

CI/CD Pipeline Optimization Using AI: A Systematic Mapping Study [†]

Redouan Farihane ^{*}, Imane Chlioui  and Maryam Radgui 

SI2M Laboratory, National Institute of Statistics and Applied Economics (INSEA),
P.O. Box 6217, Rabat 10112, Morocco; ichlioui@insea.ac.ma (I.C.); m.radgui@insea.ac.ma (M.R.)

^{*} Correspondence: redouanefarihane@gmail.com

[†] Presented at the 7th edition of the International Conference on Advanced Technologies for Humanity (ICATH 2025), Kenitra, Morocco, 9–11 July 2025.

Abstract

Continuous Integration/Deployment (CI/CD) pipelines are essential in the software engineering field, and improving automation and efficiency in this field requires optimizing them. AI has become a crucial tool for optimizing these pipelines at different stages. This research examines the application of AI approaches to CI/CD pipeline optimization using a systematic mapping study (SMS) to give a global overview of this field. We examined 92 papers published between 2015 and 2025 based on five main criteria: publication year and channel, research type, CI/CD pipeline stage, empirical method, and AI techniques used. The results show a notable increase in research efforts since 2018; conferences and journals are the most popular publication channels, and most of them are the solutions proposal type, which uses hypothesis-based experimentation as an empirical method for evaluation. Moreover, the testing stage of CI/CD is the most targeted using reinforcement learning algorithms.

Keywords: continuous integration; continuous deployment; artificial intelligence; AI; CI/CD pipeline; DevOps



Academic Editors: Brahim El Bhiri
and Ayman S. Mosallam

Published: 15 October 2025

Citation: Farihane, R.; Chlioui, I.;
Radgui, M. CI/CD Pipeline
Optimization Using AI: A Systematic
Mapping Study. *Eng. Proc.* **2025**, *112*,
32. <https://doi.org/10.3390/engproc2025112032>

Copyright: © 2025 by the authors.
Licensee MDPI, Basel, Switzerland.
This article is an open access article
distributed under the terms and
conditions of the Creative Commons
Attribution (CC BY) license
(<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Software engineering has made Continuous Integration/Deployment (CI/CD) an essential practice in recent years [1]. Such practices automate the tasks of DevOps, accelerate the software delivery process, and ensure quality by reducing human error [2]. However, organizations still face several challenges when adopting CI/CD pipelines because the first implementation is a difficult task that necessitates significant changes in company culture, development processes, and tools, particularly in situations with complex architecture or old systems [3]. Resource allocation also is a critical concern, as inefficient utilization can lead to prolonged build times, deployment delays, and increased operational costs, and error detection and debugging within dynamic CI/CD pipelines is another challenge [4]. Artificial intelligence is quickly advancing to change industries, including DevOps, thanks to its ability to analyze data at scale and automate tasks [5]. That's why the application of artificial intelligence in the pipeline CI/CD has been the subject of many studies and in different stages: Machine Learning (ML) detects anomalies [6], Deep Learning (DL) predicts build failures using historical data [7–9] while Reinforcement Learning (RL) automates prioritizing test cases [10,11]. However, there is a lack of publications providing an overview of this research area. This article aims to address this lack by systematically mapping the existing scientific literature on the application of AI in CI/CD. In this SMS, we present

a review of articles published in several academic databases (IEEE, ACM, SpringerLink, Scopus, ScienceDirect, and Google Scholar), and the results of this study give a complete picture of this area of research. The remaining sections are organized as follows: The Section 2 presents the methodology of research used; Section 3 reports the results and their discussion; Section 4 is dedicated to the recommendations for the researchers, and we end with a conclusion in Section 5.

2. Research Methodology

When engaging in a new research domain, researchers aim first to track advancements and identify key areas of interest [12]. A systematic mapping study offers a comprehensive overview of a specific domain by cataloging and categorizing the volume, type, and distribution of publications within that domain; their implementation requires following an established sequence of five steps: 1. Creating research questions; 2. Locating and researching pertinent articles; 3. Selecting papers based on defined criteria; 4. Extracting keywords from resumes; and 5. Extracting and mapping data. Each step of the process provides different outputs, with the final deliverable being the systematic map that effectively synthesizes key findings, patterns, and gaps in the literature of the researched field [13].

2.1. Research Questions

The major objective of this article is to provide a comprehensive summary of the studies on the application of AI to the optimization of CI/CD pipelines that have been conducted since 2015. We developed five research questions (RQs) for a thorough understanding of this topic; they are listed in Table 1 with their justification.

Table 1. Research questions and their justification.

ID	Research Questions	Justifications
RQ1	Where and in which year were the studies on optimizing CI/CD pipelines using AI published?	Determining publication channels and the timeline of research in AI-driven CI/CD pipeline optimization.
RQ2	Which stage of the pipeline CI/CD benefits most from AI techniques for their optimization?	Identifying in which stage of the CI/CD pipeline AI is most useful.
RQ3	What AI techniques have been applied to optimize CI/CD pipelines?	Identifying the most popular AI techniques used for CI/CD pipeline optimization.
RQ4	What types of studies have been published to optimize CI/CD pipelines?	Categorizing the types of contributions in AI-driven CI/CD pipeline optimization.
RQ5	What types of empirical research were employed to evaluate the optimization of CI/CD based on AI?	Determining the empirical study that is utilized to evaluate CI/CD optimization based on AI.

After establishing the research questions, we identify relevant studies addressing these RQs. The search process was carried out across several academic databases (IEEE, ACM, SpringerLink, Scopus, ScienceDirect, and Google Scholar) by using a search string formulated from the main terms and their synonyms used in the RQs, and major sections were joined using the Boolean (AND), and the alternate terms were joined using the Boolean (OR). The following is the search string defined: (“CI/CD” OR “CI pipeline” OR “CD pipeline” OR “Continuous Integration” OR “Continuous Deployment” OR “Continuous delivery” OR “DevOps pipeline” OR “DevOps workflows”) AND (“Artificial Intelligence” OR “Machine Learning” OR “Deep Learning” OR “Neural Networks” OR “Reinforcement Learning” OR “Natural Language Processing” OR “Large Language Models” OR “LLMs”

OR “Generative AI”) AND (“optimiz*” OR “efficien*” OR “automat*” OR “enhanc*” OR “improv*” OR “bottleneck detection” OR “risk prediction”).

2.2. Study Selection

The objective of this section is to identify candidate articles that best match the objectives of this SMS study. Table 2 shows the criteria for inclusion and exclusion; these criteria were applied to decide which papers to include or exclude based on their relation to the 5 research questions (RQs). A paper should be included if it satisfies at least one of the inclusion criteria and does not meet any exclusion criteria. These papers with a very clear title are assessed by reading only the abstract to determine if it should be included or not, and if the title is not very clear, we check the abstract and then the full text.

Table 2. Inclusion and exclusion criteria.

ID	Inclusion Criteria
IC1	Articles that use or suggest new AI methods for CI/CD pipeline optimization.
IC2	Papers that provide an overview of the use of AI in the optimization of the CI/CD pipeline.
IC3	Publications that compare empirically or theoretically the AI approaches in CI/CD optimization.
ID	Exclusion Criteria
EC1	Non-English papers.
EC2	Articles posted before 2015.
EC3	Papers are accessible only as abstracts or in PowerPoint.
EC4	Duplicate papers.

2.3. Data Extraction

In this part, a structured data extraction form was employed to collect the pertinent information from the selected papers to answer the research questions:

RQ1: Determine the year, channel, and source of each selected paper.

RQ2: The CI/CD pipeline includes six key stages [14]:

- Code Commit: developers start by committing their code into a source code repository.
- Test: a set of tests to confirm the quality and functionality of the code.
- Code Review: reviewed all commits before merging to avoid future problems.
- Build: compiles the code into executables or bytecode files and generates artifacts that are well deployed.
- Deploy: when the code passes all the tests, it is ready to be deployed to staging/deployment environments.
- Monitor: check if the application is functioning as expected after deployment.

RQ3: All AI techniques utilized in each selected study were identified.

RQ4: The chosen papers are classified into [15]:

- Evaluation Research (ER): articles that evaluate the AI approach used for optimizing CI/CD pipelines by introducing new or using existing algorithms.
- Solution Proposal (SP): papers that submit a proposal solution or architecture based on AI for optimizing CI/CD.
- Experience Papers (EP): when the researchers share and discuss their practical experiences with CI/CD pipeline optimization using AI.
- Review: articles that review and summarize the current AI use in CI/CD optimization.

RQ5: The type of empirical research is classified into 3 categories [16]:

- Survey: a series of questions is asked of the developers or DevOps engineers to see if there are any real benefits when using AI in the CI/CD pipeline.

- Historical-Based Evaluation: based on historical data from a study that manipulates AI in the CI/CD pipeline.
- Case Study: a study that evaluates their work empirically, such as the use of open-source projects.

2.4. Threats to Validity

- Study selection bias: although we applied rigorous selection criteria to ensure relevance, some relevant studies were missed. We included all other notable studies that we were able to identify by reviewing the references of the selected articles.
- Publication bias: since some researchers may exaggerate the efficacy of their AI models to show how efficient they are in comparison to others, we developed inclusion criterion 3 to focus also on studies comparing AI techniques to CI/CD pipeline improvements.
- Bias in data extraction: we began by reading the abstract if it was sufficient to discover the data needed to answer the RQs. If not, we continued reading the entire article to lower the possibility of inaccurate data extraction.

3. Results and Discussion

This section will present the process of the chosen papers, followed by the results and discussion of research questions RQs 1 to 5, listed in the previous section.

3.1. Studies Selection Process

As shown in Figure 1, 17,321 candidate articles were initially obtained using the search string applied to the 6 selected academic databases. We then applied the initial exclusion criteria (EC1 and EC2), and the number of selected articles became 17,080. We then performed an initial review of titles and abstracts and applied exclusion criteria EC3 and EC4, reducing the number of articles to 1343. This was followed by a more detailed review of abstracts and, if necessary, full-text reviews to eliminate articles not meeting any of the inclusion criteria (IC1, IC2, or IC3). As a result, 75 relevant articles were selected. Finally, the reference lists of these 75 articles were reviewed, identifying 17 additional relevant articles. 92 papers in total were chosen and examined to address the research questions.

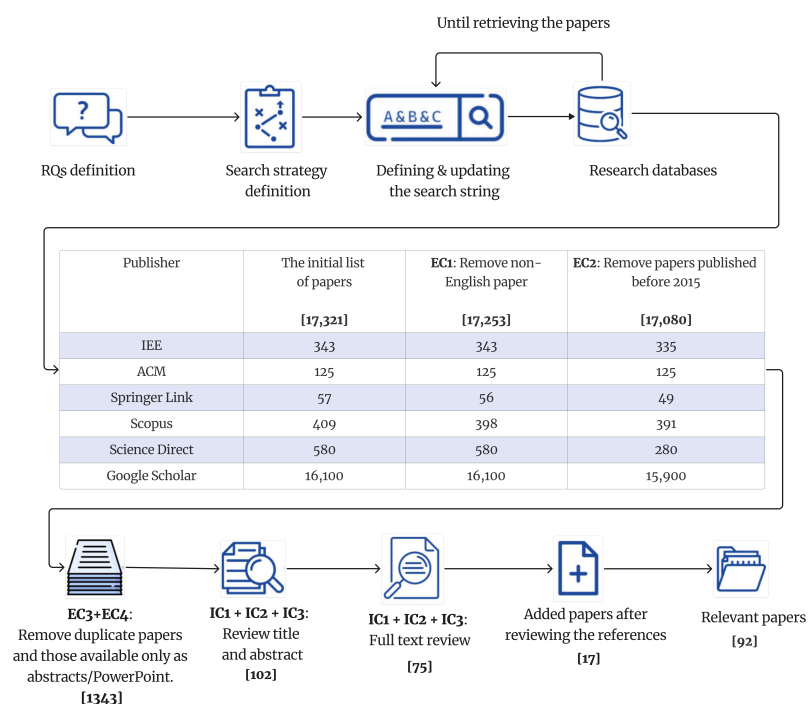


Figure 1. Selection process and results.

3.2. RQ1: Where and in Which Year Were the Studies on Optimizing CI/CD Pipelines Using AI Published?

Table 3 shows that 92 chosen studies were published in various venues. mostly journals and conferences. 47.82% of the papers appeared in journals, while 40.21% were published in conferences, and 11.97% of the research was published as technical reports, books, or other academic papers. We start with the journals; the two journals that target publications the most frequently are IEEE Access 4.35% and Information and Software Technology 4.35%. Regarding conferences, the most frequent venues are the ASE 4.35% and the QRS 3.26%.

Table 3. Sources of Publications.

Sources of Publications	# of Papers	(%)
Conferences		
International Conference on Automated Software Engineering (ASE).	4	4.35
International Conference on Software Testing Workshops (ICSTW).	3	3.26
International Conference on Software Quality and Reliability and Security (QRS).	2	2.17
International Conference on Automation of Software Test (AST).	2	2.17
Other Conferences	33	35.87
Total Conferences	44	47.82
Journals		
IEEE Access	4	4.35
Information and Software Technology.	4	4.35
Systems and Software.	2	2.17
IEEE Transactions on Software Engineering.	2	2.17
Other Journals	25	27.17
Total Journals	37	40.21
Other Sources	11	11.97

We can see from Figure 2, which visualizes the annual distribution of published papers between 2015 and 2025, that the papers published before 2017 were very limited, with 1.25% as the average. But, after 2017, a significant increase in the papers, reaching an average of 94.56%, and especially in the period from 2022 to 2025, the research in this area notably increased, representing 81.52% of the total selected papers, which explains the huge interest in research in this field, and this is also reinforced by publications in various sources mainly covering different areas of software engineering.

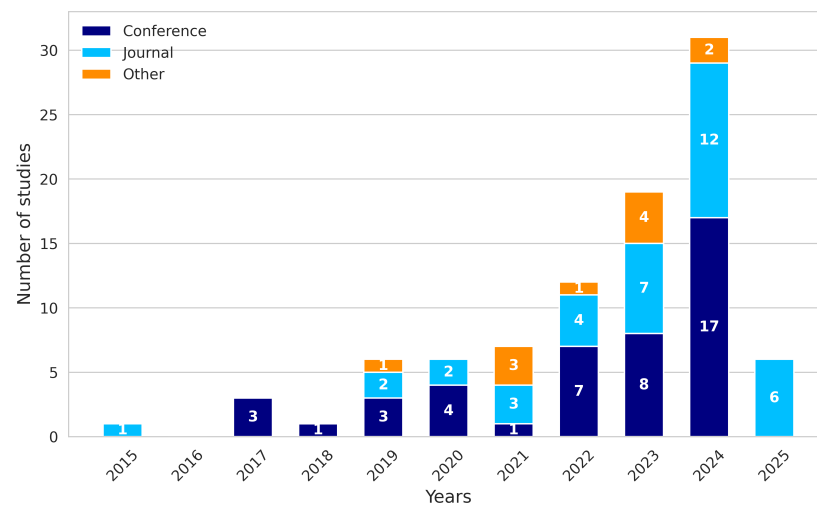


Figure 2. Channel of publication and number of papers published per year.

3.3. RQ2: Which Stage of the CI/CD Pipeline Benefits Most from AI Techniques for Their Optimization?

Figure 3 shows the six CI/CD pipeline stages most optimized by AI. The testing stage stands out as the most significant, representing 41.2% of the studies justified by AI helping to automate the tests and ensuring compliance with functional requirements. Because developing tests is a repetitive task, and not every commit needs checking all the project tests, that's why many studies talk about test case prioritization (TCP) using AI [17,18]. The build stage follows with 19.1%, it is less studied compared to the testing stage, but the need to address the build issues is found in some studies predicting build failures [8,19]. Others focus only on predicting build time [20]. While the deployment stage represents 14%, researchers aim to optimize the deployment process while ensuring system stability [21]. 7.35% represents code review of the selected studies, which is generally used to automate code review [22]. 5.88% represents the integration stage of the selected studies, which is used to automate tool tasks to facilitate the integration process [23]. 5.15% represents each code commit and monitoring stage of the selected studies, which is generally used to suggest code optimizations and also for log analysis [9].

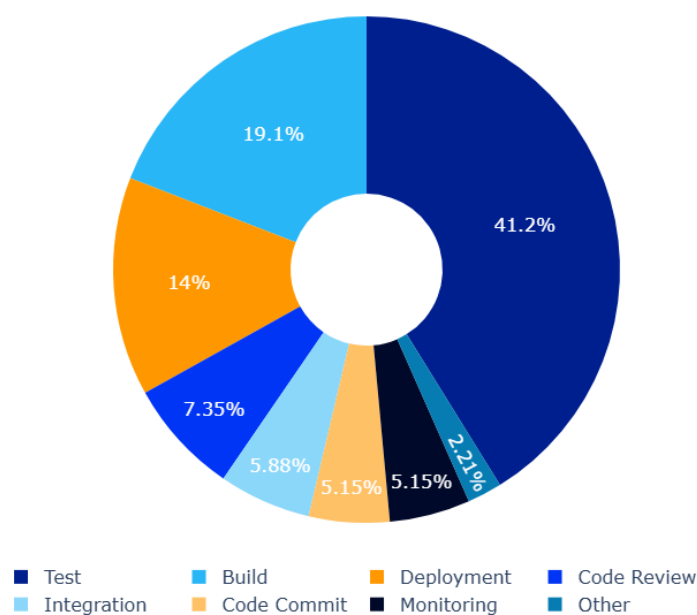


Figure 3. CI/CD pipeline stages are most automated by AI.

3.4. RQ3: What AI Techniques Have Been Applied to Optimize CI/CD Pipelines?

From the 92 articles selected in this study, researchers applied various AI algorithms to optimize one or all stages of the CI/CD pipeline. Figure 4 shows that the RL is the most common, making up 44.64%, and random forests 14.24%, SVM and decision trees follow, both at 8.93%, of algorithms used in the test stage. Other techniques, like LSTM, NLP, and LLMs, are less frequent, each at just under 8% in this stage. AI techniques are uniformly distributed in the build stage; LSTM and decision trees are the most used, representing 15.38%, random forest, SVM, RL, and LLMs follow with 11.53%. For the deployment stage, RL appears again, representing 26.67%; neural networks, decision trees, random forests, NLP, and LLMs each account for 13.33%. LLMs are the most used at the code review stage, appearing with 22.22%, but there are also neural networks, decision trees, random forests, and NLP. In the code commit stage, the distribution of AI approaches is more consistent; neural networks, NLP, decision trees, and RL each appear in 14.29%, while LLMs, LSTM, and SVM are slightly lower at 9.52%. In the integration stage, the LLMs are the most used, making up 37.5%; other techniques, such as decision trees, SVM, and RL, each appear in 12.5%. The monitoring stage has lower AI usage; LSTM and SVM are the most common at 20%, and Decision trees, NLP, and RL each makeup 10%. Decision trees, NLP, and RL are also useful in other stages present, each accounting for 16.67%. We can explain why RL is the most commonly used, especially in the test stage, due to its ability to prioritize and automate testing in real time within a dynamic environment [11,18,24]. The models, such as neural networks [8,25] and random forests [26] and SVM [20,27] are also commonly applied, mostly for predicting build failures, estimating build time, and optimizing deployment processes to maintain system stability because these algorithms capture complex feature interactions. LSTMs are applied for tasks that handle sequence data, like log analysis [7,28]. NLP and LLMs have been explored for automating code review and analyzing large amounts of textual data generated due to their ability to understand and generate natural language [29].

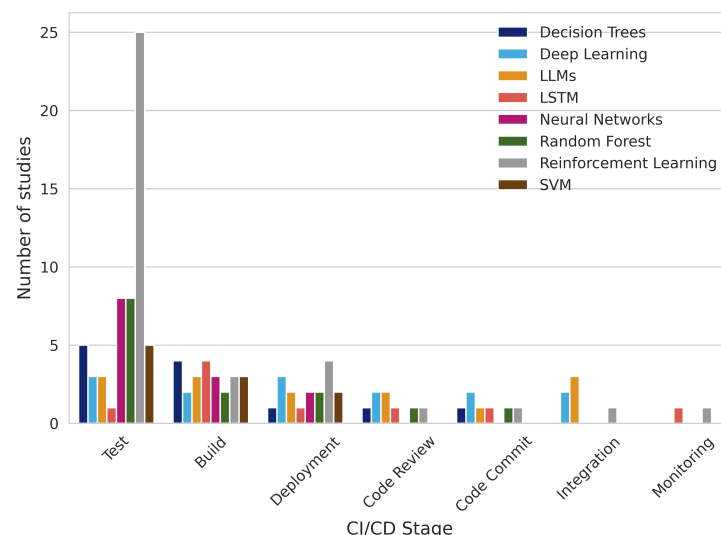


Figure 4. Distribution of AI techniques applied in CI/CD stages.

3.5. RQ4: What Types of Studies Have Been Published to Optimize CI/CD Pipelines?

During our research, we identified 4 research types as illustrated in Figure 5: 83.69% of the chosen articles were SP that proposed new or enhanced old approaches, and the first SP occurred in 2017 and has grown over the years because researchers were more focused on creating new or improving existing AI techniques for CI/CD optimization [17]. However, all other research types have only emerged since 2022 and continued through;

ER represents 9.78% of the selected articles that evaluate or compare existing AI already proposed since 2017, while only 1.08% were identified as EP [30], this highlights a clear gap of practical experience, which should be given greater emphasis. 4.43% of the chosen papers were classified as Re to give a summary of this research field after more than seven years of publications in it [31].

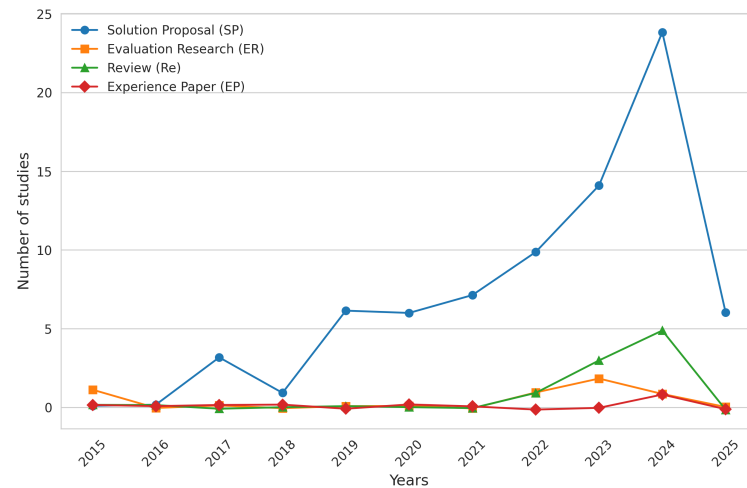


Figure 5. Evolution over the years of the types of research identified.

3.6. RQ5: What Types of Empirical Research Were Employed to Evaluate the Optimization of CI/CD Based on AI?

As shown in Figure 6, 80.43% (79 out of 92) of the selected studies were evaluated using various empirical methods. Specifically, 41.77% (33 out of 79) are categorized as Hypothesis-Based Experimentation (HBE), meaning that these studies used open-source projects to validate their hypotheses; 15.15% (5 out of 33) of them evaluate existing AI techniques to optimize CI/CD; and 84.85% (28 out of 33) propose a new solution [32]. Case studies were used for evaluating new AI techniques in 90% of cases (27 out of 30) [17], reviews in 6.67% (2 out of 30), and existing solutions in only 3.33% (1 out of 30). Only 1.26% of the studies (1 out of 79) used a survey to evaluate the research [33]. 16.30% of empirical papers did not mention the type of empirical evaluation used. Finally, 14.13% (13 out of 92) of the studies were not empirically evaluated: 53.85% were review papers, and 46.15% were solution proposals. This indicates that research favors empirical validation, primarily through case studies and experiments, to ensure the credibility and effectiveness of proposed solutions, and this is due to the nature of this field.

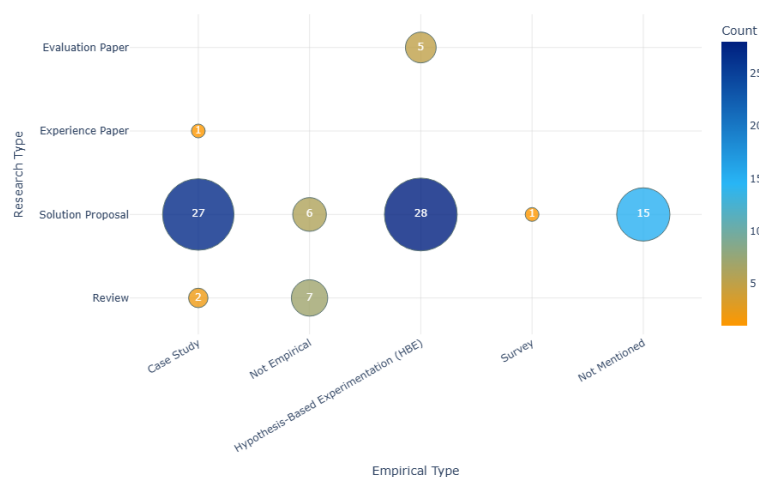


Figure 6. The empirical and research types were identified.

4. Recommendations for Researchers

The objective of this part is to give researchers the following recommendations: Regarding research question RQ2, many existing studies focus on continuous integration. Therefore, it's necessary to focus on other stages that also play a crucial role in the DevOps process and software quality. Regarding research question RQ3, which shows that research in CI/CD using AI has not yet exploited the novelties of AI, the AI techniques that are most commonly used are the popular machine/deep learning algorithms, and only some researchers are using AI generative techniques, and this last one demonstrates its effectiveness. Regarding research question RQ4, there is a large gap in experimental studies, and we found only one experimental study that indicated the need for further research.

5. Conclusions

This systematic map analyzes and synthesizes 92 articles published between 2015 and 2025 that applied the AI approach in optimizing the CI/CD pipeline. AI techniques proved their efficacy in multiple stages, notably testing, and particularly after 2017, the results indicated that many researchers were interested in applying AI techniques like reinforcement learning, neural networks, LSTM, SVM, and NLP. In addition, the chosen articles appeared in various sources specialized in the software engineering field, and only one paper concentrates on experience-based research, and the majority are solution proposals. The most often used empirical method to evaluate AI-based optimizations in CI/CD pipelines is hypothesis-based experimentation. A comprehensive systematic literature review would be a natural extension of this work to detail the most effective AI techniques for optimizing CI/CD pipelines while highlighting the technical and architectural limits to integrating AI into these pipelines.

Author Contributions: Conceptualization, R.F.; Methodology, R.F.; Software, R.F.; Formal analysis, R.F.; Visualization, R.F.; Validation, R.F.; Writing—original draft preparation, R.F.; Writing—review and editing, R.F., I.C. and M.R.; Resources, R.F., I.C. and M.R.; Supervision, I.C. and M.R.; Project administration, R.F., I.C. and M.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Humble, J.; Farley, D. *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation*; Addison-Wesley: Upper Saddle River, NJ, USA, 2010.
2. Shahin, M.; Ali Babar, M.; Zhu, L. Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices. *IEEE Access* **2017**, *5*, 3909–3943. [[CrossRef](#)]
3. Fitzgerald, B.; Stol, K.J. Continuous Software Engineering: A Roadmap and Agenda. *J. Syst. Softw.* **2017**, *123*, 176–189. [[CrossRef](#)]
4. Hilton, M.; Tunnell, T.; Huang, K.; Marinov, D.; Dig, D. Usage, Costs, and Benefits of Continuous Integration in Open-Source Projects. In Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, Singapore, 3–7 September 2016; pp. 426–437. [[CrossRef](#)]
5. Shalev-Shwartz, S.; Ben-David, S. *Understanding Machine Learning: From Theory to Algorithms*; Cambridge University Press: New York, NY, USA, 2013. [[CrossRef](#)]
6. Hany Fawzy, A.; Wassif, K.; Moussa, H. Framework for Automatic Detection of Anomalies in DevOps. *J. King Saud Univ.- Comput. Inf. Sci.* **2023**, *35*, 8–19. [[CrossRef](#)]

7. Saidani, I.; Ouni, A.; Mkaouer, M. Improving the Prediction of Continuous Integration Build Failures Using Deep Learning. *Autom. Softw. Eng.* **2022**, *29*, 21. [\[CrossRef\]](#)
8. Santolucito, M.; Zhang, J.; Zhai, E.; Cito, J.; Piskac, R. Learning CI Configuration Correctness for Early Build Feedback. In Proceedings of the 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), Honolulu, HI, USA, 15–18 March 2022; pp. 1006–1017. [\[CrossRef\]](#)
9. Myllynen, T.; Kamau, E.; Mustapha, S.D.; Babatunde, G.O.; Collins, A. Review of Advances in AI-Powered Monitoring and Diagnostics for CI/CD Pipelines. *Int. J. Multidiscip. Res. Growth Eval.* **2024**, *5*, 1119–1130. [\[CrossRef\]](#)
10. Yang, Y.; Pan, C.; Li, Z.; Zhao, R. Adaptive Reward Computation in Reinforcement Learning-Based Continuous Integration Testing. *IEEE Access* **2021**, *9*, 36674–36688. [\[CrossRef\]](#)
11. Yaraghi, A.; Bagherzadeh, M.; Kahani, N.; Briand, L. Scalable and Accurate Test Case Prioritization in Continuous Integration Contexts. *IEEE Trans. Softw. Eng.* **2023**, *49*, 1615–1639. [\[CrossRef\]](#)
12. Haneem, F.; Ali, R.; Kama, N.; Basri, S. Descriptive Analysis and Text Analysis in Systematic Literature Review: A Review of Master Data Management. In Proceedings of the 2017 International Conference on Research and Innovation in Information Systems, ICRIS, Langkawi, Malaysia, 16–17 July 2017. [\[CrossRef\]](#)
13. Petersen, K.; Feldt, R.; Mujtaba, S.; Mattsson, M. Systematic Mapping Studies in Software Engineering. In Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), University of Bari, Italy, 26–27 June 2008. [\[CrossRef\]](#)
14. Jani, Y. Implementing Continuous Integration and Continuous Deployment (CI/CD) in Modern Software Development. *Int. J. Sci. Res.* **2023**, *12*, 2984–2987. [\[CrossRef\]](#)
15. Brereton, P.; Kitchenham, B.A.; Budgen, D.; Turner, M.; Khalil, M. Lessons from Applying the Systematic Literature Review Process within the Software Engineering Domain. *J. Syst. Softw.* **2007**, *80*, 571–583. [\[CrossRef\]](#)
16. Condori-Fernandez, N.; Daneva, M.; Sikkil, K.; Wieringa, R.; Dieste, O.; Pastor, O. A Systematic Mapping Study on Empirical Evaluation of Software Requirements Specifications Techniques. In Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement, Lake Buena Vista, FL, USA, 15–16 October 2009; pp. 502–505. [\[CrossRef\]](#)
17. Spieker, H.; Gotlieb, A.; Marijan, D.; Mossige, M. Reinforcement Learning for Automatic Test Case Prioritization and Selection in Continuous Integration. In Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis, Santa Barbara, CA, USA, 10–14 July 2017; pp. 12–22. [\[CrossRef\]](#)
18. Su, Q.; Li, X.; Ren, Y.; Qiu, R.; Hu, C.; Yin, Y. Attention Transfer Reinforcement Learning for Test Case Prioritization in Continuous Integration. *Appl. Sci.* **2025**, *15*, 2243. [\[CrossRef\]](#)
19. Benjamin, J.; Mathew, J.; Jose, R.T. Study of the Contextual Factors of a Project Affecting the Build Performance in Continuous Integration. In Proceedings of the 2023 Annual International Conference on Emerging Research Areas: International Conference on Intelligent Systems (AICERA/ICIS), Kanjirapally, India, 16–18 November 2023; pp. 1–6. [\[CrossRef\]](#)
20. Bisong, E.; Tran, E.; Baysal, O. Built to Last or Built Too Fast? Evaluating Prediction Models for Build Times. In Proceedings of the 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), Buenos Aires, Argentina, 20–21 May 2017; pp. 487–490. [\[CrossRef\]](#)
21. Kaluvakuri, V.P.K. AI-Powered Continuous Deployment: Achieving Zero Downtime and Faster Releases. *SSRN J.* **2023**. [\[CrossRef\]](#)
22. Ochodek, M.; Staron, M. ACoRA—A Platform for Automating Code Review Tasks. *E-Inform. Softw. Eng. J.* **2025**, *19*, 250102. [\[CrossRef\]](#)
23. Zhang, X.; Muralee, S.; Cherupattamoolayil, S.; Machiry, A. On the Effectiveness of Large Language Models for GitHub Workflows. In Proceedings of the 19th International Conference on Availability, Reliability and Security (ARES'24), Vienna, Austria, 30 July–2 August 2024; pp. 1–14. [\[CrossRef\]](#)
24. Cao, T.; Li, Z.; Zhao, R.; Yang, Y. Historical Information Stability Based Reward for Reinforcement Learning in Continuous Integration Testing. In Proceedings of the 2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS), Hainan, China, 6–10 December 2021. [\[CrossRef\]](#)
25. Kawalerowicz, M.; Madeyski, L. Continuous Build Outcome Prediction: A Small-N Experiment in Settings of a Real Software Project. In *Advances and Trends in Artificial Intelligence. From Theory to Practice*; Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Cham, Switzerland, 2021; Volume 12799, pp. 412–425. [\[CrossRef\]](#)
26. Arani, A.K.; Zahedi, M.; Le, T.H.M.; Babar, M.A. SoK: Machine Learning for Continuous Integration. In Proceedings of the 2023 IEEE/ACM International Workshop on Cloud Intelligence and AIOps, Melbourne, Australia, 15 May 2023. [\[CrossRef\]](#)
27. S.R., D.; Mathew, J. Optimizing Continuous Integration and Continuous Deployment Pipelines with Machine Learning: Enhancing Performance and Predicting Failures. *Adv. Sci. Technol. Res. J.* **2025**, *19*, 108–120. [\[CrossRef\]](#) [\[PubMed\]](#)
28. Benjamin, J.; Mathew, J. Enhancing Continuous Integration Predictions: A Hybrid LSTM-GRU Deep Learning Framework with Evolved DBSO Algorithm. *Computing* **2024**, *107*, 9. [\[CrossRef\]](#)

29. Chen, T. Challenges and Opportunities in Integrating LLMs into Continuous Integration/Continuous Deployment (CI/CD) Pipelines. In Proceedings of the 2024 5th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT), Nanjing, China, 29–31 March 2024. [[CrossRef](#)]
30. Zhao, Y.; Hao, D.; Zhang, L. Revisiting Machine Learning Based Test Case Prioritization for Continuous Integration. In Proceedings of the 2023 IEEE International Conference on Software Maintenance and Evolution (ICSME), Bogotá, Colombia, 1–6 October 2023; pp. 232–244. [[CrossRef](#)]
31. Arani, A.K.; Le, T.H.M.; Zahedi, M.; Babar, M.A. Systematic Literature Review on Application of Learning-Based Approaches in Continuous Integration. *IEEE Access* **2024**, *12*, 135419–135450. [[CrossRef](#)]
32. Yang, L.; Xu, J.; Zhang, H.; Wu, F.; Lyu, J.; Li, Y.; Bacchelli, A. GPP: A Graph-Powered Prioritizer for Code Review Requests. In Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering (ASE '24), Sacramento, CA, USA, 27 October–1 November 2024; ACM: New York, NY, USA, 2024; pp. 104–116. [[CrossRef](#)]
33. Sharma, P.; Kulkarni, M.S. A Study on Unlocking the Potential of Different AI in Continuous Integration and Continuous Delivery (CI/CD). In Proceedings of the 2024 4th International Conference on Innovative Practices in Technology and Management (ICIPTM), Noida, India, 21–23 February 2024; pp. 1–6. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.