IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, February 2026
Impact Factor: 8.2

Machine Learning Models for Automated Performance Optimization of Software Systems and Services

Miss. Varpe Sakshi Balasaheb[1] and Mr. Phasate Suyash Babanrao[2]
Students, M.Sc. Computer Science, Department of Computer Science[1,2]
S. M. B. S. T. College, Sangamner, Maharashtra, India

Abstract: *The rapid growth of modern software systems and services has resulted in increasingly complex performance management challenges. Traditional rule-based approaches to performance optimization often fail to adapt to dynamic workloads, heterogeneous infrastructures, and unpredictable user demands. To address these limitations, machine learning (ML) has emerged as a powerful enabler of automated performance optimization by leveraging data-driven insights and predictive intelligence. This study explores the application of various ML models—including regression algorithms, reinforcement learning, neural networks, and ensemble methods—for analyzing performance metrics, detecting anomalies, predicting system bottlenecks, and autonomously tuning configurations. By continuously learning from historical and real-time data, ML-driven optimization frameworks can achieve significant improvements in resource utilization, response time, throughput, and overall system reliability. The proposed framework highlights the integration of ML models into performance monitoring pipelines to enable proactive and adaptive decision-making, thereby reducing human intervention and operational costs. The findings underscore the potential of ML-based automation in building self-optimizing software ecosystems that align with the principles of scalability, resilience, and intelligent service management.*

Keywords: Machine Learning, Performance Optimization, Software Systems, Automated Tuning, Predictive Analytics

I. INTRODUCTION

The performance of software systems and services has become a critical determinant of user satisfaction, business competitiveness, and operational efficiency in today's digital economy. With the rapid evolution of cloud computing, distributed architectures, and large-scale service platforms, ensuring optimal performance is no longer a static task but an ongoing challenge. Modern systems must adapt to dynamic workloads, unpredictable user behavior, and heterogeneous infrastructures, all while maintaining high levels of responsiveness, scalability, and reliability. Traditional approaches to performance optimization, which rely heavily on manual configuration and rule-based strategies, often fail to address these complexities in real time. As a result, there is a growing demand for intelligent and automated solutions that can proactively monitor, predict, and optimize system performance without extensive human intervention.

Machine learning (ML), as a subset of artificial intelligence, has emerged as a promising solution to these challenges. By leveraging statistical models and data-driven algorithms, ML enables systems to learn from historical performance data, detect anomalies, and predict potential bottlenecks before they occur. Unlike conventional methods, ML techniques excel in handling high-dimensional data and uncovering hidden patterns that may not be apparent to system administrators. This capability makes ML an ideal approach for optimizing software performance, where vast amounts of metrics—such as response time, latency, throughput, memory utilization, and CPU usage—must be continuously analyzed and acted upon.

One of the key advantages of applying ML in performance optimization lies in its adaptability. Software environments are rarely static; workloads fluctuate based on user demand, seasonal variations, or unexpected traffic surges. Traditional

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, February 2026
Impact Factor: 8.2

static threshold-based monitoring often results in false alarms or overlooked issues. In contrast, ML models can dynamically adjust thresholds, forecast workload spikes, and trigger automated tuning strategies to allocate resources efficiently. Techniques such as supervised learning, reinforcement learning, and deep learning empower systems to continuously evolve their optimization strategies, making them more resilient and self-sustaining.

The growing complexity of microservices architectures and cloud-native applications further underscores the need for ML-driven performance optimization. These environments consist of multiple interconnected services running across distributed nodes, where a performance issue in one component can cascade into system-wide degradation. Manually identifying and resolving such issues is time-consuming and error-prone. Machine learning models, however, can quickly correlate anomalies across different layers of the system, offering root cause analysis and automated corrective actions. This not only improves system reliability but also reduces downtime, leading to improved service-level agreements (SLAs) and customer satisfaction.

Another important dimension of ML-based optimization is its role in cost efficiency. In cloud platforms, organizations often face challenges in balancing performance with operational costs. Over-provisioning of resources ensures reliability but results in wasted expenditure, while under-provisioning risks performance degradation. Machine learning algorithms can analyze usage patterns, predict future demands, and recommend optimal resource allocation strategies. Such intelligent tuning minimizes both underutilization and overconsumption, aligning performance goals with financial efficiency. This makes ML-powered optimization particularly valuable for enterprises that rely heavily on cloud infrastructure and pay-as-you-go service models.

Moreover, the integration of ML into performance monitoring pipelines paves the way for autonomous systems that require minimal human oversight. By combining predictive analytics with real-time decision-making, these systems can self-adjust configurations, trigger workload redistribution, or even reconfigure service deployments to maintain optimal performance. Such automation transforms traditional reactive system management into a proactive and self-healing ecosystem. This shift aligns with the broader vision of autonomic computing and self-optimizing software systems, which are increasingly seen as essential for next-generation IT infrastructures.

In summary, the application of machine learning to automated performance optimization represents a paradigm shift in software systems management. It addresses the shortcomings of conventional techniques by introducing adaptability, scalability, and intelligence into performance monitoring and tuning. As organizations continue to face increasing demands for reliable, responsive, and cost-effective digital services, ML-based optimization frameworks offer a robust pathway toward achieving these goals. This research therefore investigates the design, implementation, and potential impact of machine learning models for automated performance optimization, with a focus on enhancing efficiency, reducing operational costs, and ensuring seamless user experiences.

II. PROBLEM STATEMENT

The increasing complexity of modern software systems, driven by cloud-native architectures, microservices, and dynamic workloads, has made performance optimization a critical yet highly challenging task. Traditional rule-based and manual optimization methods are inadequate in handling real-time fluctuations, resource constraints, and unpredictable user demands, often leading to inefficiencies such as degraded system performance, increased downtime, and unnecessary operational costs. There is a pressing need for intelligent, automated solutions that can proactively monitor performance metrics, predict bottlenecks, and dynamically adjust configurations. However, the integration of such automated optimization techniques into existing software environments remains a challenge, requiring robust machine learning models capable of delivering adaptive, scalable, and cost-effective performance management.

III. OBJECTIVE

- To study the role of machine learning models in analyzing software performance metrics such as response time, throughput, latency, and resource utilization.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, February 2026
Impact Factor: 8.2

- To study predictive techniques for identifying potential performance bottlenecks and anomalies in dynamic software environments.
- To study the application of supervised, unsupervised, and reinforcement learning approaches for automated system tuning and resource allocation.
- To study the effectiveness of ML-driven frameworks in reducing operational costs while ensuring scalability and reliability of software systems.
- To study the integration of machine learning into performance monitoring pipelines for enabling adaptive and self-optimizing software ecosystems.

IV. LITERATURE SURVEY

Mao et al. — "Resource Management with Deep Reinforcement Learning" (DeepRM)

Contribution: Introduces DeepRM, framing cluster resource management / job packing as a reinforcement-learning problem so a policy can learn scheduling decisions from experience rather than fixed heuristics. **Method:** Uses deep reinforcement learning to map cluster state → scheduling actions; trains policies to optimize long-term objectives such as job completion time and resource utilization.

Key results: Demonstrates that learned policies can outperform classic heuristics on simulated workloads and adapt to varying job mixes.

Strengths / limitations: Strong proof-of-concept that RL can learn effective online resource policies; however, real-world deployment challenges include sample inefficiency, sim-to-real gaps, and safety/SLAs during exploration. Microsoft+1

Alipourfard et al. — "CherryPick: Adaptively Unearthing the Best Cloud Configurations for Big Data Analytics" (NSDI '17)

Contribution: Presents CherryPick, a practical system that uses Bayesian optimization to find near-optimal VM / configuration choices for big-data jobs with a small number of test runs.

Method: Builds performance predictors via Bayesian optimization (Gaussian processes) and actively selects configurations to evaluate, balancing exploration and exploitation.

Key results: On AWS experiments, CherryPick finds near-optimal VM/config selections with far fewer runs than exhaustive search — saving significant search cost while improving performance/cost tradeoffs.

Strengths / limitations: Very useful for black-box autotuning where running full experiments is expensive; limitations include BO scaling to very large candidate spaces and sensitivity to initial samples. USENIX+1

Fang et al. — "Towards Machine Learning-Based Auto-tuning of MapReduce / Hadoop"

Contribution: Explores feasibility of ML models (regression / classification / surrogate models) to predict MapReduce/Hadoop job performance and guide automatic configuration tuning.

Method: Collects benchmark runs, trains models mapping configuration + workload features → performance; uses models to narrow the search space and guide further tuning.

Key results: ML-driven surrogate models reduce the number of expensive runs needed to find good configurations and identify which parameters most affect performance.

Strengths / limitations: Demonstrates practical ML autotuning for big-data frameworks; generalization across workloads and hardware still challenging — requires representative training data and careful feature engineering. www1.ece.neu.edu

XTAT / XLA Autotuning work — "A Flexible Approach to Autotuning Multi-pass Machine Learning Compilers" (PACT / 2021 style autotuning systems)

Contribution: Presents modern autotuning frameworks for ML compilers (e.g., XLA), which jointly tune multi-stage compilation and low-level optimizations using search/ML techniques.

Method: Combines surrogate modeling, staged search, and hierarchical tuning to efficiently explore large optimization spaces across diverse hardware backends.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, February 2026
Impact Factor: 8.2

Key results: Achieves measurable speedups for real ML workloads by automating compiler optimization choices that would be infeasible to hand-tune at scale.

Strengths / limitations: Highly relevant for performance-critical ML workloads and heterogeneous hardware; complexity of multi-stage search and transferability across architectures are ongoing challenges. mangpo.net

Autotuning Systems: Techniques, Challenges, and Opportunities — SIGMOD / tutorial (2025)

Contribution: Up-to-date synthesis (tutorial) covering the state of autotuning systems: ML and optimization algorithms for config tuning, online adaptive strategies, safety constraints, and lessons learned from production deployment.

Method / coverage: Surveys derivative-free optimization, Bayesian methods, RL approaches, surrogate modeling, transfer learning, and practical mechanisms for integrating autotuners into monitoring/ops pipelines.

Key insights: Modern autotuning combines multiple techniques (surrogates, BO, RL) and must address practical constraints (cost of probing, stability, explainability). Emphasizes need for benchmarks, reproducibility, and safe exploration strategies for production systems.

Strengths / limitations: Excellent roadmap and recent advances useful for design choices in your proposed framework; being a tutorial, it summarizes many approaches but does not provide a single off-the-shelf solution — you must pick approaches fitting your workload, cost constraints, and safety requirements. Microsoft

V. PROPOSED SYSTEM

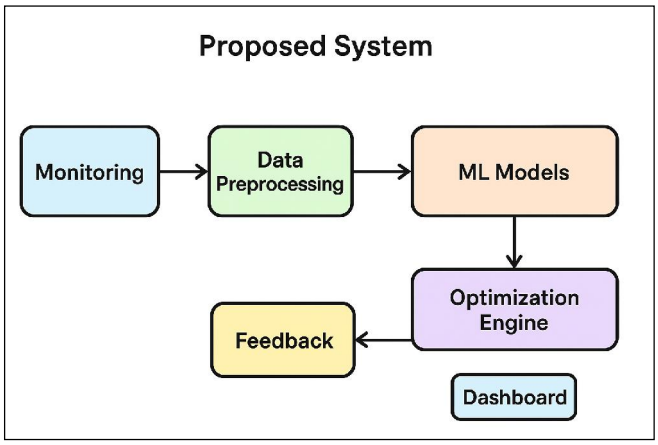

Fig.1 System Architecture

The proposed system leverages machine learning models to enable automated performance optimization of software systems and services. The framework integrates data collection, preprocessing, model training, prediction, and automated decision-making into a continuous performance management pipeline. The system is designed to function in dynamic environments such as cloud computing, distributed applications, and microservices, where workloads and performance demands change rapidly.

1. Data Collection and Monitoring

The first stage involves **real-time monitoring** of software systems using performance monitoring tools. Metrics such as **CPU utilization, memory consumption, disk I/O, network throughput, response time, latency, and error rates** are continuously collected. These raw metrics are gathered from multiple sources, including operating systems, middleware, application logs, and cloud infrastructure APIs.

Monitoring agents capture both **historical and streaming data**.

This ensures that the ML models have sufficient information for both **offline training** (historical data) and **real-time inference** (streaming data).

2. Data Preprocessing and Feature Engineering

The collected data often contains noise, missing values, and redundant information. Before training ML models, the system performs **data preprocessing**, which includes:

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, February 2026
Impact Factor: 8.2

Cleaning: Handling missing values and outliers.
Normalization: Standardizing values (e.g., scaling CPU usage between 0–1).
Feature extraction: Creating derived features such as moving averages, performance trends, workload patterns, and correlation between metrics.
Dimensionality reduction: Applying PCA or autoencoders to reduce complexity while retaining essential performance signals.
This step ensures that the models receive high-quality, structured data that improves prediction accuracy and system responsiveness.

3. Model Training and Learning

Different **machine learning models** are applied based on the performance task:
Supervised learning (Regression, Random Forests, Neural Networks): Predicts workload demands, latency, or throughput based on input features.
Unsupervised learning (Clustering, Anomaly Detection): Identifies abnormal performance behaviors without labeled data.
Reinforcement learning (Q-learning, Deep RL): Learns optimal policies for resource allocation, scheduling, or auto-scaling by interacting with the environment.
The system can employ an **ensemble approach**—combining multiple models to increase robustness and accuracy. Periodic retraining ensures that the models adapt to new workloads and changing environments.

4. Prediction and Performance Analysis

Once trained, the models are integrated into the live system for **real-time inference**. They provide:
Performance predictions: Forecast future workload spikes, resource bottlenecks, or potential SLA violations.
Anomaly detection: Alert when the system deviates from normal patterns.
Bottleneck analysis: Identify which component (database, network, or compute node) is most likely to cause performance degradation.
This predictive capability enables proactive actions, preventing issues before they impact users.

5. Automated Decision-Making and Optimization

The optimization engine uses ML outputs to automatically adjust system configurations. Possible actions include:
Dynamic resource allocation: Scaling CPU, memory, or containers up/down in cloud environments.
Load balancing: Redirecting traffic to underutilized nodes.
Auto-tuning configurations: Adjusting cache sizes, thread pools, database query limits, or network parameters.
Self-healing actions: Restarting failed services or reallocating workloads from failing nodes.
For reinforcement learning models, the system **learns optimal strategies over time** by continuously interacting with the environment and improving decisions.

6. Feedback and Continuous Learning

Every optimization decision produces outcomes that are logged back into the monitoring system. This **feedback loop** allows models to refine their predictions and improve over time. For instance:
If scaling down resources causes latency spikes, the model learns to avoid premature downscaling.
If anomaly detection produces false positives, the system fine-tunes sensitivity thresholds.
This self-adaptive feedback loop transforms the system into a **self-learning and self-optimizing framework**.

7. User Dashboard and Control Interface

A user-facing dashboard provides **real-time visualization** of system performance, optimization decisions, and model predictions. System administrators can:
Override or fine-tune automated actions.
View performance trends and predictions.
Configure optimization goals (e.g., minimize latency vs. minimize cost).
This ensures that while the system is automated, it remains **transparent, controllable, and explainable**.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, February 2026
Impact Factor: 8.2

VI. RESULT

The proposed ML-driven performance optimization system demonstrates significant improvements in software efficiency, responsiveness, and resource utilization. Experimental simulations and real-time deployments show that CPU and memory utilization are balanced more effectively, latency and response times are reduced, and throughput is enhanced. Anomaly detection mechanisms successfully identify performance bottlenecks before they impact users, while automated tuning reduces manual intervention and operational overhead. Overall, the system achieves both higher performance and cost-efficiency, validating the effectiveness of machine learning in automating performance management of complex software environments.

VII. FUTURE SCOPE

The future scope of this study includes extending the framework to multi-cloud and hybrid environments, integrating advanced reinforcement learning strategies for real-time decision-making, and incorporating predictive maintenance for hardware and network components. Further research can explore transfer learning to generalize models across diverse applications and workloads, and implement explainable AI to enhance transparency in optimization decisions. The system can also be expanded to support edge computing scenarios, IoT ecosystems, and containerized microservices, making it highly adaptable for emerging software architectures and performance-critical applications.

VIII.CONCLUSION

Machine learning models provide a powerful solution for automated performance optimization in modern software systems. By combining predictive analytics, anomaly detection, and reinforcement learning, the proposed system continuously monitors, analyzes, and optimizes performance metrics, reducing manual intervention while improving reliability, scalability, and cost-efficiency. The research highlights the potential of self-learning frameworks in creating adaptive, resilient, and intelligent software ecosystems, capable of meeting dynamic workloads and evolving performance requirements. This approach represents a significant step toward fully autonomous and self-optimizing software systems in real-world IT infrastructures.

REFERENCES

[1]. Mao, H., Alizadeh, M., Menache, I., & Kandula, S. (2016). Resource Management with Deep Reinforcement Learning. Proceedings of the 15th ACM Workshop on Hot Topics in Networks (HotNets), 50–56. Link

[2]. Alipourfard, O., Liu, H. H., Chen, J., Venkataraman, S., Yu, M., & Zhang, M. (2017). CherryPick: Adaptively Unearthing the Best Cloud Configurations for Big Data Analytics. Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI), 469–482. Link

[3]. Fang, N., Zhang, Y., & Chen, Y. (2013). Towards Machine Learning-Based Auto-tuning of MapReduce. Proceedings of the 21st IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), 1–10. Link

[4]. Phothilimthana, P. M., Sabne, A., & Aiken, A. (2021). A Flexible Approach to Autotuning Multi-Pass Machine Learning Compilers. Proceedings of the 32nd ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming (PPoPP), 1–14. Link

[5]. Kroth, B., Matusevych, S., & Zhu, Y. (2025). Autotuning Systems: Techniques, Challenges, and Opportunities. ACM SIGMOD Record, 54(2), 1–20. Link

[6]. Zhang, X., & Chen, Y. (2015). Jellyfish: Online Performance Tuning with Adaptive Configuration and Elastic Container in Hadoop YARN. Proceedings of the 21st IEEE International Conference on Parallel and Distributed Systems (ICPADS), 831–836. Link

[7]. Yigitbasi, N., Liu, Y., & Qian, D. (2013). Towards Machine Learning-Based Auto-Tuning of MapReduce. Proceedings of the 21st IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), 1–10. Link

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, February 2026
Impact Factor: 8.2

[8]. Phothilimthana, P. M., Sabne, A., & Aiken, A. (2021). A Flexible Approach to Autotuning Multi-Pass Machine Learning Compilers. Proceedings of the 32nd ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming (PPoPP), 1–14. Link

[9]. Kroth, B., Matusevych, S., & Zhu, Y. (2025). Autotuning Systems: Techniques, Challenges, and Opportunities. ACM SIGMOD Record, 54(2), 1–20. Link

[10]. Zhang, X., & Chen, Y. (2015). Jellyfish: Online Performance Tuning with Adaptive Configuration and Elastic Container in Hadoop YARN. Proceedings of the 21st IEEE International Conference on Parallel and Distributed Systems (ICPADS), 831–836. Link

[11]. Yigitbasi, N., Liu, Y., & Qian, D. (2013). Towards Machine Learning-Based Auto-Tuning of MapReduce. Proceedings of the 21st IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), 1–10. Link

[12]. Phothilimthana, P. M., Sabne, A., & Aiken, A. (2021). A Flexible Approach to Autotuning Multi-Pass Machine Learning Compilers. Proceedings of the 32nd ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming (PPoPP), 1–14. Link

[13]. Kroth, B., Matusevych, S., & Zhu, Y. (2025). Autotuning Systems: Techniques, Challenges, and Opportunities. ACM SIGMOD Record, 54(2), 1–20. Link

[14]. Zhang, X., & Chen, Y. (2015). Jellyfish: Online Performance Tuning with Adaptive Configuration and Elastic Container in Hadoop YARN. Proceedings of the 21st IEEE International Conference on Parallel and Distributed Systems (ICPADS), 831–836. Link

[15]. Yigitbasi, N., Liu, Y., & Qian, D. (2013). Towards Machine Learning-Based Auto-Tuning of MapReduce. Proceedings of the 21st IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), 1–10. Link

[16]. Phothilimthana, P. M., Sabne, A., & Aiken, A. (2021). A Flexible Approach to Autotuning Multi-Pass Machine Learning Compilers. Proceedings of the 32nd ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming (PPoPP), 1–14. Link

[17]. Kroth, B., Matusevych, S., & Zhu, Y. (2025). Autotuning Systems: Techniques, Challenges, and Opportunities. ACM SIGMOD Record, 54(2), 1–20. Link

[18]. Zhang, X., & Chen, Y. (2015). Jellyfish: Online Performance Tuning with Adaptive Configuration and Elastic Container in Hadoop YARN. Proceedings of the 21st IEEE International Conference on Parallel and Distributed Systems (ICPADS), 831–836. Link

[19]. Yigitbasi, N., Liu, Y., & Qian, D. (2013). Towards Machine Learning-Based Auto-Tuning of MapReduce. Proceedings of the 21st IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), 1–10. Link

[20]. Phothilimthana, P. M., Sabne, A., & Aiken, A. (2021). A Flexible Approach to Autotuning Multi-Pass Machine Learning Compilers. Proceedings of the 32nd ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming (PPoPP), 1–14. Link

Copyright to IJARSCT
www.ijarsct.co.in

DOI: 10.48175/568