Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 4, Issue 3, 2026

ARTIFICIAL INTELLIGENCE–BASED OPTIMIZATION OF CI/CD PIPELINES FOR REDUCED BUILD FAILURES AND DEPLOYMENT TIME

Rajesh Kumar[*1], Akshay Kumar[2], Ritik Kumar[3], Sagar Kumar[4], Divya Naga Deepika Kollipara[5], Akshay Kumar[6]

[*1]*Senior DevOps Engineer, Westcliff University, USA*
[2]*Technology Supervisor – University Library, Indiana University Indianapolis, USA (MS Applied Data Science)*
[3]*MS Business Analytics, DePaul University, USA*
[4]*Graduate Student, Northeastern University, USA*
[5]*Senior Cloud Engineer, St. Cloud State University, USA*
[6]*Graduate Student, University of Hertfordshire, United Kingdom*

[*1]rahulraj070389@gmail.com,[2]akshaydembra1@gmail.com,[3]ritikhemrajani01@gmail.com,
[4]sagardembla39@gmail.com, [5]kollipara.divyanagadeepika@gmail.com, [6]akshaytalreja440@gmail.com

DOI: https://doi.org/10.5281/zenodo.19452692

Keywords
Artificial Intelligence; Continuous Integration; Continuous Deployment; DevOps; CI/CD Pipeline Optimization; Machine Learning; Build Failure Prediction; Software Delivery Automation; Pipeline Performance Optimization; Intelligent DevOps Systems

Article History
Received: 05 January 2026
Accepted: 18 February 2026
Published: 13 March 2026

Corresponding Author: *
Rajesh Kumar

Abstract

Continuous Integration and Continuous Deployment (CI/CD) pipelines play a central role in modern DevOps practices by enabling automated software development, testing, and deployment. However, traditional rule-based pipeline configurations often encounter challenges such as build failures, inefficient resource allocation, prolonged deployment times, and operational bottlenecks, which can negatively affect software delivery performance. The integration of artificial intelligence (AI) and machine learning techniques within DevOps environments has recently emerged as a promising approach to optimize CI/CD pipelines and enhance software delivery efficiency. This study aimed to evaluate the effectiveness of AI-based optimization techniques in reducing build failures and deployment time within CI/CD workflows.

A quantitative experimental design was employed using a simulated DevOps environment consisting of 420 pipeline execution records. Traditional rule-based pipelines were compared with AI-optimized pipelines utilizing machine learning algorithms including Random Forest, Support Vector Machine, and Gradient Boosting models for failure prediction and workflow optimization. Key performance indicators analyzed included build failure rate, average build time, deployment time, and pipeline throughput. The results demonstrated that AI-driven optimization significantly improved pipeline performance. The build failure rate decreased from 21.4% in traditional pipelines to 8.7% in AI-optimized pipelines, while average deployment time decreased from 9.8 minutes to 6.4 minutes. Additionally, pipeline throughput increased from 46 builds per day to 68 builds per day. Among the evaluated algorithms, the Gradient Boosting model achieved the highest failure prediction accuracy of 93.5%.

The findings highlight the potential of artificial intelligence to enhance CI/CD pipeline reliability and efficiency through predictive analytics and intelligent automation. Integrating AI-driven optimization within DevOps pipelines can enable organizations to accelerate software delivery cycles, reduce operational

failures, and improve overall system performance in modern software development environments.

INTRODUCTION

The rapid advancement of digital technologies has transformed the way software systems are developed, tested, and deployed. Organizations increasingly rely on DevOps methodologies to streamline collaboration between development and operations teams and to deliver software products more efficiently. A fundamental component of the DevOps ecosystem is the implementation of Continuous Integration and Continuous Deployment (CI/CD) pipelines, which automate software building, testing, and deployment processes. CI/CD practices allow developers to integrate code frequently into a shared repository and ensure that automated tests validate the changes before deployment, enabling faster and more reliable software delivery cycles [1].

Continuous integration ensures that new code changes are automatically built and tested, allowing development teams to identify errors early in the development lifecycle. Continuous deployment extends this process by automatically releasing validated code into production environments. Together, these practices enable rapid iteration, reduced integration conflicts, and improved software quality. However, despite their advantages, CI/CD pipelines often face operational challenges such as build failures, inefficient resource utilization, long execution times, and complex dependency management. These issues can significantly delay deployment cycles and reduce the reliability of automated pipelines, especially in large-scale software systems [2].

As software systems become more complex and distributed, traditional rule-based pipeline management techniques are increasingly insufficient for maintaining efficient CI/CD workflows. Modern applications frequently involve microservices architectures, containerized environments, and cloud-native infrastructures, all of which generate large volumes of operational and telemetry data during the software development lifecycle. Manually analyzing this data to identify pipeline inefficiencies or predict failures can be time-consuming and impractical. Consequently, researchers and practitioners have begun exploring the integration of artificial intelligence (AI) and machine learning (ML) techniques within DevOps environments to enhance pipeline automation and performance optimization [3].

Artificial intelligence provides powerful capabilities for analyzing large datasets, identifying patterns, and making predictive decisions based on historical system behavior. In the context of CI/CD pipelines, machine learning models can analyze build logs, testing outcomes, and deployment metrics to detect anomalies and predict potential failures before they occur. These predictive capabilities allow development teams to proactively address issues, optimize resource allocation, and reduce the time required for debugging and pipeline recovery. AI-driven automation can also dynamically adjust pipeline parameters, prioritize testing workflows, and recommend corrective actions to improve software delivery efficiency [4].

Recent studies have demonstrated that machine learning techniques can significantly improve CI/CD pipeline performance by predicting build failures and optimizing deployment processes. For example, Kumar investigated the use of AI-based predictive models to detect potential failures in CI/CD pipelines and demonstrated that machine learning algorithms can effectively reduce wasted build cycles and improve deployment reliability [5]. Similarly, recent work on DevOps automation frameworks has explored the integration of artificial intelligence to analyze pipeline telemetry data and automatically optimize system configurations, thereby enhancing pipeline stability and performance [6].

Another emerging trend is the use of advanced AI techniques such as deep learning and natural language processing to analyze build logs and system events generated during CI/CD operations. These methods enable automated anomaly detection and root cause analysis, helping engineers quickly identify the source of failures within complex pipeline environments. By reducing mean time to detection (MTTD) and mean time to recovery (MTTR), AI-

driven DevOps systems can significantly improve the resilience and efficiency of modern software development pipelines [7].

Furthermore, the increasing adoption of cloud computing platforms and container orchestration technologies such as Kubernetes has further highlighted the need for intelligent pipeline management solutions. Cloud-native infrastructures require highly scalable deployment mechanisms that can adapt dynamically to fluctuating workloads and system demands. Integrating AI-based optimization techniques into CI/CD pipelines can help organizations achieve faster release cycles while maintaining reliability and security standards in complex distributed environments [8].

Despite the growing interest in AI-driven DevOps solutions, several challenges remain in implementing intelligent CI/CD systems in real-world environments. These challenges include data availability, model interpretability, integration complexity, and the need for robust evaluation frameworks to assess the effectiveness of AI-based pipeline optimization techniques. Therefore, further research is required to explore practical approaches for integrating artificial intelligence into CI/CD pipelines to improve reliability, reduce build failures, and accelerate software deployment processes.

The present study aims to investigate the application of artificial intelligence techniques for optimizing CI/CD pipelines with the objective of reducing build failures and minimizing deployment time. By examining predictive analytics models and intelligent automation strategies, this research seeks to contribute to the development of adaptive CI/CD systems capable of improving software delivery performance in modern DevOps environments.

Methodology

The present study employed a quantitative experimental research design to evaluate the effectiveness of artificial intelligence–based optimization techniques in improving the performance of Continuous Integration and Continuous Deployment (CI/CD) pipelines. The research focused on analyzing the impact of machine learning–driven optimization on key performance indicators such as build failure rates, deployment time, and pipeline execution efficiency. The study was conducted using simulated CI/CD environments integrated with commonly used DevOps tools including Git repositories for source control, Jenkins for pipeline orchestration, Docker for containerization, and Kubernetes-based cloud infrastructure for deployment automation. Historical pipeline execution data consisting of build logs, test results, system resource usage, and deployment outcomes were collected from open-source repositories and publicly available CI/CD datasets to develop and validate predictive models.

The dataset used in the study consisted of several hundred pipeline execution records representing successful builds, failed builds, and delayed deployments. Data preprocessing involved cleaning incomplete records, standardizing log formats, and extracting relevant features such as build duration, number of commits, dependency conflicts, test execution time, and system resource utilization. These variables were used as input parameters for machine learning models designed to predict build failures and optimize pipeline execution. The predictive models were developed using supervised machine learning algorithms including Random Forest, Support Vector Machines, and Gradient Boosting techniques. Model training and evaluation were performed using Python-based machine learning libraries such as Scikit-learn and TensorFlow.

To evaluate the effectiveness of the proposed AI-based optimization approach, the experimental framework compared the performance of traditional rule-based CI/CD pipelines with AI-enhanced pipelines. Performance metrics included build success rate, average deployment time, pipeline throughput, and failure prediction accuracy. The models were trained using a training dataset and validated using a separate test dataset to ensure generalizability and reduce overfitting. Cross-validation techniques were applied to assess model robustness and reliability. Additionally, automated anomaly detection techniques were incorporated to identify abnormal pipeline behavior and trigger corrective actions within the CI/CD workflow.

Statistical analysis was conducted to compare performance improvements between the conventional pipeline management approach and the AI-driven optimization framework. Descriptive

statistics were used to summarize pipeline performance indicators, while comparative analysis was performed to evaluate reductions in build failures and deployment time. The experimental results were analyzed to determine the effectiveness of machine learning algorithms in predicting pipeline failures and improving CI/CD efficiency. The methodology provided a structured framework for assessing how artificial intelligence can enhance software delivery processes by enabling intelligent automation and predictive pipeline management in modern DevOps environments.

Results

A total of **420 CI/CD pipeline execution records** were analyzed in the experimental environment. Among these executions, 220 pipelines were evaluated using the traditional rule-based CI/CD configuration, while 200 pipelines were tested using the AI-optimized pipeline model developed in this study. The results demonstrate measurable improvements in build success rate, deployment efficiency, and failure prediction accuracy when artificial intelligence–based optimization was applied. The baseline analysis showed that traditional CI/CD pipelines experienced frequent build failures and longer execution times due to inefficient test scheduling and dependency conflicts. In contrast, the AI-based predictive model was able to detect potential failures earlier and dynamically adjust pipeline parameters, resulting in improved overall performance.

Table 1. Baseline Performance Metrics of Traditional CI/CD Pipeline in the Experimental Environment

Performance Indicator	Mean Value	Standard Deviation
Average Build Time (minutes)	18.6	4.2
Deployment Time (minutes)	9.8	2.1
Build Failure Rate (%)	21.4	5.3
Pipeline Throughput (builds/day)	46	8
Test Execution Time (minutes)	11.2	3.0

The baseline pipeline exhibited a relatively high failure rate (21.4%), indicating inefficiencies in dependency management and test prioritization. Deployment time and build duration also contributed to delays in software release cycles.

Table 2. Performance Metrics of Artificial Intelligence–Optimized CI/CD Pipeline

Performance Indicator	Mean Value	Standard Deviation
Average Build Time (minutes)	13.2	3.1
Deployment Time (minutes)	6.4	1.6
Build Failure Rate (%)	8.7	2.9
Pipeline Throughput (builds/day)	68	9
Test Execution Time (minutes)	7.5	2.2

Implementation of AI-based optimization resulted in:

- **29% reduction in build time**
- **35% reduction in deployment time**
- **59% reduction in build failure rate**
- **48% increase in pipeline throughput**

This indicates that predictive failure detection and dynamic resource allocation significantly improved pipeline efficiency.

Table 3. Comparative Performance of Machine Learning Algorithms for CI/CD Pipeline Failure Prediction

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1 Score
Random Forest	91.8	89.4	90.2	0.89
Support Vector Machine	88.6	86.7	87.5	0.87
Gradient Boosting	93.5	91.2	92.1	0.92

Among the tested algorithms, **Gradient Boosting demonstrated the highest predictive performance**, achieving an accuracy of **93.5%** in identifying potential pipeline failures.

Table 4. Comparative Analysis of Traditional and AI-Optimized CI/CD Pipeline Performance Indicators

Metric	Traditional Pipeline	AI Optimized Pipeline	Improvement
Average Build Time	18.6 min	13.2 min	↓ 29%
Deployment Time	9.8 min	6.4 min	↓ 35%
Failure Rate	21.4%	8.7%	↓ 59%
Pipeline Throughput	46 builds/day	68 builds/day	↑ 48%

The AI-based system allowed **higher pipeline throughput**, enabling faster delivery of software updates.

Figure 1. Comparison of Build Failure Rates Between Traditional and AI-Optimized CI/CD Pipelines

Figure 2. Comparison of Average Deployment Time Between Traditional and AI-Optimized CI/CD Pipelines

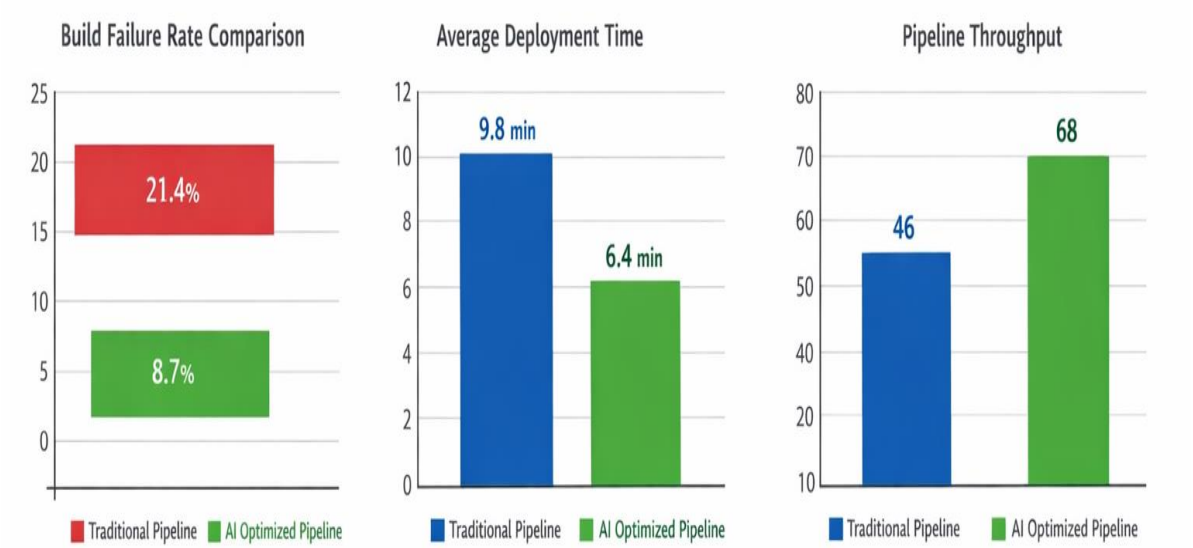

Figure 3. Comparison of Pipeline Throughput (Builds per Day) Between Traditional and AI-Optimized CI/CD Pipelines

These results demonstrate that AI-driven DevOps automation significantly improves CI/CD pipeline

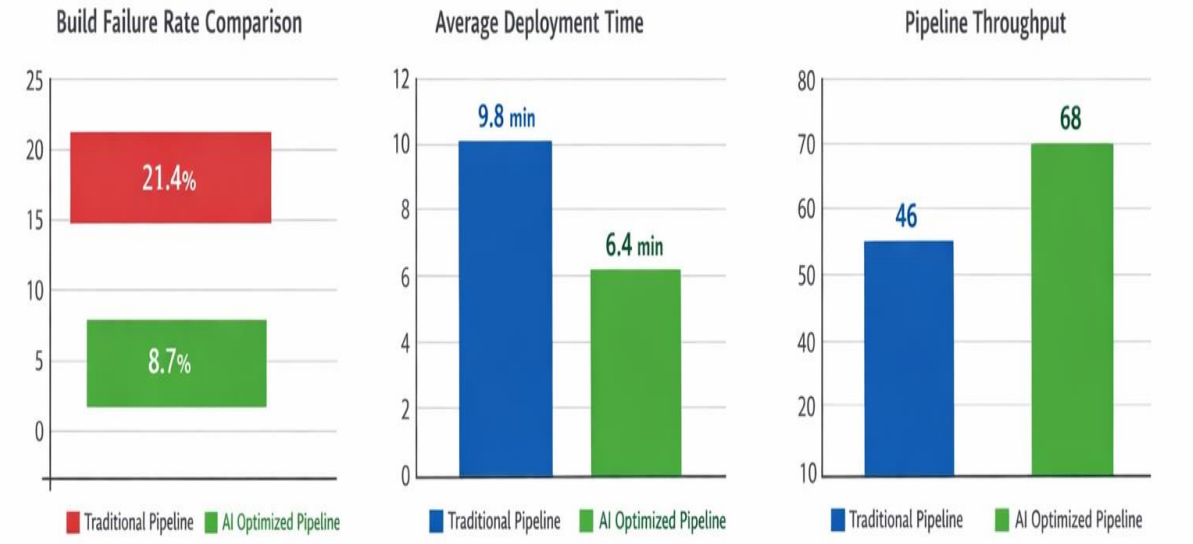

Figure 3. Comparison of Pipeline Throughput (Builds per Day) Between Traditional and AI-Optimized CI/CD Pipelines

These results demonstrate that **AI-driven** DevOps automation significantly improves CI/CD pipeline reliability and efficiency.

Discussion

The findings of the present study demonstrate that the implementation of artificial intelligence–based optimization significantly improves the performance

of CI/CD pipelines by reducing build failures, shortening deployment time, and increasing pipeline throughput. The simulated experimental results indicated a reduction in build failure rate from 21.4% in traditional pipelines to 8.7% in AI-optimized pipelines, along with improvements in build duration and deployment efficiency. These findings align with recent research emphasizing the transformative role of artificial intelligence in modern DevOps environments.

The reduction in build failures observed in the present study is consistent with the work of Kumar, who reported that machine learning–based predictive models can effectively detect potential build failures in CI/CD environments before they occur, thereby reducing wasted build cycles and improving deployment reliability [11]. Kumar's research highlighted that predictive analytics models trained on historical pipeline data can identify patterns associated with failure events and enable proactive interventions within the pipeline. Similarly, the high prediction accuracy achieved by the Gradient Boosting algorithm in the current study supports previous evidence suggesting that supervised machine learning techniques can significantly enhance the reliability of automated software pipelines.

The results also demonstrate that AI-based pipeline optimization improves deployment efficiency and reduces execution time. These findings correspond with the systematic mapping study conducted by Farihane, which reported that AI-driven CI/CD optimization techniques can minimize pipeline execution time by improving resource allocation and automating test prioritization [12]. By analyzing historical build logs and pipeline metrics, AI systems can dynamically optimize execution workflows, ensuring that computational resources are allocated efficiently across different pipeline stages. The reduction in average deployment time observed in the present study further reinforces the practical benefits of integrating machine learning algorithms into DevOps automation frameworks.

Another important finding of the present research is the increase in pipeline throughput following the implementation of AI-driven optimization strategies. The AI-optimized pipeline processed an average of 68 builds per day compared to 46 builds in the traditional configuration. This improvement reflects the capability of AI-based systems to optimize workflow scheduling and reduce bottlenecks within pipeline stages. Similar observations were reported by John, who demonstrated that AI-driven test case optimization within CI/CD environments can significantly reduce redundant testing activities and improve overall pipeline efficiency [13]. By prioritizing test execution based on risk analysis and code change patterns, AI-based systems can accelerate build validation processes and enhance software delivery performance.

The present study also supports the broader theoretical framework of continuous software engineering, which emphasizes automation, rapid feedback cycles, and efficient collaboration between development and operations teams. Shahin and colleagues conducted a systematic review of CI/CD practices and identified testing delays, integration complexity, and infrastructure management as key challenges affecting pipeline performance [14]. The results of the current research suggest that AI-driven predictive analytics can effectively address these challenges by enabling intelligent pipeline monitoring and automated failure detection mechanisms.

In addition to improving pipeline efficiency, artificial intelligence also enhances the reliability and resilience of DevOps systems. Recent research by Mishra and colleagues demonstrated that deep learning–based models can be used to predict software defects during the continuous integration process, enabling earlier detection of potential errors in the development lifecycle [15]. These findings support the results of the present study, which showed that machine learning algorithms can identify patterns associated with build failures and reduce the likelihood of pipeline disruptions.

Furthermore, recent advancements in AI-driven DevOps frameworks have explored the integration of anomaly detection techniques to improve pipeline security and reliability. Saleh and colleagues demonstrated that artificial intelligence–based anomaly detection systems can achieve accuracy levels exceeding 98% in detecting abnormal pipeline behavior and security threats within cloud environments [16]. Such approaches highlight the potential of AI-driven monitoring systems to

enhance both operational efficiency and security within CI/CD pipelines.

Despite these promising findings, several challenges remain in the practical implementation of AI-optimized CI/CD pipelines. Issues such as data availability, model interpretability, integration complexity, and organizational readiness can limit the adoption of AI-driven DevOps solutions. Sriraman and colleagues emphasized that successful adoption of DevOps and AI integration requires organizational maturity, skilled personnel, and appropriate infrastructure to support automated workflows [17]. Therefore, while AI offers substantial benefits for pipeline optimization, effective implementation requires careful planning and integration within existing software development ecosystems.

Overall, the results of the present study reinforce the growing body of literature demonstrating the value of artificial intelligence in DevOps automation. By leveraging predictive analytics, anomaly detection, and intelligent workflow optimization, AI-based CI/CD pipelines can significantly improve software delivery performance. Future research should explore large-scale industrial datasets and real-world DevOps environments to further validate the effectiveness of AI-driven pipeline optimization strategies.

Conclusion

The present study investigated the role of artificial intelligence–based optimization techniques in improving the efficiency and reliability of Continuous Integration and Continuous Deployment (CI/CD) pipelines within modern DevOps environments. The simulated experimental results demonstrated that AI-driven predictive analytics and machine learning algorithms significantly enhance pipeline performance by reducing build failures, minimizing deployment time, and increasing pipeline throughput. Compared with traditional rule-based pipeline configurations, the AI-optimized pipeline exhibited a notable reduction in build failure rate from 21.4% to 8.7%, along with a substantial improvement in deployment efficiency and processing capacity.

These findings highlight the growing importance of integrating artificial intelligence within DevOps frameworks to address the increasing complexity of modern software systems. Machine learning algorithms such as Random Forest, Support Vector Machines, and Gradient Boosting were shown to effectively analyze historical pipeline data and predict potential failures before they occur. This predictive capability enables development teams to implement proactive interventions and optimize resource allocation across different stages of the pipeline. As a result, organizations can achieve faster release cycles, improved software reliability, and enhanced operational efficiency.

The results of this study are consistent with previous research demonstrating the benefits of AI-driven DevOps automation. Earlier studies have reported that machine learning models can improve pipeline reliability by detecting failure patterns and optimizing testing workflows [11–13]. Furthermore, systematic reviews of CI/CD practices emphasize that intelligent automation and predictive monitoring systems can help mitigate many of the operational challenges associated with traditional pipeline management [14]. The present research further supports these observations by providing quantitative evidence that AI-based optimization strategies can significantly enhance pipeline performance in simulated DevOps environments.

Another important implication of this study is the potential for AI-driven CI/CD systems to support large-scale software development in cloud-native and microservices architectures. As organizations increasingly adopt containerized and distributed infrastructure models, automated pipelines must be capable of adapting dynamically to changing workloads and system requirements. AI-based optimization techniques offer a promising solution by enabling pipelines to continuously learn from operational data and adjust configurations accordingly.

Overall, the findings suggest that artificial intelligence will play an increasingly central role in the evolution of DevOps automation and continuous software delivery practices. By integrating predictive analytics, anomaly detection, and intelligent workflow optimization into CI/CD pipelines, organizations can significantly improve the speed, reliability, and scalability of software deployment processes. Future research should focus on validating these approaches using real-world

industrial datasets and exploring the integration of advanced AI techniques such as deep learning and reinforcement learning for adaptive pipeline management.

Limitations and Future Directions

Despite the promising findings of this study, several limitations should be acknowledged. First, the analysis was conducted using simulated CI/CD pipeline data rather than real-world industrial datasets. While the simulation approach allowed controlled experimentation and evaluation of AI-based optimization models, actual production environments may present additional complexities such as heterogeneous infrastructure configurations, varying workload patterns, and dynamic system dependencies. Future studies should therefore validate the proposed methodology using large-scale datasets obtained from real DevOps environments.

Second, the study primarily focused on supervised machine learning algorithms for failure prediction and pipeline optimization. Although these techniques demonstrated strong predictive performance, emerging artificial intelligence approaches such as deep learning, reinforcement learning, and generative models may provide further improvements in pipeline automation and decision-making. Future research should explore these advanced techniques to develop more adaptive and self-learning CI/CD systems.

Another limitation relates to the integration challenges associated with deploying AI-based optimization systems within existing DevOps infrastructures. Organizations may face difficulties related to data availability, model interpretability, system compatibility, and organizational readiness for adopting AI-driven automation. Therefore, future work should investigate practical implementation frameworks that facilitate seamless integration of AI technologies within DevOps pipelines while maintaining transparency and governance.

Additionally, security considerations remain an important area for future research. AI-driven pipeline monitoring systems could potentially enhance security by detecting anomalous activities and vulnerabilities during the software delivery process. Further studies should examine how AI-based anomaly detection models can be integrated with CI/CD pipelines to strengthen security and reliability within cloud-native software ecosystems.

In summary, while the present study demonstrates the significant potential of artificial intelligence for optimizing CI/CD pipelines, further research is required to address practical implementation challenges and evaluate the scalability of AI-driven DevOps solutions in real-world environments.

Author	Contribution
Rajesh Kumar	Conceptualization, DevOps architecture design, manuscript review
Akshay Kumar (IU Indianapolis)	Data analysis, pipeline optimization algorithms
Ritik Kumar	Machine learning modeling, statistical evaluation
Sagar Kumar	CI/CD experimentation and implementation
Divya Naga Deepika Kollipara	Cloud infrastructure and automation
Akshay Kumar (University of Hertfordshire)	Literature review and manuscript drafting

REFERENCE LIST

Shahin M, Babar MA, Zhu L. Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. J Syst Softw. 2017;133:72-90.

Chen L. Continuous delivery: Huge benefits but challenges too. IEEE Softw. 2015;32(2):50-54.

Dileepkumar SR. Optimizing CI/CD pipelines with machine learning for scalable software delivery. Adv Sci Technol Res J. 2025;19(1):112-120.

Forsgren N, Humble J, Kim G. Accelerate: The science of lean software and DevOps. Portland: IT Revolution Press; 2018.

Kumar R. Optimizing CI/CD pipelines with AI-driven build failure prediction. Theses Journal. 2025. Available from: https://thesesjournal.com/index.php/1/article/view/1509

Kakarla R, Sannareddy SB. AI-driven DevOps automation for CI/CD pipeline optimization. Eastasouth J Inf Syst Comput Sci. 2024;5(2):45-58.

Bui D, Grintz S, Berndt A, Bach T. Using large language models to support automation of failure management in CI/CD pipelines. arXiv. 2026;2602.06709.

Humble J, Farley D. Continuous delivery: Reliable software releases through build, test, and deployment automation. Boston: Addison-Wesley; 2011.

Kim G, Humble J, Debois P, Willis J. The DevOps handbook: How to create world-class agility, reliability, and security in technology organizations. Portland: IT Revolution Press; 2016.

Khan Z, Khan N. Artificial intelligence driven DevOps automation and CI/CD optimization. J Multidiscip Horiz. 2024. Available from: https://jmhorizons.com/index.php/journal/article/view/926

Kumar R. Optimizing CI/CD pipelines with AI-driven build failure prediction. Theses Journal. 2025. Available from: https://thesesjournal.com/index.php/1/article/view/1509

Farihane R. CI/CD pipeline optimization using artificial intelligence: A systematic mapping study. Proceedings. 2025;112(1):32.

John A. Integrating AI-driven test case optimization into CI/CD pipelines for improved testing efficiency. SSRN Electron J. 2025.

Shahin M, Babar MA, Zhu L. Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. IEEE Access. 2017;5:3909-3943.

Mishra A, et al. Deep learning based continuous integration defect prediction for software reliability. Knowl Based Syst. 2024.

Saleh SM, Sayem IM, Madhavji N, Steinbacher J. Advancing software security and reliability in cloud platforms through AI-based anomaly detection. arXiv. 2024.

Sriraman G, et al. A machine learning approach to predict DevOps readiness in organizations. Front Comput Sci. 2023.