2308.10908
[5] S. J. Warnett and U. Zdun, “On the Understandability of MLOps System Architectures,” IEEE Trans. Software Eng., vol. 50, no. 5, pp. 1015–1039, May 2024.
[6] D. O. Hanchuk and S. O. Semerikov, “Automating Machine Learning: A Meta-Synthesis of MLOps Tools, Frameworks and Architectures,” in Proc. 7th Workshop for Young Scientists in Computer Science & Software Engineering (CS&SE@SW 2024), CEUR Workshop Proc., vol. 3917, pp. 362–414, 2025.
[7] B. Matthew, “Model Versioning and Reproducibility Challenges in Large-Scale ML Projects,” Advances in Engineering Software, 2025.
[8] P. Liang, B. Song, X. Zhan, Z. Chen, and J. Yuan, “Automating the Training and Deployment of Models in MLOps by Integrating Systems with Machine Learning,” Applied and Computational Engineering, vol. 76, pp. 1–7, 2024.
[9] M. I. Zarour, H. Alzabut, and K. Alsarayrah, “MLOps Best Practices, Challenges and Maturity Models: A Systematic Literature Review,” Information and Software Technology, vol. 183, Article 107733, 2025.
[10] T. Budras, M. Blanck, T. Berger, and A. Schmidt, “An In-depth Comparison of Experiment Tracking Tools for Machine Learning Applications,” International Journal on Advances in Software, vol. 15, no. 3&4, 2022.
[11] H. Safri, G. Papadimitriou, and E. Deelman, “Dynamic Tracking, MLOps, and Workflow Integration: Enabling Transparent Reproducibility in Machine Learning,” in Proc. 20th IEEE Int. Conf. on e-Science, pp. 1–10, 2024.
[12] D. A. Tamburri, “Sustainable MLOps: Trends and Challenges,” in Proc. 22nd Int. Symp. on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2020), 2020, pp. 17–23.
[13] M. Steidl, M. Felderer, and R. Ramler, “The Pipeline for the Continuous Development of Artificial Intelligence Models – Current State of Research and Practice,” Journal of Systems and Software, vol. 192, p. 111615, 2023.
[14] L. Faubel and K. Schmid, “A Systematic Analysis of MLOps Features and Platforms,” in Proc. 2024 Int. Conf. on Data Science and Software Engineering Applications (DSD/SEAA – Works in Progress Session), Aug. 2024
[15] L. Berberi et al., “Machine Learning Operations Landscape: Platforms and Tools,” Artificial Intelligence Review, vol. 58, Article 167, 2025.
[16] A. Rajagopal et al., “Machine Learning Operations in Health Care: A Scoping Review,” Mayo Clinic Proceedings: Digital Health, vol. 2, no. 3, pp. 421–437, 2024.
[17] A. M. Burgueño-Romero, C. Barba-González, and J. F. Aldana-Montes, “Big Data-driven MLOps Workflow for Annual High-Resolution Land Cover Classification Models,” Future Generation Computer Systems, vol. 163, Article 107499, 2025.
[18] R. C. Godwin and R. L. Melvin, “Toward Efficient Data Science: A Comprehensive MLOps Template for Collaborative Code Development and Automation,” SoftwareX, vol. 26, Article 101723, 2024.