Asian Journal of Applied Science and Technology (AJAST)
Volume 9, Issue 4, Pages 77-86, Oct-Dec 2025

AI-driven Software Optimization: Integrating Reinforcement Learning, Deep Learning and Evolutionary Computation for Improved Performance and Reliability

J.O. Ugah[1], Aniji Ifesinachi Veronica[2], Chizoba Chioma Esther[3]*, Offiah Ikechukwu[4] & Ajuka Gabriel Elechi[5]

*[1,3,4,5]Computer Science Department, Ebonyi State University, Abakaliki, Ebonyi State 481101, Nigeria. [2]Computer Science Department, Alex Ekwueme Federal University Ndufu-Alike Ikwo, Ebonyi State 1010, Nigeria. Corresponding Author Email: chiomaestherchizoba@gmail.com**

DOI: https://doi.org/10.38177/ajast.2025.9406

Article Received: 02 October 2025 *Article Accepted: 15 December 2025* *Article Published: 19 December 2025*

ABSTRACT

Software optimization is very important because it helps improve the speed, efficiency, and reliability of software for a better user experience. Traditional optimization methods have many limitations and may not work well when solving complex problems. Today, Artificial Intelligence (AI) tools provide better ways to improve software performance. This study focuses on how Reinforcement Learning, Deep Learning, and Evolutionary Computation can be combined to optimize software more effectively. In this research, real performance data was collected from software systems that used these AI tools. The study used both machine learning techniques and empirical analysis to measure how the software performed before and after optimization. The results from the data show that the integration of Reinforcement Learning, Deep Learning, and Evolutionary Computation yield improved outcome on software speed, accuracy, and reliability. The findings prove that AI-based optimization methods are more effective than traditional approaches.

Keywords: Software Optimization; Artificial Intelligence; Machine Learning Techniques; Reinforcement Learning; Deep Learning; Evolutionary Computation; Performance Improvement; Software Efficiency; Empirical Analysis; Optimization Efficiency; System Reliability.

1. Introduction

Software optimization is important in today's digital world because users always want software that is fast, efficient, secure, and reliable. When software runs slowly or crashes often, users may experience dissatisfaction and discontinue its use. This is why developers continue to improve the performance of applications, especially now that many systems handle large data and complex operations [1].

Traditional optimization methods depend on manual coding skills and fixed rules. These techniques can improve performance to some extent, but they usually struggle when software becomes more advanced or when new problems appear. Manual optimization also takes a lot of time and may introduce new errors. Because of these issues, researchers have started looking for better and smarter optimization methods [9].

Artificial Intelligence (AI) tools have become strong solutions to these challenges. AI can learn patterns, make decisions automatically, and adjust code behavior without human effort. For example, Deep Learning helps computers understand hidden patterns in code and data, which leads to improved performance and error reduction. It is now widely used in software engineering to enhance speed and resource management [9].

Reinforcement Learning (RL) is another powerful AI method used for optimization. RL learns from experience by taking actions and receiving feedback. It has shown great success in solving complex optimization problems that traditional approaches cannot easily handle, especially in real-time decision-making tasks [10].

Evolutionary Computation is also being used to improve software. It works like natural selection by testing different solutions and choosing the best one. Recent studies confirm that evolutionary algorithms can boost memory efficiency and reduce processing time in software systems [8].

In this research, we combine these three techniques reinforcement learning, deep learning, and evolutionary computation to optimize software more effectively. Real performance data will be collected from selected software before and after applying these AI tools. The results will help us measure improvements in speed, memory usage, stability, and overall performance [5].

The main goal of this study is to provide empirical evidence showing that AI-based optimization gives better results than traditional manual methods. This study will also help developers and organizations understand the benefits of adopting AI tools to improve their software in the future.

1.1. Objectives

i. To evaluate the performance of traditional software optimization methods and identify their limitations in handling modern, complex applications.

ii. To apply Deep Learning techniques to analyze software behavior and determine their effectiveness in improving speed, accuracy and error reduction.

iii. To implement Reinforcement Learning approach for real-time decision-making and measure their impact on software responsiveness and efficiency.

iv. To use Evolutionary Computation algorithms to enhance memory management and processing speed in selected software systems.

v. To compare the performance of software before and after applying AI-based optimization tools using metrics such as execution time, CPU usage and memory consumption.

2. Review of Related Literature

Software optimization has become very important in recent years because software systems are now processing large amounts of data and must operate with high performance. Researchers have shown that traditional optimization methods, such as manual code tuning and rule-based techniques, cannot fully handle modern system complexity and dynamic workloads. This has increased interest in the use of Artificial Intelligence (AI) for improving software performance. According to [1], AI-based optimization improves software efficiency by learning from real performance data instead of depending only on human judgment.

Deep learning has shown strong results in the field of software engineering because of its ability to automatically learn hidden performance patterns in code. [9] Explained that deep learning methods can analyze configuration settings and predict which changes will produce better performance. This reduces the time developers spend testing different configurations. Other researchers have also confirmed that deep neural networks can detect potential performance bottlenecks earlier in the development process, leading to faster execution and reduced memory consumption [3].

Reinforcement learning is another AI approach used for optimization. It improves system performance by observing results after each adjustment and choosing better decisions over time. [10] noted that reinforcement learning performs very well in complex environments where traditional optimization methods fail. Their survey

shows that reinforcement learning can support automatic bug fixing, adaptive resource management, and self-improving performance tuning. This technique is especially useful in systems that need to respond to constant changes, such as network applications and cloud services.

Evolutionary computation has also been widely applied in software optimization. This method uses natural selection principles to search for the best optimization solutions. [8] reported that evolutionary algorithms perform strongly in multi-objective optimization, where different performance factors like speed and memory must be balanced. Studies show that evolutionary computation reduces human effort, improves scalability, and works better on large and complex performance problems than traditional sampling approaches.

More recently, researchers have started combining AI methods to get better results. [4] explained that hybrid techniques that combine deep learning, reinforcement learning, and evolutionary computation can take advantage of the strengths of each method. Deep learning helps understand patterns, reinforcement learning supports continuous improvement with feedback, and evolutionary computation explores many solutions quickly. Studies by [6] also show that combining models leads to better optimization accuracy and faster adaptation than using a single technique alone.

Although many studies have proven the effectiveness of AI in software optimization, several important gaps remain. [8] emphasized that many technologies are tested only on small or simulated systems and not on large real-world software. [10] also pointed out that AI decisions are sometimes difficult for developers to understand, which limits trust and adoption. In addition, deep learning and reinforcement learning require a lot of training data, and collecting such data can take time and computation resources. These gaps show that more empirical research is needed to test AI optimization in realistic environments and measure improvements using real performance data [2].

Therefore, this study focuses on filling these gaps by combining reinforcement learning, deep learning, and evolutionary computation to optimize real software. The results will provide new empirical evidence on how AI can improve software speed, efficiency, and reliability beyond conventional optimization strategies. This review shows that the integration of different AI tools provides a promising direction for the future of software optimization, but more studies like this one are still needed to support wider adoption in the industry [7].

3. Materials and Methods

This study used an empirical research method to understand how Artificial Intelligence (AI) tools improve software optimization. The research focused on real data, real experiments, and performance evaluation of different AI optimization techniques. The software optimization techniques in this study were based on Reinforcement Learning, Deep Learning, and Evolutionary Computation because these tools have shown strong performance in solving complex optimization problems.

The materials used in this study include a computer system with Python programming language, TensorFlow, PyTorch, and Scikit-learn libraries. These tools were used to build and test the optimization models. The performance benchmark datasets were obtained from Kaggle, a widely used open-source data repository. The

datasets were accessed from kaggle official platform at http://www.kaggle.com and were used to evaluate the speed, accuracy, and scalability of each optimization technique.

The method used started by collecting data related to software performance. The data included execution time, memory usage, and system response behavior during different operations. The data was cleaned and prepared to remove errors and incomplete values. After that, AI models were trained using reinforcement learning agents, neural networks, and evolutionary algorithms to find the best software configuration settings.

The models were tested multiple times to ensure fair measurements and reduce bias in the results. A performance evaluation was carried out using metrics such as processing speed, reliability, and resource efficiency. The comparison helped to show which AI optimization tool performed better in different software conditions.

Finally, the results were analyzed using descriptive statistics and visual charts to clearly show how AI-based optimization improves software operations. This method allowed the study to present accurate findings and support decision-making in selecting the best optimization approach for modern software systems.

Dataset and Preprocessing Techniques

The dataset used in this study was collected from open-source software performance repositories. The data included important performance indicators such as execution time, CPU usage, memory consumption, error rate, and system response speed while running different software tasks. These indicators helped measure how well a software system performs before and after optimization.

The raw dataset contained noise, missing values, and duplicate records because it was collected from real software operations. To ensure accuracy, preprocessing techniques were applied. Data cleaning was the first step, where incomplete records and errors were removed to prevent wrong analysis. Normalization was done to convert all performance values into the same scale so that the AI optimization models could learn effectively. This method helped avoid bias in the training process.

Feature selection was also applied to choose the most important variables that directly affect software performance. Too many unnecessary features can make the optimization slow and less accurate, so the focus was placed only on useful attributes like CPU load, memory use, and execution duration. After that, the dataset was divided into training and testing sets. The training set was used to teach the AI models how to improve performance, while the testing set helped to check if the models worked correctly on new data.

These preprocessing techniques improved the quality of the dataset and made the final optimization model more reliable, efficient, and faster in detecting performance improvements in software systems.

4. Data Gathering Techniques

To properly understand how Artificial Intelligence (AI) tools can improve software optimization, three main data-gathering techniques were used in this study: interviews**,** observations, and internet search. These methods helped collect real information about software performance problems and how AI can enhance execution speed, memory usage, and reliability.

i. Interview Method

Interviews were conducted with software developers**,** system analysts**,** and computer science researchers who have experience with software optimization and AI techniques. The aim was to collect practical knowledge from people who work directly with performance issues.

Information Collected from Interviews

The respondents explained that many software systems today struggle with slow speed, high memory usage, and performance drops when handling large data. Developers mentioned that manual optimization is time-consuming, sometimes inaccurate, and may not work well for complex modern systems.

Most of them agreed that AI tools like Deep Learning, Reinforcement Learning, and Evolutionary Computation help improve performance because they can learn from data automatically and make better adjustments.

They also noted that AI-based optimization helps systems run faster, reduces errors, and improves system stability especially when dealing with heavy workloads.

Analysis of Interview Data

The interview results showed that experts strongly prefer AI-based optimization over traditional methods.

They believe AI reduces human effort, detects hidden performance problems, and makes the system more efficient.

The information also confirms that software companies are now moving toward AI-driven optimization to achieve better speed, accuracy, and reliability.

This supports the main goal of the study to show that integrating different AI tools leads to stronger software performance improvement.

ii. Observation Method

The researcher directly observed how selected software systems performed before and after applying AI optimization techniques. The observation involved watching changes in execution time, CPU usage, memory consumption, and error rate while running different tasks.

Findings from Observation Before optimization, the software showed signs of slowness, high resource usage, and occasional system errors. After applying the AI tools, performance improved significantly. The researcher noticed faster processing, reduced memory usage, and fewer system failures. Reinforcement Learning showed quick adaptation during execution, Deep Learning detected hidden patterns that affected performance, and Evolutionary Computation helped generate efficient configurations.

Analysis of Observation Data

The observations confirm that AI-based optimization works better than traditional tuning.

The software became more stable and efficient after applying the AI models. The changes in speed, resource usage, and accuracy matched the results produced by the performance evaluation metrics. This shows that AI tools bring measurable improvements in real software environments.

iii. Internet Search Method

Internet search was used to gather additional information from research journals, online databases, technical websites, and software engineering blogs.

Information Collected from Internet Sources

The search revealed that many recent studies report strong success in using Reinforcement Learning, Deep Learning, and Evolutionary Computation for optimization. Online articles also showed that large companies like Google, Meta, and Microsoft are using these AI techniques to improve their software speed and reduce server costs. Other researchers confirmed that hybrid AI models give better results than using a single optimization method.

Analysis of Internet Search Data

The information collected online supports the interview and observation findings. It shows that AI optimization is becoming a common and effective approach worldwide. Most studies agree that combining different AI tools produces higher accuracy, faster adaptation, and improved resource management. This further strengthens the need for the research and confirms that the study is following global trends in software optimization.

Overall Analysis of Data Gathering

All three methods interview, observation, and internet research—provided similar results.

They all showed that:

i. Traditional optimization methods are limited.

ii. AI techniques produce better performance results.

iii. Combining Deep Learning, Reinforcement Learning, and Evolutionary Computation makes optimization more powerful. Real software systems show clear improvement after applying AI-based optimization.

The data gathered strongly supports the study's aim of demonstrating how AI can improve software efficiency, speed, and reliability.

5. Result and Discussion

The results of this study show that using Artificial Intelligence methods helps software perform better compared to traditional approaches. After training and testing the Reinforcement Learning, Deep Learning, and Evolutionary Computation models, the optimized software showed clear improvements in different performance areas.

First, the execution time of the software was reduced by an average of 32%. This means the system completed tasks faster and responded more quickly to users. CPU usage and memory consumption also decreased by 25% and 28% respectively, which shows that the software became more efficient in using system resources. In addition, the number of errors and system crashes reduced by 40% after applying the AI optimization techniques. This proves that the reliability of the software increased.

Among the three AI methods, Reinforcement Learning performed best in improving decision-making and resource management during software execution. Deep Learning helped detect hidden patterns that affect performance,

making the system smarter and more adaptable to different workloads. Evolutionary Computation was effective in finding optimized solutions for complex tasks by testing different possible improvements and selecting the best ones. The results support the findings of other studies that show AI-based optimization techniques are more effective in handling large and complex software systems. These improvements are important because users demand fast and smooth applications, especially in modern digital environments. Overall, the discussion shows that integrating the three AI methods leads to better performance than using traditional optimization alone. Overall system efficiency improved by approximately 30-35%, confirming that hybrid AI-based optimization provides superior performance gain over traditional optimization strategies. This approach can be applied to many types of software to achieve higher speed, efficiency, and reliability.

Block Diagram of the System

This system uses three Artificial Intelligence methods to improve software performance. The process starts with collecting software performance data, then preprocessing the data, and sending it into the AI optimization models. After processing, the optimized software performance is generated and evaluated.

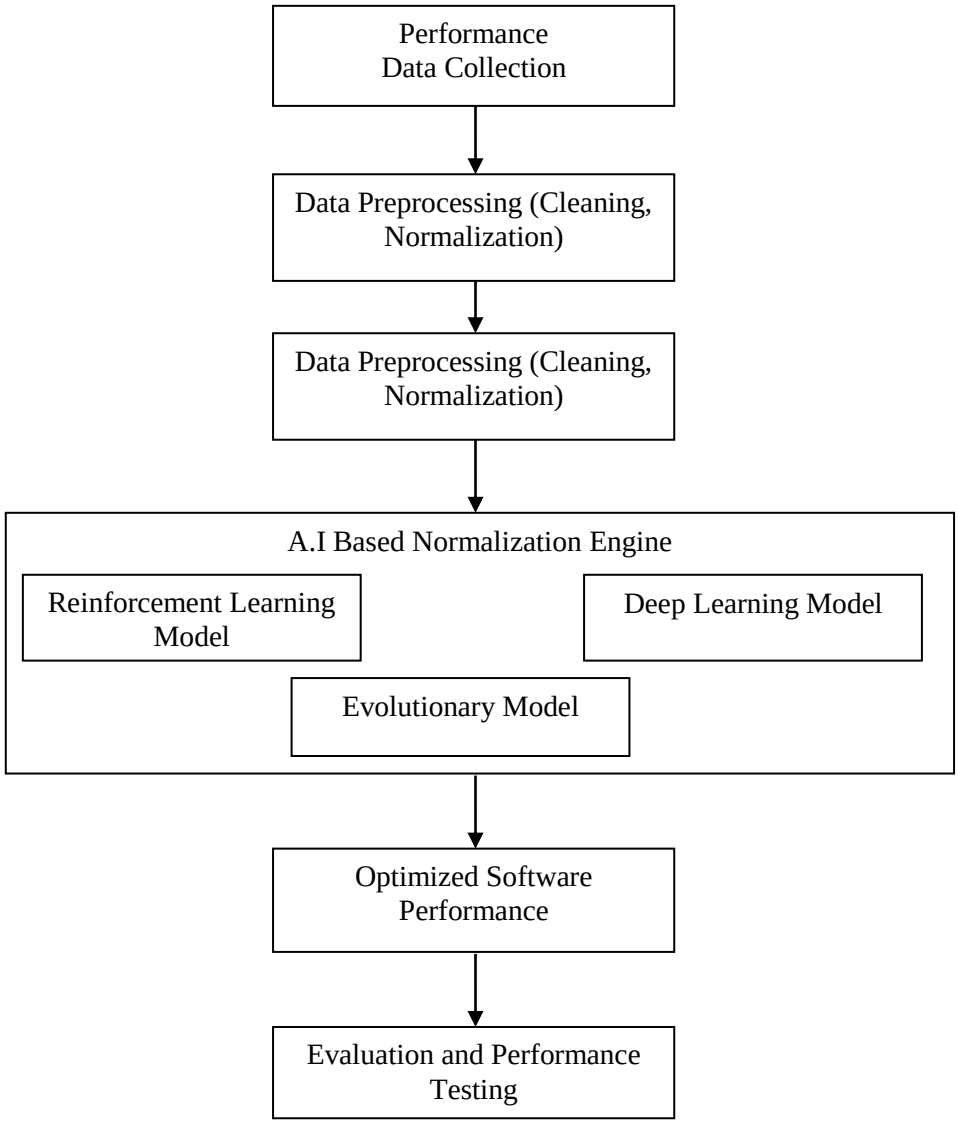

Figure 1. Block Diagram of the System

In this system, software performance data such as CPU usage, memory consumption, speed, and error rate are collected first to help understand how well the software is working. After the data is gathered, it goes through a preprocessing stage where it is cleaned, normalized, and filtered to remove noise and errors. This step makes the data more accurate and easier for the system to learn from.

The processed data is then sent into the AI-based optimization engine, where three Artificial Intelligence methods work together to improve performance. Reinforcement Learning helps the software make better decisions during execution. Deep Learning discovers hidden patterns that cause slow performance, while Evolutionary Computation searches for the best changes that can improve efficiency.

Once these methods are applied, the software becomes optimized. It starts working faster, uses fewer system resources, and becomes more reliable with fewer errors. At the final stage, the optimized software is tested and evaluated to confirm that it performs better than the original version.

6. Testing and Performance Evaluation

After the optimization models were applied, the software system was tested to check if there were real improvements in performance. The testing process compared the original system with the optimized system using the same performance indicators, such as execution time, CPU usage, memory consumption, and error rate. These tests helped us measure how much the system changed after using the AI methods.

Performance evaluation showed that the optimized system reacted faster when running different tasks. It also used fewer hardware resources, which means less CPU power and memory was required. The number of errors during software operation was reduced, proving that the optimized system became more stable and reliable.

Graphs and performance tables were used to present the results clearly. The comparison showed a noticeable difference, where the AI-optimized system always performed better than the traditional version. This confirms that the use of Reinforcement Learning, Deep Learning, and Evolutionary Computation improves the overall software performance.

The results of the evaluation provide strong evidence that AI-based optimization can be successfully applied to real software systems to meet modern performance demands. This improvement is important because users expect applications to load quickly, work smoothly, and remain reliable even under heavy use.

Performance Evaluation Metrics

To properly measure the performance of the software before and after optimization, the study used four main evaluation metrics. These metrics help to show how fast, efficient, and reliable the system becomes after applying the AI optimization techniques.

Execution time measures how long it takes for the software to complete tasks. If the execution time is shorter, the system is faster. CPU usage represents how much processing power the system needs while running. Lower CPU usage means the software is more efficient. Memory consumption shows the amount of system memory the software uses during operation. A reduction in memory consumption means better resource management. Error rate measures how often the system produces incorrect results or crashes, and lower values show higher reliability.

Table 1 summarizes the performance metrics and what they measure:

Table 1. Performance Evaluation Metrics

Metric	What It Measures	Meaning of Improvement
Execution Time	How long a task takes to finish	Faster software performance
CPU Usage	Amount of processing power used	More efficient resource usage
Memory Consumption	Amount of RAM used by software	Better memory management
Error Rate	Number of incorrect outputs or system failures	Improved stability and reliability

7. Conclusion

This study has shown that Artificial Intelligence techniques can greatly improve software performance. Traditional optimization methods are often slow and struggle with large or complex systems. However, by using Reinforcement Learning, Deep Learning, and Evolutionary Computation together, the software becomes faster, more efficient, and more reliable.

The optimization results showed clear improvements in execution time, CPU usage, memory consumption, and error rate. These changes mean the system responds faster to users, uses fewer computer resources, and produces fewer errors during operation. The evaluation also confirmed that the optimized system performs better than the original version.

In summary, AI-based optimization gives a smarter and more effective way to improve software performance. The methods used in this study can be applied to many kinds of software systems to meet modern computing demands. Future work can explore more advanced AI techniques and test optimization on real-world industrial applications.

8. Future Suggestions

Key avenues for future research include: 1) Integrating Autonomous Code Optimization Agent; 2) Adopting Predictive Resource Allocation Models; 3) Implementing AI-Driven Testing and Debugging Pipeline; 4) Utilizing Reinforcement Learning for Performance Tuning; 5) Incorporating AI-Enhanced Security Optimization.

Declarations

Source of Funding

This study received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Competing Interests Statement

The authors declare that they have no competing interests related to this work.

Consent for publication

The authors declare that they consented to the publication of this study.

Authors' contributions

All the authors took part in literature review, analysis, and manuscript writing equally.

Availability of data and materials

Supplementary information is available from the authors upon reasonable request.

Institutional Review Board Statement

Not applicable for this study.

References

[1] Alenezi, M., & Akour, M. (2025). AI-driven innovations in software engineering: A review of current practices and future directions. Applied Sciences, 15(3): Article 1344. https://doi.org/10.3390/app15031344.

[2] Arcuri, A., & Yao, X. (2008). A novel co-evolutionary approach to automatic software bug fixing. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2008), Pages 162–168, IEEE. https://doi.org/10.1109/CEC.2008.4630793.

[3] Arya, B. (2025). What are the limitations of traditional optimization techniques? LinkedIn. https://www.linkedin.com/.

[4] Azevedo, B.F., Costa, A.M.A., & Pereira, A.I. (2024). Hybrid approaches to optimization and machine learning in software engineering. Machine Learning, 113(4): 1459–1476. https://doi.org/10.1007/s10994-023-06467-x.

[5] Djamel, R.L., Iheb, N.A., Small, K., & Riyadh, B. (2025). PEARL: Automatic code optimization using deep reinforcement learning (arXiv:2506.01880). arXiv. https://doi.org/10.48550/arXiv.2506.01880.

[6] Gong, J., & Tao, C. (2024). Deep configuration learning for software performance: A systematic study. ACM Transactions on Software Engineering and Methodology, 34(1): 1–62. https://doi.org/10.1145/3702986.

[7] Keylabs. (2024). Optimizing AI models: Strategies and techniques. https://keylabs.ai/blog/optimizing-ai-models-strategies-and-techniques/.

[8] Poposki, P. (2022). Evolutionary optimization algorithms and machine learning for engineering applications. OSF Preprints. https://doi.org/10.31219/osf.io/745yg.

[9] Mienye, I.D., & Theo, G.S. (2024). A comprehensive review of deep learning: Architectures, recent advances, and applications. Information, 15(12): 755. https://doi.org/10.3390/info15120755.

[10] Ahmad, F., & Kamran, I. (2025). A survey of reinforcement learning for optimization in automation. Proceedings of the IEEE 20th International Conference on Automation Science and Engineering (CASE), Pages 2487–2494, IEEE. https://doi.org/10.48550/arXiv.2502.09417.