Load Balancing Optimization through Multi-Label Classification of Network Traffic Using Deep Learning in Distributed Computing Systems

Ferry Fachrizal
Department of Computer Engineering and Informatics, Politeknik Negeri Medan, Medan, Indonesia
ferryfachrizal@polmed.ac.id (corresponding author)

Al-Khowarizmi
Department of Information Technology, Universitas Muhammadiyah Sumatera Utara, Medan, Indonesia
alkhowarizmi@umsu.ac.id

Okvi Nugroho
Department of Information Technology, Universitas Muhammadiyah Sumatera Utara, Medan, Indonesia
okvinugroho@umsu.ac.id

Received: 6 September 2025 | Revised: 19 October 2025, 1 November 2025, and 5 November 2025 | Accepted: 6 November 2025

Licensed under a CC-BY 4.0 license | Copyright (c) by the authors | *DOI: https://doi.org/10.48084/etasr.14567*

ABSTRACT

This study discusses the application of a deep learning-based multi-label classification method integrated with an adaptive load balancing mechanism in a distributed computing system. The main objective of this study is to improve the efficiency, stability, and responsiveness of the system in handling complex and dynamic network traffic. The simulation process begins with traffic data processing, feature extraction using a Recurrent Neural Network (RNN) with a Long Short-Term Memory (LSTM) architecture, and the implementation of a multi-label classification framework. The classification results are then used as input for the adaptive load balancing mechanism, which is further optimized through integration with Deep Reinforcement Learning (DRL). Performance evaluation shows that the multi-label RNN model is able to achieve an Area Under the Curve (AUC) value close to 0.95 on training and validation data, indicating good generalization ability. However, the confusion matrix reveals that there are still quite high classification errors, especially in Video Streaming and Web Traffic, whereas performance on VoIP is relatively more stable. The implementation of DRL is proven effective with a consistently increasing reward trend, indicating the agent's ability to adapt to system conditions. Furthermore, the results of throughput and latency measurements show significant improvements after the implementation of adaptive load balancing. Average throughput increased from 85–115 requests/s to 120–150 requests/s, whereas latency decreased from 95–120 ms to 65–95 ms.

Keywords-multi-label classification; deep learning; adaptive load balancing; Deep Reinforcement Learning (DRL); distributed computing

I. INTRODUCTION

The rapid advancement of digitalization has led to significant developments in information and communication technology, particularly in Industry 4.0. This progress has driven a growing need for computing systems capable of handling large data volumes and complex computing processes [1, 2]. Distributed computing systems have driven major advancements in business models across industries, making traditional centralized computing models increasingly unable to accommodate these needs due to resource, scalability, and flexibility limitations [3, 4]. Therefore, they have emerged as a new paradigm that allows workloads to be distributed across multiple nodes or servers connected via a network. Distributed computing systems enable more efficient resource utilization, improved fault tolerance, and the ability to adapt to dynamic user demand. In this context, one critical issue that must be addressed is how to manage workload distribution, or load balancing, to maintain optimal system performance and prevent bottlenecks on specific nodes [5, 6]. Multi-label classification of network traffic using deep learning algorithms is a computational approach that aims to recognize and group data

packets based on more than one label or category simultaneously. In the context of load balancing optimization in distributed computing systems, this mechanism is important because it can provide a more comprehensive picture of the characteristics of ongoing traffic, so that resource allocation can be carried out adaptively and efficiently. Mechanistically, this process begins with the collection and preprocessing stage of network data, where various parameters such as packet size, throughput, latency, source/destination IP, and protocol type are collected from network logs or network sensors. The data are then normalized and represented in the form of numeric vectors for processing by the deep learning model.

Load balancing is a mechanism for distributing traffic or computing requests evenly among available resources. The goal is to increase system throughput, minimize response times, and avoid overloading specific nodes [7, 8]. However, as the complexity of modern applications increases, network traffic patterns are becoming increasingly heterogeneous and dynamic. Traffic from video streaming applications, social media, cloud services, and Internet of Things (IoT) devices produces different characteristics, including packet size, transmission frequency, and bandwidth requirements [9-11]. This complexity makes traditional static rule-based load balancing mechanisms ineffective [12]. This research uncovers deep issues in load balancing, particularly in distributed computing systems. Several fundamental problems persist that contribute to suboptimal system performance. First, conventional load balancing methods often rely on static or simple heuristic-based approaches, such as round-robin, least connections, or random allocation. These methods fail to account for the heterogeneous nature of modern network traffic [13, 14]. As a result, workloads are not always distributed according to application capacity and needs, which can lead to resource imbalances, increased response times, and reduced Quality of Service (QoS). Second, network traffic classification in the context of load balancing is often performed using a single-label approach, assuming that each data flow has only one dominant category. However, in practice, many data flows have more than one important attribute simultaneously. Third, the increasing scale of data in modern networks poses its own challenges. The volume of traffic generated by millions of IoT devices, cloud services, and real-time applications is very difficult to analyze with traditional machine learning-based methods that still rely on manual feature engineering. Fourth, modern distributed computing systems demand adaptability and real-time decision making. Load balancing mechanisms must not only accurately determine load allocation but also be able to make decisions quickly to avoid disrupting overall system performance. These issues demonstrate the need for a new, more intelligent, adaptive, and accurate approach to load balancing in distributed computing systems [15].

Extensive research has been conducted on load balancing and network traffic classification using both traditional and modern approaches. Many studies integrate machine learning methods into network traffic analysis. Some studies use classical algorithms such as Support Vector Machine (SVM) [16], Decision Tree (DT) [17], and K-Nearest Neighbor (KNN) to classify traffic [18]. For example, a study by authors in [19] using Bayesian neural networks for network traffic classification based on flow-level statistics demonstrated improved accuracy compared to traditional port number-based methods. However, this research was limited to single-label classification. Subsequently, multi-label approaches have been introduced in the context of network traffic. Previous research has highlighted the importance of multi-label learning in the computer networking domain, given the multifunctional nature of many modern applications. Subsequently, deep learning began to be widely used for network traffic classification. Models such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Long Short-Term Memory (LSTM) have proven capable of capturing complex patterns in traffic data [20-22]. For example, authors in [23] used CNNs for encrypted traffic classification and obtained more accurate results than traditional machine learning methods. Thus, it can be concluded that there is a clear research gap. Although there has been research on load balancing and traffic classification with deep learning, the integration of the two in the form of multi-label classification for optimizing load balancing in distributed computing systems is still relatively limited. This gap is the main motivation for this research.

To address the problems outlined, an integrative approach is needed that combines multi-label network traffic classification with deep learning-based load balancing optimization. This solution is expected to provide the system with more accurate traffic classification capabilities. In this research, a deep learning algorithm is the primary approach due to its ability to handle high-dimensional data, automatically extract features, and recognize complex nonlinear patterns [24, 25]. This research uses an RNN algorithm and techniques from the multi-label classification framework, and then integrates them with a load balancing mechanism. The multi-label classification results from deep learning will serve as input for an adaptive load balancing algorithm. One option is to integrate it with Deep Reinforcement Learning (DRL) so the system can learn from interactions with the environment and continuously optimize load distribution.

Based on the above description, it is concluded that optimizing load balancing in distributed computing systems is an urgent need due to the increasing complexity of modern network traffic, which is heterogeneous, dynamic, and often has more than one characteristic simultaneously. Traditional approaches have proven unable to effectively address these challenges, necessitating new, more adaptive, and intelligent solutions. The integration of deep learning-based multi-label classification of network traffic with a load balancing mechanism offers a promising direction, as it can capture complex patterns, minimize load imbalance, and improve overall service quality. Thus, this research is expected to provide not only theoretical contributions in the development of multi-label classification and load balancing methods, but also practical contributions in supporting the performance of more efficient, reliable, and sustainable distributed computing systems in an increasingly connected digital era.

II. LOAD BALANCING PERFORMANCE

Load balancing performance in distributed computing systems can be measured through several mathematical parameters that represent efficiency, distribution fairness, and

quality of service. In general, the workload allocated to each node i with capacity C_i is formulated as:

$$L_i = \sum_j w_j X_{ij} \quad (1)$$

where w_j represents the weight or size of the task j, and X_{ij} is a binary variable that equals one if task j is placed on node i. From here, node utilization is calculated as:

$$u_i = L_i / C_i \quad (2)$$

where C_i is the maximum capacity of the node. Load balance can be assessed through the standard deviation of utilization according to the following equation:

$$\sigma_u = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(u_i - \bar{u})^2} \quad (3)$$

This concept aligns with Jain's Fairness Index (JFI), which is used to assess the level of distributional fairness, with a value range of 0 to 1. As the standard deviation approaches zero, the JFI value approaches one, indicating a highly fair distribution. This is expressed in (4) for JFI and in (5) for the imbalance factor:

$$F = \frac{\left(\sum_{i=1}^{N} u_i\right)^2}{N\sum_{i=1}^{N} u_i^2} \quad (4)$$

$$\beta = \frac{\max_i u_i - \min_i u_i}{\bar{u} + \varepsilon} \quad (5)$$

System performance parameters also include throughput, given by (6):

$$TH = \frac{1}{T}\sum_j 1(t_j^{finish} \le T) \quad (6)$$

The average response time is given in (7):

$$\bar{R} = \frac{1}{|\mathcal{J}|}\sum_j (t_j^{\text{finish}} - t_j^{\text{arrival}}) \quad (7)$$

In addition, the network QoS is assessed based on the average latency:

$$\bar{D} = \frac{1}{|\mathcal{J}|}\sum_j D_j \quad (8)$$

In the context of this research, each traffic flow j is predicted by the deep learning model as a multi-label vector $\hat{Y}_j = [\hat{y}_{j1}, \hat{y}_{j2}, ..., \hat{y}_{jK}]$, where each element $\hat{y}_{jk}$ expresses the probability that task j meets the class-of-service k. The results of this multi-label classification are used as a reference in determining load balancing decisions.

Based on this formulation, the load balancing problem can be viewed as a multi-objective optimization problem with objective functions to minimize the average response time, makespan, utilization deviation, Service Level Agreement (SLA) violation rate, packet loss, energy consumption, and migration overhead, while still considering capacity and QoS constraints. The general form of the weighted objective function can be expressed in (9):

$$\min_x \mathcal{J}(x) = \sum_m \omega_m \frac{P_m}{P_{m,0}} \quad (9)$$

where P_m represents each performance metric, $P_{m,0}$ represents its normalization value, and ω_m represents its preference weight.

This mathematical formulation provides a comprehensive framework for evaluating and optimizing the performance of multi-label classification-based load balancing of network traffic using deep learning.

This study uses an RNN algorithm because network traffic is generally sequential and has temporal correlation. RNNs, particularly the LSTM variant, are capable of modeling long-term dependencies of incoming traffic patterns, resulting in more accurate predictions than feedforward-based models. This multi-label classification process is optimized using the binary cross-entropy loss function shown in (10):

$$L_{BCE}(\theta) = -\frac{1}{|\mathcal{J}|}\sum_j \sum_{k=1}^{K} [y_{jk} \log \hat{y}_{jk} + (1 - \hat{y}_{jk}) \log(1 - \hat{y}_{jk})] \quad (10)$$

with y_{jk} representing the ground truth labels. With this framework, each traffic is not only classified into a single category but can have more than one label; for example, traffic that is both latency-sensitive and requires high bandwidth.

The results of the multi-label classification are then integrated into an adaptive load balancing mechanism. Mathematically, the decision on placing task j on nodes i can be formulated in (11):

$$x_{ij} = \arg\min_{i \in N} \left\{ \eta \hat{D}_j(i) + \eta_2 u_i + \eta_3 \text{risk}_j(i) \right\} \quad (11)$$

with $D_j(i)$ representing an estimate of the latency if the task j is placed on node i, u_i is the utilization of node i, and $\text{risk}_j(i)$ representing the SLA violation penalty based on the classification result labels.

To strengthen the decision-making mechanism, this study considers integration with DRL. In this framework, the multi-label classification results from the RNN serve as the state representation, whereas the traffic allocation decisions on nodes become actions. The reward function is designed based on a combination of performance metrics, such as high throughput, balanced utilization, minimized response time, and low SLA violation rates. Through an iterative process of exploration and exploitation, DRL algorithms such as the Deep Q-Network (DQN) can learn from interactions with the environment to find optimal load balancing policies. In this way, the system is capable of not only performing static, classification-based load balancing but also continuously adapting to changing traffic dynamics and node conditions.

With this formulation, this study provides a complete framework for optimizing load balancing through multi-label classification of network traffic using deep learning. The use of RNN enables sequential pattern recognition of traffic, the

multi-label classification framework provides a comprehensive understanding of traffic characteristics, whereas integration with load balancing mechanisms and DRL ensures adaptive, intelligent, and sustainable load distribution. Thus, this approach is expected to overcome the limitations of traditional methods while improving the efficiency and reliability of modern distributed computing systems.

III. RESEARCH METHODOLOGY

This research methodology is designed to integrate a deep learning-based approach with an adaptive load balancing mechanism in a distributed computing system. The research phase begins with the collection and preprocessing of network traffic data, including temporal parameters, task arrival patterns, and QoS characteristics such as latency, throughput, and bandwidth requirements. The data are then modeled sequentially and processed using an RNN, specifically an LSTM architecture, to predict traffic patterns and generate relevant feature representations. Furthermore, a multi-label classification framework is used to categorize traffic into more than one label, such as latency-sensitive, bandwidth-intensive, and SLA-critical, resulting in more comprehensive information than a single classification. The results of this multi-label classification serve as input for the adaptively designed load balancing mechanism. The research architecture is shown in Figure 1.

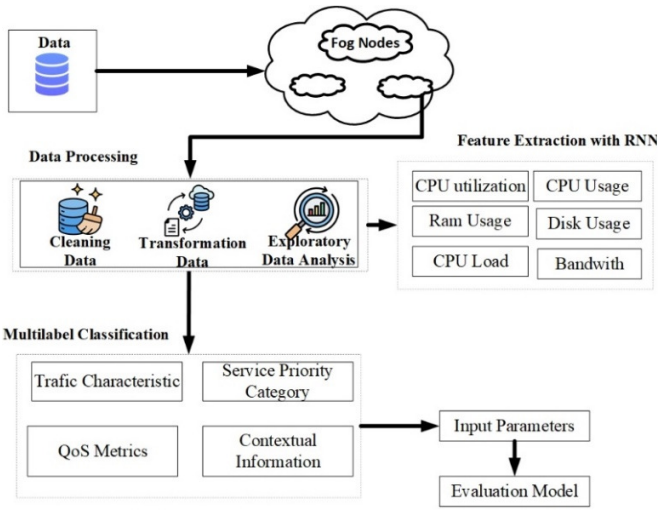

Fig. 1. Research architecture of data processing and load balancing.

Figure 1 illustrates the architecture of the data analysis process, including its utilization in an artificial intelligence-based load balancing mechanism. The steps in this research architecture are as follows:

- The dataset used in this study is sourced from the research in [26], which was originally taken from [27]. The dataset consists of 13 features, each accompanied by an explanation of the parameters that represent important characteristics of a network traffic flow. The Source.IP indicates the originating IP address that initiates communication, whereas the Destination.Port indicates the destination service port. Flow.duration describes the duration of a data flow. Total.Fwd.Packets and Total.Backward.Packets measure the number of packets sent from source to destination and vice versa, respectively, thus providing an overview of the direction of traffic dominance. Flow.Bytes.s indicates the data transfer rate in bytes per second, which helps to understand traffic intensity. Inter-packet time statistical parameters such as Flow.IAT.Mean and Flow.IAT.Std reflect the overall packet delivery delay pattern, whereas Fwd.IAT.Mean and Fwd.IAT.Std describe the packet delay characteristics in the forward direction only. In addition, Fwd.Packets.s and Bwd.Packets.s assess the packet delivery speed in each flow direction. Finally, Average.Packet.Size provides the average size of packets in a flow, which can be used to detect anomalies or certain traffic patterns.
- Feature extraction was then performed using an RNN. RNNs play a crucial role in extracting temporal or sequential information from the data, such as CPU utilization, RAM usage, disk usage, CPU load, and bandwidth.
- After feature extraction, multi-label classification is applied to the extracted features, where each data sample can belong to more than one category. These categories include traffic characteristics, QoS metrics, service priority categories, and contextual information. This multi-label classification process is crucial because network or system conditions often have multiple interrelated attributes. Next, the results of the multi-label classification are used as input for the load balancing mechanism. This mechanism receives the parameters generated from the classification and applies them to the load balancing algorithm to optimize load distribution across the network or computing system. The final stage is model evaluation, which assesses the effectiveness of the load balancing mechanism in improving system performance, both in terms of efficiency, speed, and service stability.

A. Dataset

This research uses the dataset contained in the study by authors in [26], which was originally taken from [27]. This dataset is derived from TCP/UDP traffic and contains 87 features. However, this study only uses 13 parameters for training and testing the model, as shown in Table I.

TABLE I. SELECTED DATASET PARAMETERS

No.	Parameter
1	Source.IP
2	Destination.Port
3	Flow.duration
4	Total.Fwd.Packets
5	Total.Backward.Packets
6	Flow.Bytes.s
7	Flow.IAT.Mean
8	Flow.IAT.Std
9	Fwd.IAT.Mean
10	Fwd.IAT.Std
11	Fwd.Packets.s
12	Bwd.Packets.s
13	Average.Packet.Size

IV. SIMULATION RESULTS

This section presents the results of implementing a deep learning-based multi-label network traffic classification method integrated with a load balancing mechanism in a distributed computing system. Following the design phase of the research methodology, this section aims to demonstrate the performance of the proposed system under experimental conditions that mimic a real-world environment. The simulation process is carried out in stages, starting with traffic data processing, feature extraction with an RNN, and then implementing a multi-label classification framework. The classification results are then used as input for a load balancing mechanism combined with an adaptive approach, even integrating with DRL to optimize load distribution decisions dynamically. A comparison between static and adaptive load distribution is shown in Figure 2.

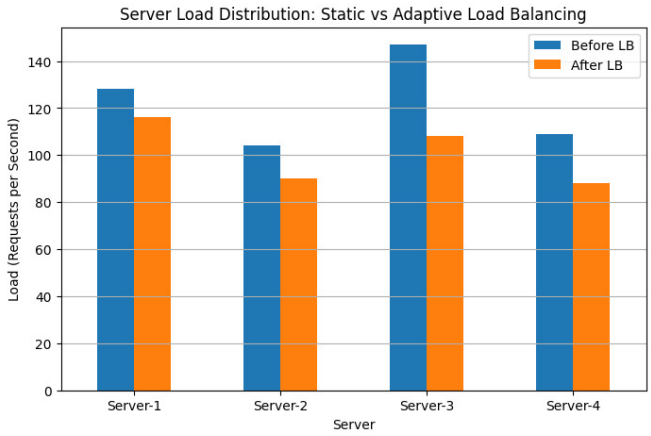

Fig. 2. Server load distribution before and after adaptive load balancing.

The figure illustrates a bar graph comparing the server load distribution before and after load balancing. The X-axis shows the four servers used, Server-1 to Server-4, whereas the Y-axis shows the number of requests received in requests per second. The blue bars represent the server load conditions before load balancing, whereas the orange bars represent the conditions after load balancing is implemented. The graph shows that before load balancing, the load between servers is uneven; for example, Server-3 receives a very high load above 140 requests/s, whereas Server-2 and Server-4 remain below 110 requests/s. After load balancing, the load on each server becomes more balanced, with values converging toward a similar average. This result indicates that the implementation of load balancing, especially the adaptive approach, successfully reduces the load imbalance between servers, resulting in more stable and efficient system performance. After applying load balancing, the training accuracy is evaluated using an RNN with an LSTM architecture.

Figure 3 displays two graphs depicting the results of training a multi-label RNN model based on the loss and Area Under the Curve (AUC) metrics for both the training and validation data. Figure 3(a) shows the comparison between the training loss and validation loss over 20 epochs. It can be seen that the loss values in both datasets tend to decrease as the number of epochs increases, indicating that the model is increasingly able to minimize prediction errors. Initially, the validation loss is higher than the training loss, but both continue to decrease consistently and approach lower values at the end of training, indicating no significant overfitting. Meanwhile, Figure 3(b) illustrates the AUC trends for training and validation. The AUC values in both datasets gradually increase from around 0.5 to nearly 0.95, indicating that the model's ability to distinguish between classes is improving. The similarity in the pattern of increase between the training AUC and validation AUC indicates good generalization, thus it can be concluded that the multi-label RNN was trained effectively with stable performance. After determining accuracy using the AUC metric, the model is further evaluated using confusion matrices.

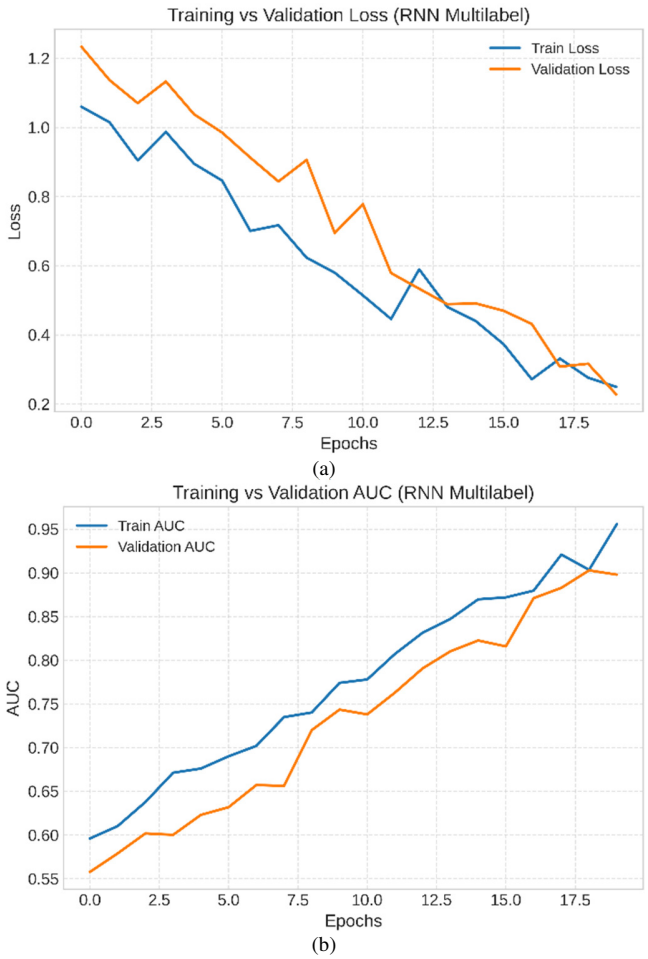

Fig. 3. Training performance of the multi-label RNN model: (a) training and validation loss, (b) training and validation AUC.

Figure 4 displays three confusion matrices used to evaluate the performance of the classification model for three types of network traffic: Video Streaming, VoIP, and Web Traffic. In the Video Streaming confusion matrix, the model was able to correctly classify 23 class-0 data and 21 class-1 data, but errors still occurred, with 29 class-0 data being incorrectly predicted as class-1 and 27 class-1 data being incorrectly predicted as

class-0, indicating a relatively high error rate. In the VoIP confusion matrix, the model successfully recognized 31 class-0 data and 18 class-1 data correctly, but there were 33 class-0 data that were incorrectly predicted as class-1 and 18 class-1 data that were incorrectly predicted as class-0, showing that performance is balanced but the false-positive rate remains high. In the Web Traffic confusion matrix, the model was able to classify 20 class-0 data and 31 class-1 data correctly, but there were still 27 class-0 data that were incorrectly predicted as class-1 and 22 class-1 data that were incorrectly predicted as class-0. After forming the multi-label model, it is applied to load balancing with DRL, and the resulting rewards are shown in Figure 5.

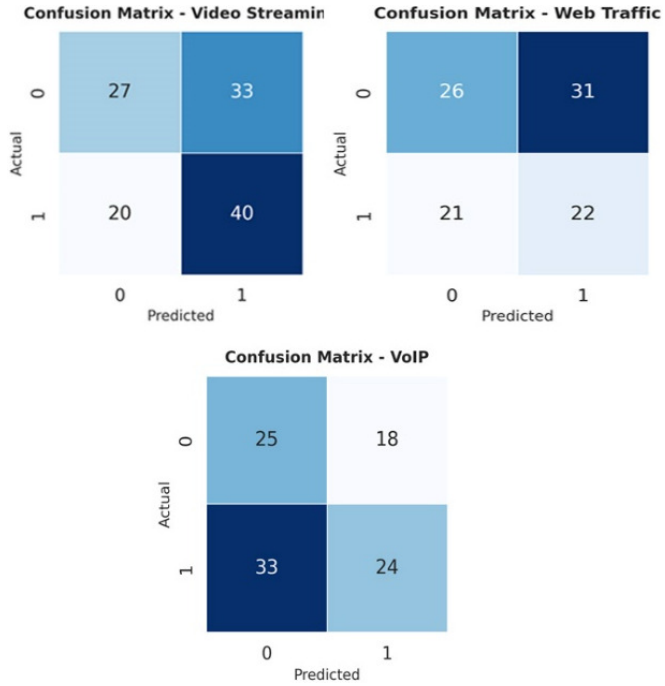

Fig. 4. Confusion matrices for evaluating the multi-label classification model on Video Streaming, VoIP, and Web Traffic.

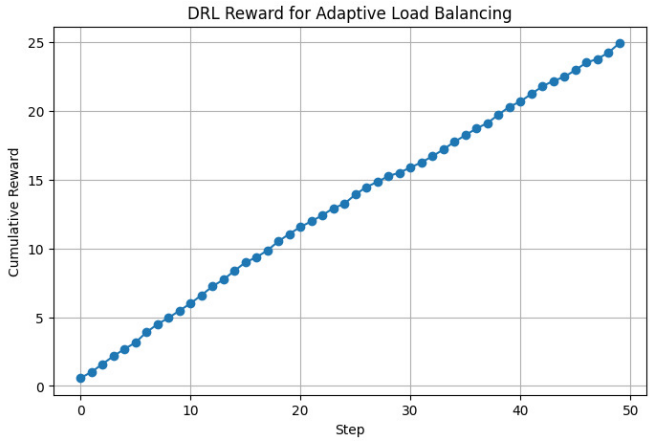

Fig. 5. DRL reward progression for adaptive load balancing.

Figure 5 shows the results of a reward simulation using DRL for adaptive load balancing, where the X-axis represents the number of steps and the Y-axis shows the cumulative reward. The graph shows that the reward increases consistently from around 0.5 at the initial step to nearly 25 at the 50-th step. The increase tends to be linear, for example, at the 10-th step the reward is around 5, at the 20-th step it increases to around 10, then continues to rise to 15 at the 30-th step, 20 at the 40-th step, and approaches 25 at the 50-th step. This steady growth trend indicates that the DRL agent is able to learn and adapt well, making the applied load balancing strategy more optimal as the training steps increase. The absence of sharp drops or large fluctuations in the curve also indicates that the learning process is effective and consistent.

The Receiver Operating Characteristic (ROC) curves for the three traffic types (Video Streaming, VoIP, and Web Traffic) are shown in Figure 6. The X-axis shows the False Positive Rate (FPR) and the Y-axis shows the True Positive Rate (TPR). The dotted diagonal line represents the baseline of the random model with an AUC of 0.5. The results show that Video Streaming achieves an AUC of 0.53, slightly better than the baseline, indicating that the model has limited predictive ability but is still better than random guessing. In contrast, VoIP only achieves an AUC of 0.46, indicating that its performance is slightly below random and therefore cannot predict well. Meanwhile, Web Traffic has the lowest AUC value, 0.37, indicating very poor classification performance because its predictions are more often wrong than correct.

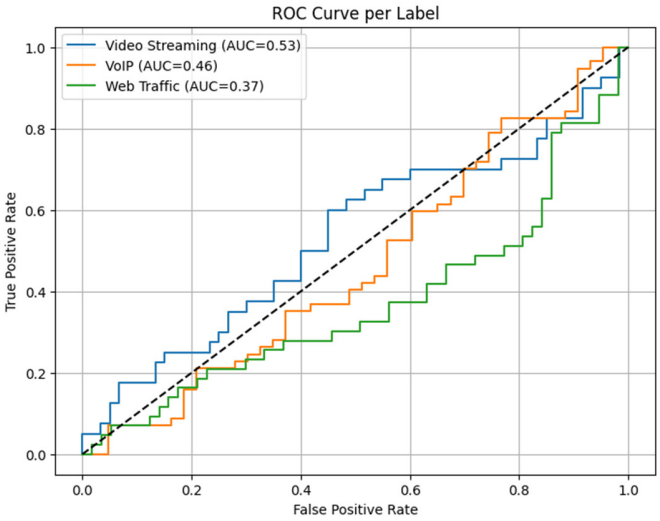

Fig. 6. ROC curves for multi-label network traffic classification.

Figure 7 shows the precision-recall curves for the three traffic categories. The horizontal axis represents recall, which indicates the extent to which the model is able to find true positive data, whereas the vertical axis represents the precision, which describes the accuracy of the positive predictions produced by the model. The graph shows that the VoIP curve (orange line) has better stability with relatively high precision compared to the other two labels, especially when recall increases to near 1. The Video Streaming curve (blue line)

shows fluctuating variations but tends to maintain a precision of around 0.4–0.6 at medium recall. Meanwhile, Web Traffic (green line) has the lowest precision, generally in the range of 0.3–0.4, although at low recall it can reach quite high precision values.

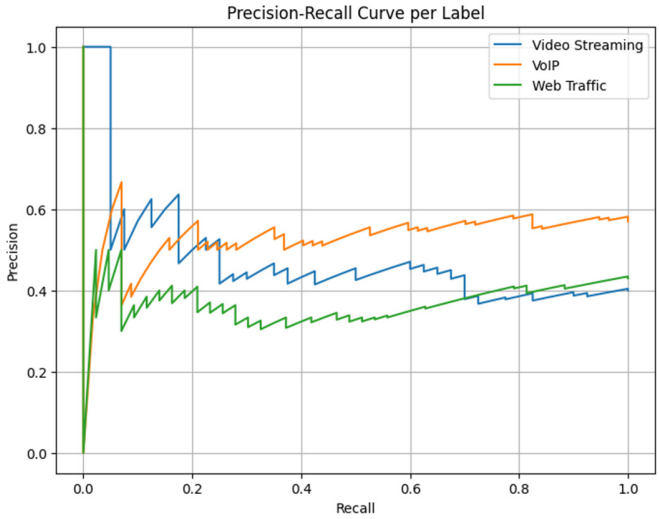

Fig. 7. Precision–recall curves for multi-label network traffic classification.

Figures 8 and 9 show a comparison of the system performance before and after adaptive load balancing using throughput and latency metrics. In Figure 8, the horizontal axis shows the test time in seconds, whereas the vertical axis shows the throughput in requests per second. It can be seen that the system before load balancing had lower throughput, generally in the range of 85–115 requests/s, and exhibited quite sharp fluctuations. Conversely, after adaptive load balancing, throughput increased significantly, averaging above 120 requests/s to nearly 150 requests/s, indicating that this mechanism was able to consistently increase request processing capacity. Figure 9 shows a comparison of latency in milliseconds against time. Before load balancing, latency was relatively high, often in the range of 95–120 ms. However, after adaptive load balancing, latency values dropped dramatically and stabilized at around 65–95 ms. This indicates that the system is more responsive after adaptive load balancing.

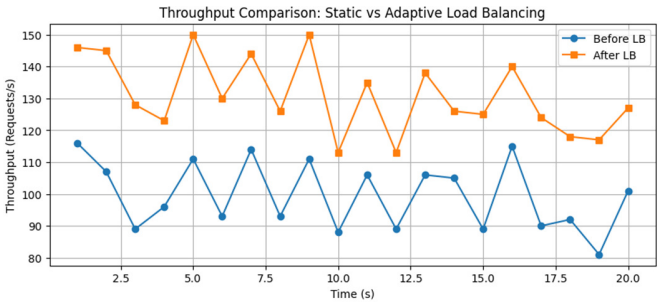

Fig. 8. Throughput before and after adaptive load balancing.

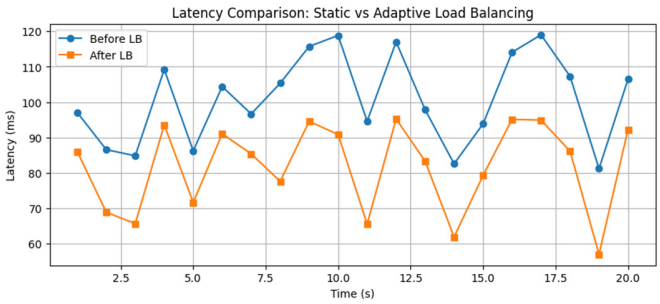

Fig. 9. Latency before and after adaptive load balancing.

V. CONCLUSION

The application of a deep learning-based multi-label classification method integrated with an adaptive load balancing mechanism in a distributed computing system significantly improves system performance. The classification process using a Recurrent Neural Network (RNN) with a Long Short-Term Memory (LSTM) architecture, demonstrates stable training performance without overfitting, achieving an Area Under the Curve (AUC) value approaching 0.95 on both training and validation data. The evaluation using confusion matrices indicates that the model still produces a number of classification errors, particularly for Video Streaming and Web Traffic, although performance for VoIP traffic is relatively better. The integration with Deep Reinforcement Learning (DRL) proved effective, as the reward increased consistently, indicating that the agent can learn and adapt in optimizing load distribution. However, the Receiver Operating Characteristic (ROC) curve results show that classification performance still varies between traffic types, with VoIP and Web Traffic exhibiting lower AUCs than Video Streaming. In terms of network performance, adaptive load balancing successfully balances the load distribution between servers, increasing the average throughput from around 85–115 requests/s to 120–150 requests/s and reducing latency from around 95–120 ms to 65–95 ms. Overall, this approach positively impacts the efficiency, stability, and responsiveness of the system, although there is still room for improvement, especially in the classification accuracy of certain traffic types.

This research introduces significant novelty in both methodology and application in the context of distributed computing systems. Most previous research in the field of network load balancing optimization has focused on heuristic and metaheuristic approaches, such as Genetic Algorithms (GAs), Particle Swarm Optimization (PSO), or Ant Colony Optimization (ACO). In contrast, this study presents a multi-label classification-based approach using deep learning, allowing the system to simultaneously recognize various network performance dimensions, including throughput, latency, jitter, and packet loss. The approach provides enhanced adaptive capabilities because the model is trained using a rich and varied network traffic dataset, thus being able to adjust the load balancing strategy based on actual network conditions.

Future directions for expanding this work could focus on several strategic developments relevant to advances in network technology and artificial intelligence. First, integrating deep

learning models with hybrid optimization algorithms, combining adaptive learning approaches with bio-inspired methods such as DRL with PSO or GAs, could further improve prediction accuracy and system stability under highly dynamic traffic conditions. Second, the study could be expanded to real-time edge and cloud computing environments, testing multi-label classification models against ultra-low latency requirements and large-scale data spread across multiple network nodes. Third, future directions could also include exploring more efficient neural network architectures, such as Graph Neural Networks (GNN) or Transformer-based models, which are capable of understanding topological relationships between nodes and temporal dependencies in network traffic.

REFERENCES

[1] I. C. Obasi and C. Benson, "The Impact of Digitalization and Information and Communication Technology on the Nature and Organization of Work and the Emerging Challenges for Occupational Safety and Health," *International Journal of Environmental Research and Public Health*, vol. 22, no. 3, Mar. 2025, Art. no. 362, https://doi.org/10.3390/ijerph22030362.

[2] A. A. Vărzaru and C. G. Bocean, "Digital Transformation and Innovation: The Influence of Digital Technologies on Turnover from Innovation Activities and Types of Innovation," *Systems*, vol. 12, no. 9, Sept. 2024, Art. no. 359, https://doi.org/10.3390/systems12090359.

[3] A. T. Rosário and R. Raimundo, "Internet of Things and Distributed Computing Systems in Business Models," *Future Internet*, vol. 16, no. 10, Oct. 2024, Art. no. 384, https://doi.org/10.3390/fi16100384.

[4] P. J. Escamilla-Ambrosio, A. Rodríguez-Mota, E. Aguirre-Anaya, R. Acosta-Bermejo, and M. Salinas-Rosales, "Distributing Computing in the Internet of Things: Cloud, Fog and Edge Computing Overview," in *NEO 2016: Results of the Numerical and Evolutionary Optimization Workshop NEO 2016 and the NEO Cities 2016 Workshop held on September 20-24, 2016 in Tlalnepantla, Mexico*, Y. Maldonado, L. Trujillo, O. Schütze, A. Riccardi, and M. Vasile, Eds. Cham, Switzerland: Springer International Publishing, 2018, pp. 87–115, https://doi.org/10.1007/978-3-319-64063-1_4.

[5] A. Barrak, F. Petrillo, and F. Jaafar, "Architecting Peer-to-Peer Serverless Distributed Machine Learning Training for Improved Fault Tolerance." arXiv, Feb. 27, 2023, https://doi.org/10.48550/arXiv.2302.13995.

[6] P. K. Donta, I. Murturi, V. Casamayor Pujol, B. Sedlak, and S. Dustdar, "Exploring the Potential of Distributed Computing Continuum Systems," *Computers*, vol. 12, no. 10, Oct. 2023, Art. no. 198, https://doi.org/10.3390/computers12100198.

[7] B. Pourghebleh and V. Hayyolalam, "A comprehensive and systematic review of the load balancing mechanisms in the Internet of Things," *Cluster Computing*, vol. 23, no. 2, pp. 641–661, June 2020, https://doi.org/10.1007/s10586-019-02950-0.

[8] P. Kumar and R. Kumar, "Issues and Challenges of Load Balancing Techniques in Cloud Computing: A Survey," *ACM Computing Surveys*, vol. 51, no. 6, Feb. 2019, Art. no. 120, https://doi.org/10.1145/3281010.

[9] X. Liu, Y. Han, and Y. Du, "IoT Device Identification Using Directional Packet Length Sequences and 1D-CNN," *Sensors*, vol. 22, no. 21, Nov. 2022, Art. no. 8337, https://doi.org/10.3390/s22218337.

[10] C. Zhao, L. X. Liao, G. Chen, and H.-C. Chao, "Condensation of Data and Knowledge for Network Traffic Classification: Techniques, Applications, and Open Issues," *Sensors*, vol. 25, no. 8, Apr. 2025, Art. no. 2368, https://doi.org/10.3390/s25082368.

[11] Y. Guo *et al.*, "Traffic Management in IoT Backbone Networks Using GNN and MAB with SDN Orchestration," *Sensors*, vol. 23, no. 16, Sept. 2023, Art. no. 7091, https://doi.org/10.3390/s23167091.

[12] M. Amjad Dipa, S. Yakoob, F. Rasid, F. Ahmad, and A. Mahmud, "Deep Reinforcement Learning Based Load Balancing Scheme in Dense Cellular Network Using RoF Technology," *Journal of Communications Software and Systems*, vol. 21, no. 3, pp. 317–326, July 2025, https://doi.org/10.24138/jcomss-2025-0056.

[13] F. Garcia-Carballeira, A. Calderon, and J. Carretero, "Enhancing the power of two choices load balancing algorithm using round robin policy," *Cluster Computing*, vol. 24, no. 2, pp. 611–624, June 2021, https://doi.org/10.1007/s10586-020-03139-6.

[14] D. Choi, K. S. Chung, and J. Shon, "An Improvement on the Weighted Least-Connection Scheduling Algorithm for Load Balancing in Web Cluster Systems," in *2010 International Conferences on Grid and D-tributed Computing (GDC 2010), and Control and Automation (CA 2010)*, Jeju Island, Korea, 2010, pp. 127–134, https://doi.org/10.1007/978-3-642-17625-8_13.

[15] T. Akhtar, N. G. Haider, and S. M. Khan, "A Comparative Study of the Application of Glowworm Swarm Optimization Algorithm with other Nature-Inspired Algorithms in the Network Load Balancing Problem," *Engineering, Technology & Applied Science Research*, vol. 12, no. 4, pp. 8777–8784, Aug. 2022, https://doi.org/10.48084/etasr.4999.

[16] H. M. Abdelghany, "SVM-Based Load Balancing for Efficient Edge Computing," *International Journal of Telecommunications*, vol. 5, no. 1, pp. 1–20, Feb. 2025, https://doi.org/10.21608/ijt.2025.339202.1068.

[17] P. Moutis, S. Skarvelis-Kazakos, and M. Brucoli, "Decision tree aided planning and energy balancing of planned community microgrids," *Applied Energy*, vol. 161, pp. 197–205, Jan. 2016, https://doi.org/10.1016/j.apenergy.2015.10.002.

[18] A. Banerjee, G. Chatterjee, D. Chakraborty, and S. Majumder, "Cluster Based Intelligent Load Balancing Algorithm Applied in Cloud Computing Using KNN," in *2nd International Conference on Non-Conventional Energy: Nanotechnology & Nanomaterials for Energy & Environment*, Rochester, NY, 2019, https://doi.org/10.2139/ssrn.3503518.

[19] S. Izadi, M. Ahmadi, and A. Rajabzadeh, "Network Traffic Classification Using Deep Learning Networks and Bayesian Data Fusion," *Journal of Network and Systems Management*, vol. 30, no. 2, Jan. 2022, Art. no. 25, https://doi.org/10.1007/s10922-021-09639-z.

[20] S. Guo, Y. Lin, S. Li, Z. Chen, and H. Wan, "Deep Spatial–Temporal 3D Convolutional Neural Networks for Traffic Data Forecasting," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 10, pp. 3913–3926, Oct. 2019, https://doi.org/10.1109/TITS.2019.2906365.

[21] N. Ramakrishnan and T. Soni, "Network Traffic Prediction Using Recurrent Neural Networks," in *2018 17th IEEE International Conference on Machine Learning and Applications*, Orlando, FL, USA, 2018, pp. 187–193, https://doi.org/10.1109/ICMLA.2018.00035.

[22] T.-Y. Kim and S.-B. Cho, "Web traffic anomaly detection using C-LSTM neural networks," *Expert Systems with Applications*, vol. 106, pp. 66–76, Sept. 2018, https://doi.org/10.1016/j.eswa.2018.04.004.

[23] S. Soleymanpour, H. Sadr, and M. Nazari Soleimandarabi, "CSCNN: Cost-Sensitive Convolutional Neural Network for Encrypted Traffic Classification," *Neural Processing Letters*, vol. 53, no. 5, pp. 3497–3523, Oct. 2021, https://doi.org/10.1007/s11063-021-10534-6.

[24] M. Li and Z. Wang, "Deep learning for high-dimensional reliability analysis," *Mechanical Systems and Signal Processing*, vol. 139, May 2020, Art. no. 106399, https://doi.org/10.1016/j.ymssp.2019.106399.

[25] T. Georgiou, Y. Liu, W. Chen, and M. Lew, "A survey of traditional and deep learning-based feature descriptors for high dimensional data in computer vision," *International Journal of Multimedia Information Retrieval*, vol. 9, no. 3, pp. 135–170, Sept. 2020, https://doi.org/10.1007/s13735-019-00183-w.

[26] A. Kumar, D. Anand, S. Jha, G. Joshi, and W. Cho, "Optimized Load Balancing Technique for Software Defined Network," *Computers, Materials & Continua*, vol. 72, no. 1, pp. 1409–1426, Feb. 2022, https://doi.org/10.32604/cmc.2022.024970.

[27] J. S. Rojas, "IP Network Traffic Flows Labeled with 75 Apps." Kaggle. [Online]. Available: https://www.kaggle.com/datasets/jsrojas/ip-network-traffic-flows-labeled-with-87-apps.