STM JOURNALS

Journal of
Advances in Shell Programming

ISSN: 2395-6690
Volume 12, Issue 2, 2025
May–August
DOI (Journal): 10.37591/JoASP

https://journals.stmjournals.com/joasp

Review JoASP

Machine Learning Pipelines: A Survey on Automation, Scalability, and Deployment Strategies

Sandeep Gupta*

Abstract

Machine learning (ML) has become a critical enabler of intelligent applications across domains, requiring robust, efficient, and scalable deployment workflows. This review paper provides an in-depth overview of machine learning pipelines, emphasizing three key dimensions: automation, scalability, and deployment methodologies. It begins by exploring automation techniques that reduce manual effort in data ingestion, preprocessing, model selection, and hyperparameter tuning. Tools such as AutoML, TFX, and workflow orchestration platforms are examined for their role in streamlining repetitive tasks and improving consistency. The paper then delves into the scalability of ML pipelines, addressing horizontal and vertical scaling, resource optimization using Kubernetes and ML-driven autoscores, and the integration of real-time and streaming architectures. Finally, investigated modern deployment strategies, including model serving frameworks, CI/CD pipelines for continuous updates, and edge-hybrid architectures that ensure low latency, data privacy, and efficient resource use. The paper synthesizes current research trends, highlights challenges in integrating modular, scalable components, and discusses opportunities for enhancing MLOps practices. By surveying recent advancements and limitations, this work provides researchers and practitioners with valuable insights to design resilient and future-ready ML pipeline architectures suited for production environments.

Keywords: Machine learning pipelines, MLOps, automation, scalability, deployment strategies, CI/CD, edge computing

INTRODUCTION

Machine learning (ML) changed the way data is used in every industry, catalyzed business efficiency and redefined the advertising ecosystem as well as healthcare technologies. ML has now gotten intertwined with a variety of applications and services in various fields [1, 2]. As the management and implementation of ML has become more complex, the requirement for a strong operational framework has increased, resulting in Machine Learning Operations (MLOps).

MLOps is a combination of ML, the whole production lifetime of ML systems is managed using DevOps and data engineering concepts. Data preparation, training, validation, deployment, monitoring, and governance are all part of this [3]. MLOps allows data scientists, ML engineers, and IT operations to have a smooth workflow to integrate ML into enterprise applications, to improve flexibility and innovation [4]. Continuous integration and continuous deployment (CI/CD) originally developed to fit into the traditional software development process has been modified to suit the ML workflow. As such, it provides consistent testing and validation of models as well as ensuring that models are deployed consistently, and reduces any manual errors and shortened time to market [5].

***Author for Correspondence**
Sandeep Gupta
E-mail: Sandeepguptabashu@gmail.com

Senior Researcher, Department of Computer Science Engineering, Samrat Ashok Technological Institute (SATI), Vidisha, Madhya Pradesh, India

Received Date: July 20, 2025
Accepted Date: July 24, 2025
Published Date: September 08, 2025

Citation: Sandeep Gupta. Machine Learning Pipelines: A Survey on Automation, Scalability, and Deployment Strategies. Journal of Advances in Shell Programming. 2025; 12(2): 17–28p.

The importance of automation in any modern ML pipeline is the ability to produce repeatable, durable, as well as modular workflows that have a minimum amount of human input [6]. Orchestration systems and end-to-end Auto ML frameworks allow automating the shift across various stages of pre-processing, training, and serving. The most typical problems that this automation solves are manual retraining, brittle scripts, and convoluted pipelines, ensuring quick experimentation and effective production cycles [7].

The sheer increase in the amount of data and dependence on distributed systems have made modularity and scalability important aspects of pipeline architecture. The microservice-based tools and architecture, containerized, serverless computing, and cloud-edge orchestration-based tools and architecture provide scalable solutions, which can scale with changing workloads [8, 9]. In spite of this, current research has highlighted that other bottlenecks exist in the context of data engineering, pipeline coordination, and system maintainability, in which architectural designs must be able to support flexibility but in a manner that would not affect performance and/or the resources deployed related to FPGAs.

Experiment tracking (including monitoring features of the model serving), and modular inference components start to appear in modern MLOps platforms. Scalably implementing these features under the organizational infrastructure is a very complicated undertaking [10, 11]. The solution to this concurrence may be focused on a series of key dimensions: automation, Auto ML, orchestration, scale, cloud-native and edge-side computing and deployment schemes of CI/CD pipelines, monitoring, and secure rollouts [12].

Structure of the Paper

The structure of this paper is as follows: The next section outlines the fundamentals and automation of ML pipelines. The subsequent section covers automation and scalability techniques. The following section discusses deployment strategies, including CI/CD and edge computing. The ensuing section reviews recent literature, and the final section concludes with future research directions.

FUNDAMENTALS AND AUTOMATION IN MACHINE LEARNING PIPELINES

ML pipelines are structured, automated workflows that streamline and manage the end-to-end process of building, training, deploying, and maintaining ML models. A distinct function within the ML lifecycle. These components are designed to be reusable and independently testable, enabling flexible development, consistent integration, consisting of a sequence of modular, interconnected steps, including gathering data, preprocessing, creating features, training models, assessing them, and implementing them, each performing, and efficient scaling. By encapsulating the entire ML workflow into a reproducible and modular framework, pipelines facilitate robust model development and operationalization across various applications.

Automation is a critical enabler of scalable and efficient ML pipelines. It streamlines repetitive and resource-intensive tasks such as data ingestion, preprocessing, model selection, hyperparameter tuning, and workflow orchestration. Manual intervention becomes minimal with the use of such tools as Apache NiFi, TFX, and Auto ML frameworks, which provide more consistency and have a faster deployment. Automation also provides high-quality data and enables complex workflows through containerized microservices and dynamic model updates in real-time applications like the IoT. The consideration of this section is to survey the development and usage of automation technologies in different ML pipelines with the focus on the effect on the reliability, scalability as well as integration of the automation in MLOps environments.

Automated Data Ingestion and Preprocessing

The two important procedures in the development of ML scalable pipelines are data ingestion and pre-processing. Ingestion is a perpetual activity of the collection of data in APIs, databases and cloud storage, and they are typically managed with software, often Apache NiFi or AWS Glue [13, 14]. The

pre-processing steps include cleaning, normalizing, encoding and feature extraction that can also be automated at the platforms such as TFX or Kubeflow Pipelines. The major benefits include the following:

- *Reduced Manual Effort:* Reduces the time taken to do data duplicates.
- *Scalable Ingestion:* Tools like Apache NiFi, AWS Glue, and Azure Data Factory handle large and varied data sources.
- *ML4IoT:* A Structure for coordinating ML processes on IoT information.
- *Improved Data Quality:* Schema preservation and finding of missing or inconsistent values are the roles of automated validation checks.
- *Faster Model Deployment:* Speeds up end-to-end pipeline execution, leading to quicker production readiness.

Such automated processes are the core of MLOps workflows so that teams ensure effective, efficient, and production-driven machine learning systems.

Model Selection and Hyperparameter Tuning

The parameters that control the structure and training procedure of ML models are known as hyperparameters. The process of determining the ideal collection of hyperparameters that provides a ML model with the highest performance is known as hyperparameter tuning [15]. This procedure is crucial because hyperparameters directly affect the model's performance by controlling the learning process and its structure, such as learning rate, neural network neuron count, or support vector machine kernel size.

The fundamental idea underlying auto-ML approaches is to train many models simultaneously using the same data set, then choose the best model by developing a successful assessment plan [16]. This includes several processes that are applied one after the other, as shown in Figure 1.

The four main steps of an Auto ML pipeline are feature engineering, model development, data preparation, and model validation. Each of the aforementioned phases entails a number of tasks, some of which are created and applied in a particular application.

Workflow Orchestration

In order to exploit online IoT data to infer new information, trained ML models are deployed utilizing the online ML process. The online datasets, trained ML models, and deployment settings needed to carry out this kind of process must be chosen and configured.

ML workflows are modeled as a series of stages that a containerized microservice may carry out. The amount of datasets multiplied by the number of ML algorithms determines the number of ML models constructed, as seen in Figure 2, the actions taken to coordinate online and batch ML procedures, respectively [17].

It's crucial to remember that any combination selected in the designer stage creates a unique container that operates on its own. An online ML pipeline where real-time datasets are created, preprocessed through multiple tasks (Figure 3), and then fed into trained models to generate online predictions, enabling continuous and automated data processing and inference.

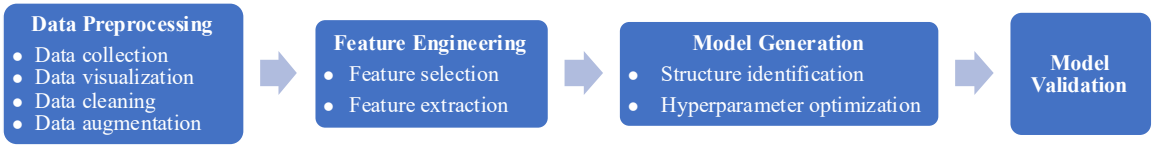

Figure 1. The typical pipeline of an Auto ML approach involves four basic stages.

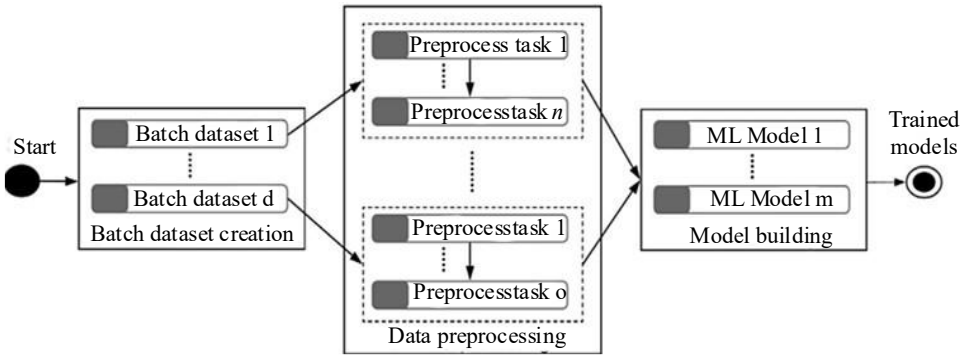

Figure 2. Orchestration steps of batch ML workflows

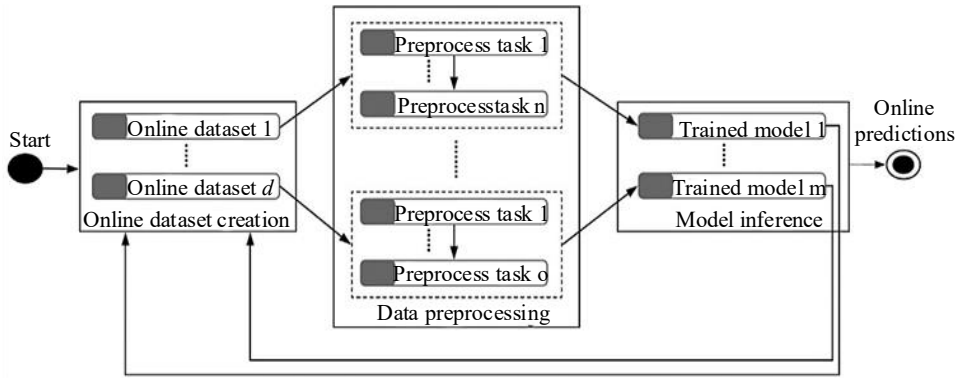

Figure 3. Orchestration steps of online ML workflows.

SCALABILITY OF MACHINE LEARNING PIPELINES

ML pipelines must be capable of scaling to accommodate the growth in the volume of data required, the computations involved and the processing in real-time. In the following section, it looks at methods of supporting scalable ML systems such as horizontal and vertical scaling, smart resource allocation and elastic infrastructure management. Autoscaling, container orchestration, and ML-assisted resource tuning are some of the methods that keep performance high, but the cost and efficiency are optimal. Moreover, there is the ability to learn continuously when using real-time and streaming pipelines to provide inference to time-based applications. Such scaling procedures are necessary to support ML processes in varied contexts, particularly cloud-native and edge-oriented deployments, both of which are very similar in their goal of ensuring resilience, flexibility, and the sustainability of pipelines.

Horizontal and Vertical Scaling

Scalability is concerned with establishing storage limits, distributing computations, and effectively training ML models with bigger or growing datasets. Especially for DL models that need a lot of training to reach acceptable performance levels, large datasets can greatly increase training durations [8]. This extended training puts a burden on computer power and can raise expenses, particularly when using cloud-based systems.

- *Horizontal scaling (scale-out):* on the other hand, means adding more instances to spread out processing; it allows scaling and fault tolerance and introduces a potential complexity of orchestration as well as latency during the startup of additional instances.
- *Vertical scaling (scale-up):* increases the capacity of a single node by adding CPU, memory, or GPU to handle a growing workload and offers quick responsiveness, but is constrained by hardware limits.

Advanced ML infrastructure is a hybrid autoscaling approach that begins with vertical scaling to capture demand spikes and gradually shifts into horizontal scaling to support the steady-state workload.

A new article presented Themis, a two-stage auto scaler that takes a thoughtful approach, leaning on each strategy to minimize SLO breaches and maximize resource utilization in inference-serving pipelines.

Resource Management and Optimization

Resource utilization is extremely important to the scaling and cost-effectiveness of ML pipelines. These include the dynamic assignment of computer resources, including CPU, GPU, memory and I/O, to workloads, which prevents either overprovisioning or overuse. As an example, in pipelines on Kubernetes, resources are requested and limited on a per-task basis to avoid overcommitment and a balance across cluster usage, whereas autoscores let the infrastructure scale gently up and down.

ML-driven structures in streamlining resources are becoming more common. An example of recent research in this direction is with DNN-driven auto-scalers, making it possible to tune the resource allocation of LLM pipelines. Here are some key benefits based on the scalability of ML pipelines are given below:

- *High utilization:* Autoscores that are powered by ML automatically scale node numbers, which avoids unutilized infrastructure.
- *Performance tuning:* Applications such as Plumber make data ingestion perform efficiently by parallelism and caching.
- *Cost efficiency:* The costs of using the cloud are decreased with the help of spot instances, preemptible VMs and efficient assignments based on optimizing the use of GPUs.
- *Adaptivity:* Systems are being learnt to understand patterns of usage so that they anticipate a demand and deploy proactively ahead of time.

These plans are a highlight of a new trend: smart, automated management of resources as a central feature to scale ML projects, particularly under a cloud and GPU-intensive infrastructure.

Real-Time and Streaming Pipelines

The way that data is processed and analyzed is being drastically changed by contemporary ML technologies. The exponential rise in data creation and the growing complexity of ML applications have propelled the shift towards real-time processing.

Stream pipeline for PdM to address real-time monitoring and PdM of equipment health, the pipeline consists of four principal modules and an optional filtering Figure 4. Each module is described in detail in the following subsections.

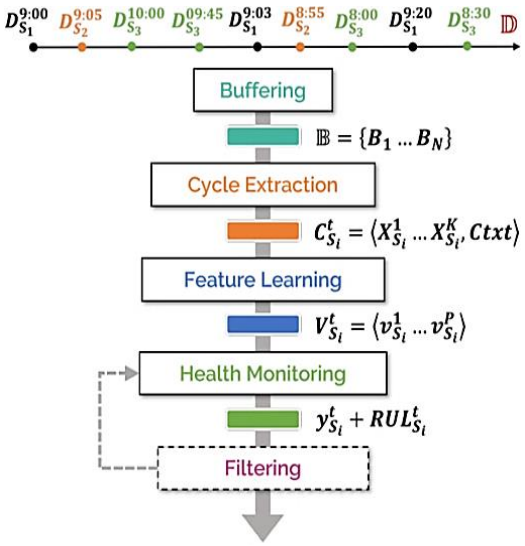

Figure 4. The five modules of the streaming pipeline.

Some ML applications on streaming pipelines are as follows:

- To make analysis easier, the buffering module arranges the supplied data.
- The cycle extraction (CE) module finds recurring data patterns that are shown by the system's cyclic behavior over time.
- The feature learning (FL) module produces relevant indicators.
- The health monitoring (HM) module carries out anomaly detection, anomaly prediction, and real-time monitoring.
- The filtering module runs the final verification before a maintenance order is sent to avoid false alerts; removing an alarm may induce reiteration of the HM module.

The pipeline receives as input a stream of data units D = {Dt, Si} t>0, where t marks the creation time and Si the ID of the system of provenance. No assumption is made about the format of Dt Si: it can be a data point, a file, or a batch of files.

Challenges in Scaling Model Training and Inference

Scalability remains a significant challenge in deploying ML models, particularly in operational settings where data volume, velocity, and diversity are constantly growing. As models grow in complexity and datasets expand, ensuring efficient and cost-effective training and inference becomes increasingly difficult. The key challenges in scaling model training and inference can be categorized as follows:

1. *Computational Resource Constraints:* Training large-scale models requires substantial compute power, often necessitating specialized hardware such as GPUs or TPUs. Memories and processing are increasingly becoming cost-prohibitive or otherwise inaccessible to large numbers of organizations because of the massive scale of the models (e.g., transformers with billions of parameters).
2. *Data Bottlenecks and I/O Limitations:* The training of models consumes a lot of data, data pipelines that are not optimized well, a lack of disk throughput, or networking may end up as a performance bottleneck, and are facilitated by unused compute resources. If large volumes of data are streamed or entire storage systems are accessed, the process usually has latency, and overall scalability is also impacted.
3. *Distributed Training Complexity:* Distributed training, though having the ability to scale the workload by applying to different nodes, brings issues like overhead of synchronization, latency of communication, and fault tolerance. Such methods as data parallelism or model parallelism must be implemented and coordinated with much care to prevent the dangers of inefficiency or convergence.
4. *Inference Latency and Throughput:* The models need to give rapid and correct predictions at inference time, with a limited latency threshold, particularly on real-time or edge applications. Serving inference for high throughput without explicitly creating degraded performance (or high cost) demands the use of optimized serving architecture, model compression schemes and acceleration of serving inference on hardware.
5. *Cost and Energy Efficiency:* The bigger the model, the more energy it consumes and higher the costs of operations. Training big models on the cloud may be a costly venture, especially in cases when a lot of hyperparameter optimization and re-training are performed. It also has a high energy consumption that poses some concerns in sustainability, hence the need of energy-efficient architectures and green AI practices.
6. *Model Versioning and Lifecycle Management:* Continuous training and deployment cyclesoffer advantages, but poses challangesin managing several versions of models in use. To enable reproducibility, backward compatibility and replicable performance across versions at scale, well-defined automation of pipeline and model registry systems are critical.
7. *Scalability Across Diverse Platforms:* The deployment of models on multiple platforms, i.e., cloud, edge, and mobile, requires scalable implementations based on the limitations of such environments. The portability and interoperability become a problem when trying to scale on a homogeneous basis among heterogeneous systems.

DEPLOYMENT STRATEGIES FOR ML PIPELINES

An important stage in ML pipelines is deployment strategies, which should see the models move easily between the development and production stages. Here, it will discuss major elements of deployment, such as the serving techniques of models, integration of CI/CD, and edge-hybrid structures. Low-latency, scalable inference is supported by model serving systems such as TensorFlow Serving and Serve, and continuous integration/continuous deployment pipelines automate testing and circle updates. Edge and hybrid deployments optimize resource usage, reduce latency, and enhance data privacy by distributing computation across cloud and local devices. Together, these deployment strategies enable reliable, scalable, and efficient ML systems tailored to diverse environments and real-time application needs.

Model Serving Techniques

Model serving is a key to success in the deployment of ML pipelines, which turns trained models into deployment-ready endpoints. There are two broad categories to model serving:

- *Model-serving runtimes:* Used to compile model packages into optimized containers or servers capable of handling inference requests.
- *Model-serving platforms:* That scale handle container deployment, auto-scaling and request routing e.g., KServe, Seldon Core, AWS SageMaker, and Databricks Mosaic AI Serving.

All these technologies compete at the level of development and production because it provides standard APIs, optimization capabilities, and are compatible with popular ML frameworks. The serving solution to be used should be able to balance important aspects such as latency, throughput, resource consumption, and integration requirements.

Using efficient model serving pipelines is essential in the transformation of static models into dynamic and production-ready services. In case it wants to consider adding performance benchmarking or containerization patterns, or integrating specimens with CI/CD pipelines.

Continuous Integration and Continuous Deployment (CI/CD)

In order to satisfy these needs, continuous integration and deployment, or CI/CD, have become essential procedures. The goal of continuous integration (CI) is to integrate code changes often, which enables the early identification and fixing of integration problems, some of which are listed below:

- *Continuous Integration (CI):* In order to rapidly get input on the feasibility of code, developers frequently integrate work into a shared repository throughout the day. This is a commonly used software development technique. Because CI facilitates automated builds and testing, teams may work together on a single project more rapidly. Software firms can also have a shorter and more frequent release cycle.
- *Continuous Deployment:* The technique of going live without human involvement as soon as qualifying modifications to software code or architecture are prepared is known as continuous deployment. Distinguishing between CD and CDT can be quite challenging.

An organization will no longer be able to complete a CI/CD pipeline on its own when it tries to implement one. They ought to embrace CD by practicing CI. The pipeline decreases the amount of manual execution during the transition from CI to CD, and eventually, the entire process is automated. Every step of the CD adoption process is carried out automatically. The term "CI/CD pipeline" describes the strategy, development, and deployment planning process.

Edge and Hybrid Deployments

Low latency, bandwidth, privacy, and system resources have become more and more viable trade-offs in the efficient deployment of ML models via edge architectures, and edge- and hybrid-architectures. Edge implementations remove inference to centralized cloud servers to mobile computer devices, and by doing so, can achieve real-time responsiveness and edge privacy. Hybrid deployments

combine cloud and edge computing, dynamically allocating workload, and operating at the best performance and cost, e.g., by performing basic inference locally and sending more arduous tasks to the cloud.

There are some key considerations and benefits below:

- *Latency and bandwidth efficiency:* Continuous On-Device investing prevents cloud round-trip latency constraints, which are crucial for time-sensitive applications like self-driving cars, augmented reality/virtual reality, and the IoT.
- *Resource constraints:* Runtime optimizations like TensorFlow Lite or ONNX Runtime, as well as model reduction methods like pruning, quantization, and distillation, provide the foundation of edge execution.
- *Scalability and system balance:* Partly to support hybrid arrangements, elements of the pipeline run in the cloud, but real-time processing is performed on edge-commodity devices (coordinated via orchestration systems).
- *Deployment frameworks:* Real-world architectures rely on Edge AI platforms, including TensorFlow Lite, Open VINO, Core ML, and hardware such as Google Coral and Jetson Nano.
- *Security & privacy:* Computation of sensitive data on the device minimizes the exposure, but distributed systems have a demand for secure communications, on-system software updates, and secured hardware.

LITERATURE REVIEW

This section reviews recent studies on automation, scalability, and deployment in machine learning pipelines, covering MLOps tools, observability frameworks, and pipeline architectures.

Berberi et al. (2025) [16] analyzed 16 popular MLOps products that focus on AI infrastructure management features, such as model deployment, experiment tracking, and model inference. Their evaluation process consists of three steps: first, a feature study of the MLOps platforms; second, a growth assessment of GitHub stars for acceptance and importance; and third, a weighted score system to identify the most influential platforms. This approach obtained important knowledge about the fundamental elements of successful MLOps systems from, and produced a decision-making flowchart that makes platform selection easier.

Singh (2025) [17] investigated how artificial intelligence enables observable functions that support the smooth integration of operations and automation between DevOps and MLOps development cycles. The research methodology includes a mixed approach that starts with a literature study, which combines practitioner interviews with DevOps and MLOps professionals, and a prototype AI observability framework development. The developed prototype utilizes ML analytics to detect anomalies, along with root cause identification and automated alert functions during evaluation on hybrid CI/CD and ML pipelines. AI-driven observability provides comprehensive application and model performance visibility.

Nielsen and Ayvaz (2025) [18] presents POE-ML, a Pipeline for Optimization and Evaluation of ML models that enables dynamic creation and testing of pipelines through experimental configurations. It allows exploring a wide range of models with different hyperparameters for model optimization and producing easily comparable results. To validate its effectiveness, the framework was evaluated using multiple datasets in regression and classification tasks. The findings showed that the suggested framework provides a workable way to optimize ML workflows in a variety of applications and is both reliable and flexible.

Lakatos et al. (2024) [19] demonstrate a model that uses ML techniques incorporated within a pipeline to collect insights from customer evaluations. The composite model employs transformer-based neural networks for clustering, vector-embedding-based keyword extraction, and NLP for topic

modeling. The model's components have been combined and modified to better satisfy the needs of topic modeling of the information that has been retrieved for opinion mining and effective information extraction. With the use of publicly accessible benchmark datasets, their methodology was verified and contrasted with other cutting-edge techniques. Results indicate that the system outperforms current keyword extraction and topic modeling techniques in this challenge.

Ogrizović et al. (2024) [20] included judgements on potential future development tendencies, a taxonomy, and a methodologically consistent presentation of all solutions offered to the aforementioned problems. Big data is nearly always the foundation of ML models, which have attracted a lot of interest in a range of applications, from computer vision to NLP. This paper's primary contributions include a classification that closely resembles the ML-pipeline's structure, a detailed description of each team member's role within that pipeline, and an overview of the trends and difficulties in combining ML and big data analytics, with applications in both industry and education.

Shojaee Rad and Ghobaei-Arani (2024) [21] offered a taxonomy of serverless computing data pipeline techniques. Workflow orchestration mechanisms, data processing methods, and architectural aspects provide the basis for classification. These strategies are divided into three main categories: framework-based, ML-based, and heuristic-based strategies. Additionally, a thorough analysis of current data pipeline frameworks and tools is given, including their advantages, disadvantages, and practical applications. Each strategy's benefits and drawbacks have been examined, as well as the challenges and performance metrics that influence its efficacy. Every data pipeline approach, whether it is framework, heuristic, or ML-based, has advantages and disadvantages. Every technique is effective for specific use cases.

Bogacka et al. (2024) [22] proposed system with a modular design and that can be used with a range of user-defined ML pipelines. To improve scalability and energy efficiency, a high-performance communication protocol for inference and a load balancer-based scale-out technique are proposed. The inference service plugs into the ASSIST-IoT reference architecture to access its other components. In two scenarios that closely reflected real-life use cases, the system was tested under demanding workloads and requirements that made up several deployment scenarios. The results of the evaluation show that the proposed software effectively adapts to the hardware available while meeting the requirements of the use cases for low inference latency and high throughput. The code, documentation, and evaluation results were all made publicly available to promote the solution's uptake.

Table 1 highlights focal areas, methodological approaches, major findings, recognized limitations, and suggested future research paths for important studies on automation and deployment techniques in ML processes.

CONCLUSION AND FUTURE WORK

It is important to have community outreach programs, deployment strategies and efforts to provide a smooth transition between the model development and production. With increasing ML system scale, optimized pipelines are increasingly needed that can enable automation, low-latency inference, and execute this across a variety of environments. These needs can be addressed through strong serving mechanisms, incorporation of CI/CD workflow, and deployment architectures that are low-demanding on resources. The performance, scalability, and dependability of ML pipelines may all be impacted by deployment tactics. With the help of model serving technologies, real-time, production-ready inference is made possible. CI/CD integration allows for quick and frequent updates with minimal human involvement. Edge and hybrid deployments take this even further by making ML applications available at low latency and preserving privacy in dynamically evolving computing contexts. The combination of these elements constitutes an integrated infrastructure for end-to-end automation and sustainable growth. The next research areas should include providing a standardized deployment strategy, better orchestration architectures, and compatibility within evolving hardware and software environments.

Table 1. Comparative analysis of recent studies on automation and deployment strategies in machine learning workflows.

Reference	Study On	Approach	Key Findings	Challenges	Future Direction
Berberi et al. (2025) [16]	Evaluation of 16 MLOps platforms for automation, deployment, and model management	3-step framework: feature analysis, GitHub trend tracking, weighted scoring	Identified essential MLOps components and created a platform selection flowchart	Lack of standardization across platforms	Encourages adaptive tool selection and benchmarking in the evolving MLOps landscape
Singh et.al. (2025) [17]	AI-driven observability in MLOps and CI/CD pipelines	Mixed-method: literature study, practitioner interviews, prototype development	Developed an observability framework for anomaly detection and root cause analysis in hybrid pipelines	Integration complexity with existing CI/CD systems	Scaling AI observability to more diverse operational contexts
Nielsen et.al. (2025) [18]	Optimization and evaluation of ML pipelines	Developed POE-ML, a pipeline framework for dynamic creation, hyperparameter tuning, and testing of ML models in regression/classification tasks	The framework is robust, adaptable, and enables comparative analysis of models with diverse configurations	Scalability across very large datasets and integration with emerging ML ops tools could be limiting	Extend POE-ML for real-time ML pipelines and automated deployment across platforms
Lakatos et al. (2024) [19]	Insight extraction from customer reviews through ML pipelines	An integrated pipeline for opinion mining that combines transformer-based natural language processing, vector embedding, keyword extraction, and clustering	Performs better on benchmark datasets than current topic modeling and keyword extraction models.	Complexity of integrating transformer models with traditional clustering and ensuring interpretability	Improve model generalizability across languages/domains and explore real-time topic tracking applications
Ogrizović et.al. (2024) [20]	Taxonomy and role distribution in ML pipelines using big data	Structural taxonomy and role-based analysis	Defined pipeline stages and team responsibilities, highlighted industrial trends	Complexity in coordinating ML teams in big data environments	Calls for educational and tooling support to simplify big data ML pipelines
Shojaee Rad et al. (2024) [21]	Data pipeline strategies in serverless computing	Taxonomy of heuristic-, ML-, and framework-based approaches	Detailed pros/cons of various serverless pipeline architectures and tools	Performance tuning and latency control in serverless workflows	Future hybrid models combining multiple approaches for flexibility
Bogacka et al. (2024) [22]	Modular deployment framework for ML inference in IoT	Software implementation with scale-out mechanisms and load balancing	Demonstrated high-throughput and low-latency deployment across scenarios	Balancing energy efficiency with performance in edge environments	Promote open-source adoption and integration into industrial IoT ecosystems

Future efforts can later prioritize the creation of standardized and framework-agnostic deployment procedures in order to promote interoperability between different ML sectors of operation. Its deployment will need to be scaled up through improvements to the deployment of lightweight models in optimization, safer edge computing, or smarter orchestration systems. Auto-updated ML systems can

also enhance automation, resilience, and performance in changing ML environments by integrating adaptive CI/CD pipelines that automatically adjust according to data and concept drift in real time.

REFERENCES

1. Liang P, Song B, Zhan X, Chen Z, Yuan J. Automating the training and deployment of models in MLOps by integrating systems with machine learning. arXiv preprint arXiv:2405.09819. 2024 May 16.
2. Pahune S, Akhtar Z. Transitioning from MLOps to LLMOps: Navigating the unique challenges of large language models. Information. 2025;16(2):87.
3. Neeli SS. Optimizing Database Management with DevOps: Strategies and Real-World Examples. J. Adv. Dev. Res. 2020;11(1):8.
4. Singla A. Machine learning operations (MLOps): challenges and strategies. J Knowl Learn Sci Technol. 2023 Aug 14;2(3):333–40.
5. Peter H. Cloud Native MLOps Automating Machine Learning Pipelines with Continuous Integration Continuous Deployment and Infrastructure as Code. 2024; https://www.researchgate.net/profile/Harry-Peter/publication/392101895_Cloud_Native_MLOps_Automating_Machine_Learning_Pipelines_with_Continuous_Integration_Continuous_Deployment_and_Infrastructure_as_Code/links/683500ed026fee1034fc2233/Cloud-Native-MLOps-Automating-Machine-Learning-Pipelines-with-Continuous-Integration-Continuous-Deployment-and-Infrastructure-as-Code.pdf
6. Garg S. AI/ML Driven Proactive Performance Monitoring, Resource Allocation and Effective Cost Management in SAAS Operations. Int J Core Eng Manag. 2019;6(6):263–73.
7. Shivashankar K, Hajj GS, Martini A. Scalability and Maintainability Challenges and Solutions in Machine Learning: Systematic Literature Review. arXiv preprint arXiv:2504.11079. 2025 Apr 15.
8. Arora S, Thota SR, Gupta S. Artificial Intelligence-Driven Big Data Analytics for Business Intelligence in SaaS Products. In2024 First International Conference on Pioneering Developments in Computer Science & Digital Technologies (IC2SDT) 2024 Aug 2 (pp. 164–169). IEEE.
9. Chitraju Gopal Varma S. Optimizing Machine Learning Pipelines: Best Practices for Scalable and Efficient Model Deployment. Available at SSRN 5226791. 2024 Nov 20.
10. Malali N. Cloud-Native Security and Compliance in Life and Annuities Insurance: Challenges and Best Practices. International Journal of Interdisciplinary Research Methods (IJIRM). 2025;12(1):50–73.
11. Pal G, Li G, Atkinson K. Big data real time ingestion and machine learning. In2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP) 2018 Aug 21 (pp. 25–31). IEEE.
12. Thota SR, Arora S, Gupta S. Al-Driven Automated Software Documentation Generation for Enhanced Development Productivity. In2024 International Conference on Data Science and Network Security (ICDSNS) 2024 Jul 26 (pp. 1–7). IEEE.
13. a Ilemobayo J, Durodola O, Alade O, Awotunde OJ, Olanrewaju AT, Falana O, Ogungbire A, Osinuga A, Ogunbiyi D, Ifeanyi A, Odezuligbo IE. Hyperparameter tuning in machine learning: a comprehensive review. J Eng Res Reports. 2024 Jun 7;26(6):388–95.
14. Filippou K, Aifantis G, Papakostas GA, Tsekouras GE. Structure learning and hyperparameter optimization using an automated machine learning (AutoML) pipeline. Information. 2023 Apr 9;14(4):232.
15. Pradhan AK, Kumar D, Mishra MK, Singh MK. Edge, Fog, and Cloud Computing in Industry 5.0. InIndustry 5.0: Key Technologies and Drivers 2025 Jul 18 (pp. 1–27). Cham: Springer Nature Switzerland.
16. Berberi L, Kozlov V, Nguyen G, Sáinz-Pardo Díaz J, Calatrava A, Moltó G, Tran V, López García Á. Machine learning operations landscape: platforms and tools. Artif Intell Rev. 2025 Mar 17;58(6):167.
17. Singh S. Unifying DevOps and MLOps Pipelines Via AI-driven Observability: A Mixed-Methods Study. Asian J Res Comput Sci. 2025 Jun 10;18(6):334–42.
18. Nielsen MC, Ayvaz S. POE-ML: An Automated Pipeline for Optimization and Evaluation of Machine Learning. In2025 IEEE 22nd International Conference on Software Architecture Companion (ICSA-C) 2025 Mar 31 (pp. 508–515). IEEE.

19. Lakatos R, Bogacsovics G, Harangi B, Lakatos I, Tiba A, Tóth J, Szabó M, Hajdu A. A machine learning-based pipeline for the extraction of insights from customer reviews. Big Data Cogn Comput. 2024 Feb 22;8(3):20.
20. Ogrizović M, Drašković D, Bojić D. Quality assurance strategies for machine learning applications in big data analytics: an overview. J Big Data. 2024 Oct 30;11(1):156.
21. Shojaee Rad Z, Ghobaei-Arani M. Data pipeline approaches in serverless computing: a taxonomy, review, and research trends. J Big Data. 2024 Jun 11;11(1):82.
22. Bogacka K, Sowiński P, Danilenka A, Biot FM, Wasielewska-Michniewska K, Ganzha M, Paprzycki M, Palau CE. Flexible deployment of machine learning inference pipelines in the cloud–edge–IoT continuum. Electronics. 2024 May 11;13(10):1888.

