Performance Evaluation of Web Applications Using JMeter Load Testing for Server Capacity and Response Efficiency

Aulia Rahma Annisa[1]*, Mirza Ardiana [2], Putri Nur Rahayu [3]**, Thomas Brian[4]*, Mohammad Abu Jami'in[5]***
* Jurusan Teknik Kelistrikan Kapal, Politeknik Perkapalan Negeri Surabaya
** Jurusan Teknik Bangunan Kapal, Politeknik Perkapalan Negeri Surabaya
auliaannisa@ppns.ac.id[1], mirzaardiana@ppns.ac.id[2], putri.nur@ppns.ac.id[3], thomasbrian@ppns.ac.id[4], jammy@ppns.ac.id[5]

Article Info

Article history:

Received 2025-11-24
Revised 2026-01-05
Accepted 2026-01-07

Keyword:

Apache JMeter,
Load Testing,
Response Time,
Server Scalability,
Web Performance.

ABSTRACT

The reliability of web-based conference systems is crucial for ensuring smooth services during periods of high activity. This research evaluates the performance of the ICOMTA PPNS journal website by conducting load testing using Apache JMeter with scenarios ranging from 50 to 5000 virtual users, each executed for one hour. The evaluation focuses on response time, error rate, throughput, and bandwidth usage. The results indicate that the website performs reliably with up to 200 concurrent users, demonstrating stable response times and no recorded errors. However, once the load surpasses 300 users, response times increase sharply exceeding 60 seconds and errors begin to appear, suggesting that the server has reached its performance limit. Under the heaviest load of 5000 users, throughput continues to rise, but overall service quality declines significantly. These findings highlight the need for server enhancements or migration to cloud-based infrastructure to ensure stable performance during peak usage.

I. INTRODUCTION

Conference websites such as ICOMTA PPNS hold a critical role in supporting academic workflows, ranging from article submission and reviewer evaluation to publication. System failures during peak periods, such as paper submission deadlines or announcement phases can lead to process delays, user frustration, and potential reputational impacts on the institution. Therefore, maintaining server performance and stability is essential to ensure a seamless user experience and reliable system operation [1].

Load testing is an appropriate method for assessing how much simultaneous traffic a local server can handle [2]. By simulating gradually increasing workloads, load testing enables the analysis of key performance metrics such as response time, error rate, throughput, and bandwidth usage. This method provides a realistic representation of the server's capacity under conditions that reflect typical usage patterns in academic conference systems. Several studies have demonstrated the effectiveness of Apache JMeter for evaluating the performance of academic information systems, highlighting its flexibility and suitability for HTTP-based load generation [3][4]. Comparative analyses further indicate that JMeter remains widely adopted due to its extensibility and consistent performance across diverse testing scenarios [5].

Recent evaluations comparing on-premise and cloud infrastructures show that cloud-based architecture particularly those equipped with load balancing offer superior scalability and faster response times when subjected to heavy workloads [6][7]. These findings are reinforced by both international and national studies demonstrating that cloud environments with load-balancing mechanisms maintain stable performance even during significant traffic spikes, outperforming traditional on-premise deployments in terms of latency, throughput, and resource efficiency [8][9]. Such evidence underscores the importance of assessing whether the local server infrastructure remains adequate for platforms like ICOMTA, particularly during peak academic conference activities.

Focusing on establishing a performance baseline, this study employs load testing exclusively, rather than stress or endurance testing, because the objective is to evaluate realistic operational capacity aligned with typical conference usage patterns. Based on the resulting performance metrics, recommendations for server optimization such as system configuration tuning, caching strategies, or potential migration to cloud infrastructure can be formulated to ensure stable and responsive service continuity.

This study goes beyond a mere technical case study by offering a general framework for evaluating the performance of academic web-based systems using Apache JMeter. The proposed testing methodology, workload modeling, and performance threshold identification can be replicated and adapted for similar journal or conference management platforms. These findings enhance our broader understanding of server capacity limits, patterns of performance degradation, and scalability indicators pertinent to web performance engineering in academic information systems.

II. RESEARCH METHODOLOGY

The data collection process in this research followed a series of systematic stages designed to obtain accurate quantitative and qualitative information regarding the performance of the local and cloud server environments. Three primary methods were employed: infrastructure observation, system documentation analysis, and direct measurement through performance testing tools. Each method was selected to ensure that the evaluation reflects real operational conditions and provides a reliable foundation for comparing both server architectures.

A. Infrastructure Observation

Infrastructure observation was conducted to ascertain the actual operating conditions of the server environments used by the ICOMTA PPNS platform. This assessment covered local server hardware (CPU, RAM, storage, and internal bandwidth), cloud server configurations (VM specifications, vCPU allocation, memory, storage type, and external bandwidth), the network topology linking clients and servers, and the systems' operational status including uptime and recurrent constraints [10].

Performing such a preliminary evaluation is essential to ensure that performance tests are executed in representative and reproducible conditions, to identify pre-existing bottlenecks, and to validate that test results reflect operational realities rather than configuration anomalies. Prior studies emphasize that careful environment inspection and documentation including server specifications and monitoring data improve the accuracy of load-testing outcomes and support correct interpretation of observed performance metrics [11][12].

Moreover, comparative investigations of web infrastructure have shown that differences in underlying hardware, virtualization, and network topology significantly affect measured response times and scalability, reinforcing the necessity of a thorough observational phase before formal load testing [13].

The local server selected as one of the test targets was configured to represent a minimum operational environment typically used for small to medium-scale academic web applications. The machine was provisioned with 1 CPU (dual-core or equivalent), 2 GB of RAM, and 50 GB of storage capacity, running a Linux-based operating system. It was set up with web services using Apache or Nginx and hosted a web-based application capable of processing HTTP requests from client machines. This configuration was chosen to mirror realistic deployment conditions and to ensure that the test environment can be reproduced in comparable on-premise server installations.

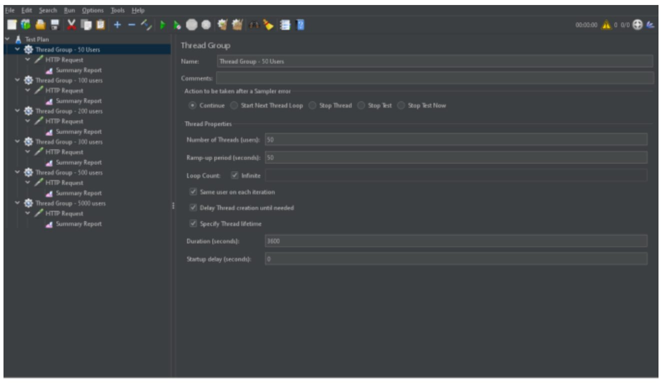
B. System, Documentation and Server Log Analysis

System documentation and server log analysis were performed to collect historical and operational records necessary for a rigorous performance assessment. Collected artifacts included application and web server logs, historical usage statistics for CPU, memory, disk, and network traffic, monitoring dashboards, and incident records documenting journal website downtime. These data sources provide essential context for interpreting load-test results by revealing prior resource utilization patterns, transient faults, and recurring failure modes that might otherwise be misattributed to the test itself; prior studies demonstrate that integrating log analysis and monitoring data substantially improves the validity and diagnostic power of performance evaluations [12].

C. Performance Testing Using Apache JMeter

Primary data collection was conducted through an instrumentation process using Apache JMeter, which served as the main tool for executing the load-testing procedures. This stage aimed to evaluate the system's behavior under varying levels of concurrent access by systematically generating virtual user traffic and recording key performance indicators. The testing process measured several critical parameters, including response time, throughput, error rate, packet loss, data sent and received, as well as CPU and memory utilization during execution [14]. These metrics were collected to enable a comprehensive assessment of the operational capacity of both the local server and the cloud server configurations.

Figure 1 presents the Apache JMeter configuration employed in this study, in which load testing was conducted using a Thread Group whose number of virtual users was specified according to each test scenario. A ramp-up period was configured to gradually introduce users and mitigate abrupt traffic surges, and the thread lifetime option was activated with a fixed test duration of 3,600 seconds to ensure that response time, throughput, and error rate metrics were captured under steady-state conditions. HTTP requests were executed continuously throughout the test window, and performance data were recorded using the Summary Report listener, while each scenario was run with a controlled and repeatable setup consistent with established practices in performance engineering.



Test Plan
- Thread Group - 50 Users
  - HTTP Request
    - Summary Report
- Thread Group - 100 users
  - HTTP Request
    - Summary Report
- Thread Group - 200 users
  - HTTP Request
    - Summary Report
- Thread Group - 300 users
  - HTTP Request
    - Summary Report
- Thread Group - 500 users
  - HTTP Request
    - Summary Report
- Thread Group - 5000 users
  - HTTP Request
    - Summary Report

Thread Group

Name: Thread Group - 50 Users

Comments:

Action to be taken after a Sampler error

[✓ Continue] Start Next Thread Loop Stop Thread Stop Test Stop Test Now

Thread Properties

Number of Threads (users): 50

Ramp-up period (seconds): 50

Loop Count: ✓ Infinite

✓ Same user on each iteration

✓ Delay Thread creation until needed

✓ Specify Thread lifetime

Duration (seconds): 3600

Startup delay (seconds): 0

Figure 1. Apache JMeter Thread Group Configuration for Load Testing

The test plan incorporated sampler configurations, timers, assertions, and listeners to more accurately emulate user behavior and to generate detailed performance logs, aligning with prior work that shows JMeter-based instrumentation can deliver reliable benchmarks of web applications and expose bottlenecks and scalability limitations, particularly when contrasting on-premise and cloud-based deployment environments [15].

The load-testing scenario targeted the homepage of the ICOMTA PPNS website, as this page functions as the main entry point and is the most frequently accessed component during users' initial interaction with the system. Requests to the homepage precede subsequent activities such as browsing articles or submitting manuscripts, thereby serving as a critical indicator of baseline system responsiveness under realistic operating conditions. All test requests were issued using the HTTP GET method, with a controlled think time inserted between successive requests to emulate natural user access patterns and to avoid generating unrealistically aggressive traffic levels.

III. Result and Analysis

Load testing of the ICOMTA PPNS journal website was conducted using Apache JMeter. This load testing aims to evaluate how well the ICOMTA PPNS website can handle simultaneous user access, particularly during critical periods such as paper submission, article uploading, reviewer evaluation, and conference result announcements. The test was carried out using a 1-hour scenario for each user load (virtual users). Six load variations were tested, namely 50, 100, 200, 300, 500, and 5000 users virtual. The performance indicators analyzed include response time (Average, Min, Max), error rate, throughput, and bandwidth consumption (Received and Sent KB/sec).

The test results show variations in website performance as the number of users increases. A detailed analysis is presented in Table 1.

TABLE I
Load Testing Results

User Virtual	Average (ms)	Std. Dev	Error	Throug-hput	Received KB/Sec	Sent KB/Sec
50	452	100.4	0 %	51.0 /hour	0.51	0
100	3307	7157.33	0 %	1.7 /min	1.01	0.01
200	675	870.58	0 %	3.3 /min	2.01	0.02
300	3233	8662.09	1.67 %	5.0 /min	2.96	0.02
500	1610	6630.12	2.40 %	8.3 /min	4.89	0.04
5000	1538	6944.67	2.06 %	1.4 /sec	49.01	0.39

The load scenarios were constructed to represent normal, transitional, and overload operating conditions, derived from the system's observed performance characteristics. As presented in Table I, the system sustains stable behavior with a zero-error rate for up to 200 concurrent users, indicating reliable operation under nominal load conditions. Once the number of simultaneous users surpasses 200, however, signs of performance degradation begin to emerge. At 300 concurrent users, errors are first recorded (1.67%), accompanied by a marked increase in maximum response time, which exceeds 60 seconds. Although throughput continues to grow under heavier loads, this trend is followed by escalating error rates and fluctuating response times, suggesting that server-side resources have reached a saturation point. These patterns indicate that the primary source of degradation lies in bottlenecks affecting server resources such as CPU processing, memory capacity, and backend request handling, rather than solely in network-related constraints. In line with widely cited web performance guidelines, the system's effective operational capacity can therefore be placed at approximately 200 concurrent users.

A. Response Time Analysis

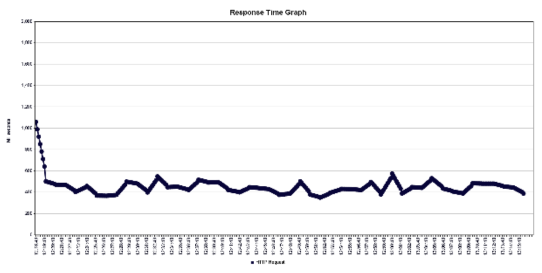
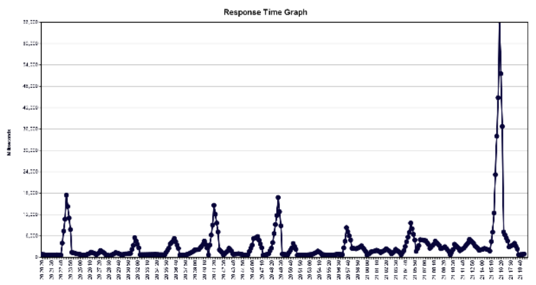
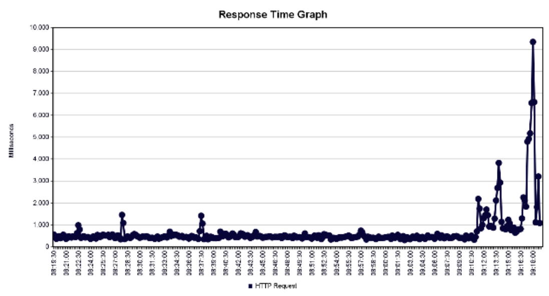
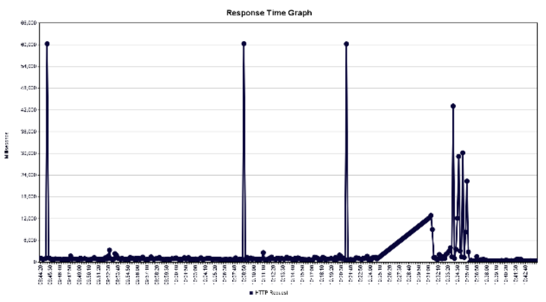
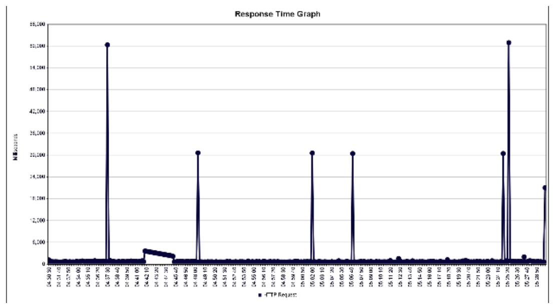
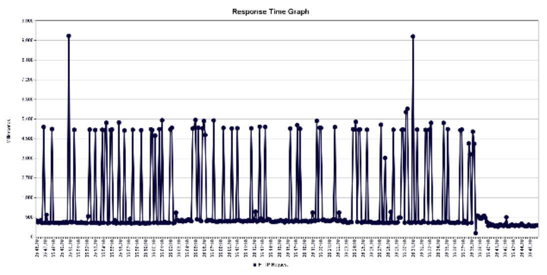
Response time is a key indicator used to evaluate the server's ability to respond to user requests. Based on the test results, the response time shows a fluctuating pattern as the number of users increases. This condition generally occurs because the higher the number of incoming requests, the heavier the processing load on the server, resulting in longer time needed to generate a response. In performance testing, a significant increase in response time may indicate the presence of bottlenecks in the application layer, network configuration, or hardware capacity.

Based on the test results, the response time shows a fluctuating pattern as the number of users increases. In the 50-user scenario, the average response time was 452 ms, indicating that the server was able to provide fast and stable service. However, when the number of users increased to 100, the average response time rose sharply to 3307 ms. This significant increase indicates a decline in service quality due to the growing workload on the server.

Interestingly, in the 200-user scenario, the response time decreased again to 675 ms. This suggests that the server load was not evenly distributed during the test duration, or that there were internal processes causing the load distribution to behave non-linearly relative to the number of users. When the number increased to 300 users, the response time rose dramatically to 3233 ms, and errors began to appear, indicating that the server was entering an unstable state.

Under higher load scenarios, namely 500 and 5000 users, the average response times were 1610 ms and 1538 ms, respectively. Although these averages appear lower than the 300-user scenario, the maximum response time reached more than 60 seconds, showing that some requests experienced extreme delays. Below is the comparison graph of response time for each scenario, as shown in Table I.

TABLE II
RESPONSE TIME COMPARISON

User	Response Time Graph (ms)
50	
100	
200	
300	
500	
5000	

Overall, these results indicate that the server does not maintain stable performance under medium to high load conditions, with an estimated capacity limit of around 200–300 users before performance degradation begins to occur. This condition suggests that the server's resource utilization such as CPU, memory, and I/O operations may be reaching critical thresholds when handling simultaneous requests at higher volumes. As user load approaches or exceeds this limit, response time increases significantly, throughput begins to decline, and error rates may start to appear, indicating that the system is no longer able to process requests efficiently.

This performance behavior is typically associated with limitations in server configuration, hardware capacity, or the performance of backend services supporting the application. Therefore, the identified threshold can serve as a practical reference point for determining when scaling strategies, such as horizontal or vertical server upgrades, load balancing, or application optimization, may be required to maintain reliable service quality under peak operational conditions.

B. *Error Rate Analysis*

The error rate is an important indicator for assessing how consistently the server can handle user requests without failures. In the early test scenarios 50, 100, and 200 users the error rate remained at 0%, indicating that the server was able to process all requests without any issues. However, starting from the 300-user scenario, an error rate of 1.67% appeared, suggesting that the server was beginning to experience overload conditions, causing some requests to fail.

When the load increased to 500 users, the error rate rose further to 2.40%. This indicates that the server had entered an overloaded state, where requests could not be processed in time, often resulting in timeouts due to limited system resources. Under the extreme load of 5000 users, the error rate was still recorded at 2.06%. Although this value did not increase significantly compared to the 500-user scenario, it still shows that the server was unable to consistently handle a large surge of requests.

Overall, these findings confirm that the server's ability to process requests without failures lies below 300 users. Beyond this threshold, failure rates begin to appear, and system stability decreases.

C. *Throughput Analysis*

Throughput reflects the number of requests the server can process within a specific time interval. The throughput results show an increase as the number of users grows, although this increase does not always correlate with better service quality. In the 50-user scenario, throughput was measured at 51 requests per hour, indicating a relatively low but stable user activity level.

From 100 to 500 users, throughput increased from 1.7 to 8.3 requests per minute. This indicates that the server could process more requests as demand grew. However, the increase in throughput was not accompanied by stable overall performance, as both response time and error rate also increased during this period.

In the 5000-user scenario, throughput reached 1.4 requests per second. Although this value appears high, it was achieved at the cost of system stability, as reflected by the high maximum response time and the continued presence of errors. Therefore, high throughput alone cannot be considered a sign of successful performance, since the server exhibited unstable behavior under large-scale load conditions.

D. *Bandwidth Consumption Analysis*

Bandwidth consumption, measured through the Received KB/sec and Sent KB/sec parameters, provides insight into the volume of data transferred between the server and users. In the scenarios with 50 to 500 users, bandwidth usage remained relatively low, with values under 5 KB/sec. This indicates that the data traffic was still within normal limits and did not create significant pressure on the server's network capacity.

However, in the 5000-user scenario, there was a substantial spike in bandwidth consumption, with Received KB/sec reaching 49 KB/sec and Sent KB/sec reaching 0.39 KB/sec. This significant increase suggests that the server was receiving a very large volume of incoming traffic, which likely caused congestion in the network capacity. This congestion contributed to the rising response times and occurrence of errors.

The high bandwidth consumption under extreme load indicates that the local server's network infrastructure has limitations in handling large amounts of data traffic, making it one of the potential bottlenecks affecting overall system performance.

IV. CONCLUSION

The load testing results of the ICOMTA PPNS website using Apache JMeter indicate that the server has not been able to demonstrate stable performance under medium to high load conditions. The system is only able to maintain response times within an acceptable range and process requests without errors at low load levels, namely up to approximately 200–300 concurrent users. Once this threshold is exceeded, there is a significant increase in response time accompanied by request failures, indicating that the capacity of server and network resources has approached or reached their operational limits. Although throughput continues to increase along with the number of users, this phenomenon is not followed by corresponding stability in service performance, so throughput cannot be used as the sole indicator of system reliability. Based on these identified performance limitations, migration to cloud-based infrastructure is recommended as an alternative solution to enhance scalability, performance consistency, and operational efficiency. Nevertheless, further validation through comparative testing between on-premise and cloud environments is required to empirically confirm the effectiveness of this approach and to provide a stronger basis for improving service quality during periods of high traffic.

ACKNOWLEDGEMENT

The authors would like to express their sincere gratitude to Politeknik Perkapalan Negeri Surabaya (PPNS) for providing financial support, research facilities, technical assistance, and other essential resources that enabled the successful completion of this research.

REFERENCES

[1] J. van Riet, I. Malavolta, and T. A. Ghaleb, "Optimize along the way: An industrial case study on web performance," *J. Syst. Softw.*, vol. 198, p. 111593, 2023, doi: https://doi.org/10.1016/j.jss.2022.111593.

[2] M. H. Irsyadi, F. Indra, N. Alam, and A. P. Sari, "Journal of Computer Networks , Architecture and High Performance Computing Application of Google cloud computing for web-based library information systems at Bhayangkara University Surabaya Journal of Computer Networks , Architecture and High Performance Computing," vol. 7, no. 4, pp. 1066–1080, 2025.

[3] I. Indrianto, "Performance Testing On Web Information System Using Apache Jmeter And Blazemeter," *J. Ilm. Ilmu Terap. Univ. Jambi*, vol. 7, no. 2, pp. 138–149, Dec. 2023, doi: 10.22437/jiituj.v7i2.28440.

[4] F. Ramadhan, G. Garno, and A. Solehudin, "Comparative Study of

Web Server Performance Testing with and without Docker Based on Virtual Machines," *J. Appl. Informatics Comput.*, vol. 8, pp. 155–166, Jul. 2024, doi: 10.30871/jaic.v8i1.3884.

[5] E. N. Alam and F. Dewi, "Performance Testing Analysis Of Bandungtanginas Application With Jmeter," *Int. J. Innov. Enterp. Syst.*, vol. 06, no. 02, pp. 146–155, 2022, doi: https://doi.org/10.25124/ijies.v6i02.172.

[6] B. Li, W. Liu, S. Nader, J. Song, C. Zhang, and M. Aibin, "Cloud Load Balancing Algorithms Performance Evaluation Using a Unified Testing Platform," in *2024 International Conference on Computing, Networking and Communications (ICNC)*, 2024, pp. 271–277. doi: 10.1109/ICNC59896.2024.10555949.

[7] M. Zboril and V. Svatá, "Performance comparison of cloud virtual machines," vol. 27, no. 2, pp. 197–213, 2025, doi: 10.1108/JSIT-02-2022-0040.

[8] M. Vieira, "ICPE '23 Companion: Companion of the 2023 ACM/SPEC International Conference on Performance Engineering," New York, NY, USA: Association for Computing Machinery, 2023.

[9] R. F. Saleh, "Comparative Analysis of Public Relations Web Performance of On-Premise and Cloud Computing Infrastructure with Round Robin Load Balancer Analisis Perbandingan Kinerja Web Humas Infrastruktur On-Premise dan Cloud Computing dengan Load Balancer Round Robin," vol. 5, no. July, pp. 1107–1116, 2025.

[10] A. Ismail, A. Y. Ananta, S. N. Arief, E. N. Hamdana, J. T. Informasi, and P. N. Malang, "Performance Testing Sistem Ujian Online," pp. 159–164, 2021.

[11] G. M. Putra, "Pengujian Kinerja Web Server Atas Penyedia Layanan Elastic Cloud Compute (EC2) Pada Amazon Web Services (AWS)," vol. 1, no. 1, pp. 21–35, 2022.

[12] M. Metrik, K. A. Ashari, I. Mardianto, and D. Sugiarto, "Analisa Performa RStudio Server Berbasis Cloud Menggunakan Elastic Stack sebagai Sistem," vol. 7, no. 3, pp. 450–455, 2021.

[13] M. Bolanowski, M. Ćmil, and A. Starzec, "New Model for Defining and Implementing Performance Tests," *Futur. Internet*, vol. 16, no. 10, 2024, doi: 10.3390/fi16100366.

[14] D. I. Permatasari *et al.*, "Pengujian Aplikasi Menggunakan Metode Load Testing dengan Apache Jmeter pada Sistem Informasi Pertanian," vol. 8, no. 1, pp. 135–139, 2020.

[15] H. Setiawan and D. Enda, "Pengujian Load Testing Website Jurusan Teknik Informatika Politeknik Negeri Bengkalis," vol. 5, no. 1, pp. 1–13, 2024.