Congestion Predictive Modelling on Network Dataset Using Ensemble Deep Learning

Purnawansyah[1], Aji Prasetya Wibawa[2,*], Triyanna Widiyaningtyas[3], Haviluddin[4], Roesman Ridwan Raja[5], Herdianti Darwis[6], Andrew Nafalski[7]

[1,2,3]*Department of Electrical Engineering and Informatics, Universitas Negeri Malang, Indonesia*
[1]*Department of Information Systems, Universitas Muslim Indonesia, Indonesia*
[4]*Department of Informatics, Universitas Mulawarman, Indonesia*
[5,6]*Department of Informatics, Universitas Muslim Indonesia, Indonesia*
[7]*UniSA Education Futures, School of Engineering, University of South Australia, Australia*

(Received: August 14, 2024; Revised: September 21, 2024; Accepted: October 02, 2024; Available online: October 15, 2024)

Abstract

Network congestion arises from factors like bandwidth misallocation and increased node density leading to issues such as reduced packet delivery ratios and energy efficiency, increased packet loss and delay, and diminished Quality of Service and Quality of Experience. This study highlights the potential of deep learning and ensemble learning for network congestion analysis, which has been less explored compared to packet-loss based, delay-based, hybrid-based, and machine learning approaches, offering opportunities for advancement through parameter tuning, data labeling, architecture simulation, and activation function experiments, despite challenges posed by the scarcity of labeled data due to the high costs, time, computational resources, and human effort required for labeling. In this paper, we investigate network congestion prediction using deep learning and observe the results individually, as well as analyze ensemble learning outcomes using majority voting, from data that we recorded and clustered using K-Means. We leverage deep learning models including BPNN, CNN, LSTM, and hybrid LSTM-CNN architectures on 12 scenarios formed out of the combination of level datasets, normalization techniques, and number of recommended clusters and the results reveal that ensemble methods, particularly those integrating LSTM and CNN models (LSTM-CNN), consistently outperform individual deep learning models, demonstrating higher accuracy and stability across diverse datasets. Besides that, it is preferably recommended to use the QoS level dataset and the combinations of 3 clusters due to the most consistent evaluation results across different configurations and normalization strategies. The ensemble learning evaluation results show consistently high performance across various metrics, with accuracy, Matthews Correlation Coefficient, and Cohen's Kappa values nearing 100%, indicates excellent predictive capability and agreement. Hamming Loss remains minimal highlighting the low misclassification rates. Notably, this study advances predictive modeling in network management, offering strategies to enhance network efficiency and reliability amidst escalating traffic demands for more sustainable network operations.

Keywords: Network Congestion; K-Means Clustering; Deep Learning; Ensemble Learning; Evaluation

1. Introduction

Congestion in network communication [1], akin to traffic congestion in transportation, arises from various factors such as accidents, maintenance, or unavoidable circumstances. Congestion in network communication commonly stems from bandwidth resource misallocation, where the volume of data packets exceeds available bandwidth capacity. Network congestion can result from increased node density, many-to-one data transmission schemes, and packet collisions, leading to reduced packet delivery ratios and throughput, heightened packet loss and delay, decreased energy efficiency, and/or connection blocking. These issues invariably diminish network Quality of Service (QoS) [2] performance and user Quality of Experience (QoE) [3].

Network control, detection, maintenance, and management are common approaches to addressing congestion across network layers [4]. Within the Open System Interconnection (OSI) model [5], which serves as the primary architecture for internet communication, the transport layer is responsible for end-to-end data delivery, ensuring sequential data transmission with error checking mechanisms, congestion control, and flow control. Generally, congestion control

***Corresponding author: Aji Prasetya Wibawa (aji.prasetya.ft@um.ac.id)**
DOI: https://doi.org/10.47738/jads.v5i4.333

(CC) operates in two scenarios: preventing congestion beforehand, known as open-loop control systems, and addressing congestion after it occurs, termed closed-loop control systems. Both can be achieved through two main mechanisms: control within the Transmission Control Protocol (TCP) [6], [7], and queue management, or Active Queue Management (AQM), at routers. Predictive analysis has traditionally relied on statistical methods, data mining, and conventional neural networks to forecast time series data, but these approaches often face limitations such as rapid convergence and dependency on long-term data and limited to certain cases i.e., traffic prediction [8], [9], [10], and malware detection [11], [12]. Network congestion itself can be examined through packet-loss based analysis; delay-based which uses delay on Round Trip Transmission (RTT) as the primary indicator of congestion; hybrid approaches that combine packet loss and delay as metrics; and machine learning approaches. However, deep learning and ensemble learning for network congestion analysis is still underexplored and holds significant potential for development. This includes experimenting with parameter tuning, data labeling, architecture simulation, and activation functions. A key challenge in network data analysis is the scarcity of labeled data, which is often overshadowed by unlabeled and semi-labeled data due to the high costs, time, computational resources, and human effort required for effective labeling [13].

Deep learning [14] and ensemble learning [15], [16], are learning algorithms in the technology realm that are now widely developed across various big data analyses, including urban big data, healthcare big data, astronomy, and the development of various Internet of Things applications, smart grid, and smart city implementations such as smart home, smart healthcare, smart surveillance, smart transportation, smart agriculture, and smart environment. While deep learning has been utilized in Network Traffic Monitoring and Analysis (NTMA) [17], its quantitative usage in CC analysis remains limited. Moreover, a significant challenge in network data analysis is the scarcity of labeled data, with unlabeled and semi-labeled data being more predominant. Labeling data requires considerable resources, time, computational processes, and human effort. Based on the provided background, this research leverages deep learning models including Backpropagation Neural Network (BPNN) [18], Convolutional Neural Network (CNN) [19], Long Short-Term Memory (LSTM) [20], [21], and combinations of these models i.e., CNN-LSTM [22], [23], and LSTM-CNN architectures to conduct clustering and classification analysis to model a predictive approach to network congestion using deep learning and ensemble learning. Various scenarios are explored, including labeling, parameter tuning, architecture simulation, activation function selection, and ensemble deep learning analysis. The study will focus on network data transmitted via TCP and UDP protocols within the scope of Universitas Muslim Indonesia, a private university in the eastern Indonesia region, using hybrid metrics based on packet loss and delay congestion. Since the data collection is conducted primarily, the research also validates the clustering results to ensure the data used in the analysis process is valid and reliable.

2. Method

2.1. Research Flow

Figure 1 illustrates the comprehensive pipeline followed in this study for network congestion prediction, covering data preparation, normalization, clustering, deep learning classification, and ensemble learning evaluation. Each stage contributes to refining and optimizing the data and models to achieve accurate and reliable predictions. As depicted in figure 1, the flow of this study is a systematic process that begins with Data Preparation including the collection; feature extraction; feature selection forming 2 combinations of level datasets; and categorization of data records and features. The data then undergoes a process of Normalization using Min-Max and MaxAbs Scalers. Following normalization, the data is validated using statistical methods in the Cluster Validation step i.e., the Gap Statistic, Davies-Bouldin Index (DBI), and Elbow method that recommend 3, 4, or 5 clusters to optimize network congestion clustering. The validated data is then segregated into distinct clusters using the K-Means algorithm in the Clustering step.

 ISSN 2723-6471

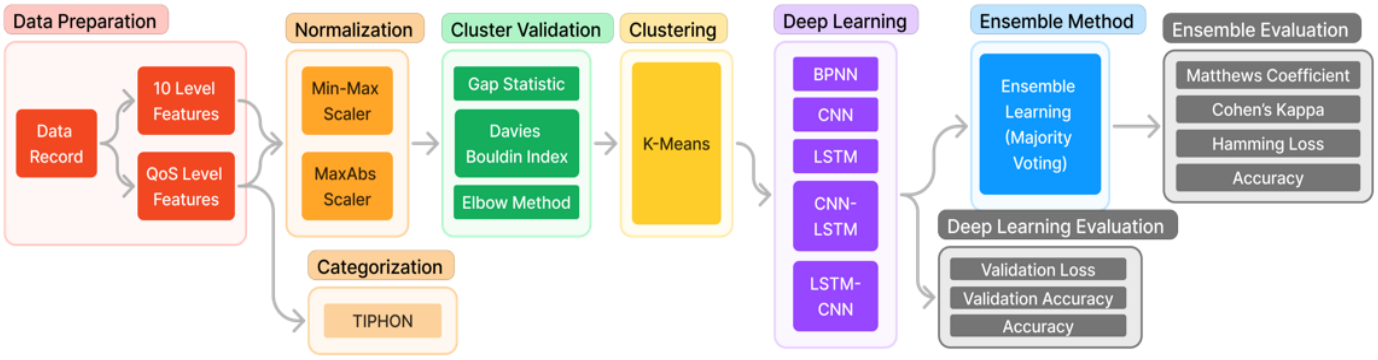


Figure 1. Research Flow Diagram

Furthermore, the clustered data is subjected to Deep Learning Classification using an individual approach and ensemble of neural networks. Lastly, the performance of these models is evaluated in the Evaluation step to assess the effectiveness of the classification. Each of these steps will be explained in detail in the subsequent sections.

2.2. Network Congestion

Network congestion, a state where data transmission decelerates due to bandwidth resource allocation errors, can diminish QoS and QoE. QoS parameters include packet loss, delay, and throughput. Packet loss, indicative of lost data packets due to network congestion, assesses the reliability of packet transmission methods. Delay measures the time for a packet to travel from source to destination. Throughput, the volume of data sent over time from one network point to another, determines the network's reliance on forwarding packets.

Bandwidth, the data transmission capacity over time, is crucial for determining transmission speed and efficiency. As per the QoS parameter assessment standard by the European Telecommunications Standards Institute (ETSI) under the Telecommunications and Internet Protocol Harmonization over Network (TIPHON) [24]. These standards help in maintaining an efficient and reliable network, ensuring that the quality of data transmission remains high despite potential congestion issues.

2.3. Data Preparation

This research employs primary data obtained from the network of the Universitas Muslim Indonesia, collected over a span of 10 weekdays from 06.00 AM to 07.00 PM in two weeks via Wireshark. This period captures a variety of network conditions and usage patterns, offering a representative snapshot of network behavior. However, a potential limitation is the inability to observe long-term trends and rare events, which may affect the generalizability of the results. Extending the data collection period in future research could address these limitations and provide a more comprehensive understanding of network performance over time.

Following data aggregation, the dataset comprises 255148 records, meticulously processed to eliminate noise and address missing values. Ten key variables have been meticulously chosen, comprising the independent variable for each record. These variables include the number of packets (X_{1t}), packet loss (X_{2t}), throughput (X_{3t}), delay (X_{4t}), file size (X_{5t}), data size (X_{6t}), data byte rate (X_{7t}), data bit rate (X_{8t}), average packet size (X_{9t}), and average packet rate (X_{10t}). We selected ten key variables for their critical roles in understanding and predicting network congestion covering essential aspects of network traffic, performance, and reliability. Each variable contributes uniquely to capturing the dynamics of data transmission, enabling a comprehensive analysis that enhances the predictive accuracy and robustness of our models. By addressing various dimensions of network activity, such as volume, efficiency, and latency, these variables ensure a holistic approach to diagnosing and forecasting congestion, ultimately aiding in more effective network management and optimization strategies. In this research, we conduct the same experiment focusing on three QoS variables: packet loss (X_{2t}), throughput (X_{3t}), and delay (X_{4t}). We narrow our investigation to these specific variables due to their critical importance in understanding network performance and congestion dynamics. By

focusing on these key indicators, we aim to gain deeper insights into the factors contributing to network congestion and enhance the predictive accuracy of our model.

2.4. Data Normalization

Normalization is an indispensable step in the preprocessing of data, particularly when dealing with data of high dimensionality. It serves to scale the input attributes, ensuring that no single attribute dominates others due to disparities in their respective scales. This, in turn, augments the performance of the model. In this study, we employ two normalization techniques: Min-Max Scaler and MaxAbs Scaler. Min-Max Scaler is a technique transforms the attributes by scaling each attribute to a specified range, typically between 0 and 1 [25]. The transformation is given in (1).

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (1)$$

X_{norm} denotes the normalized attribute, while X represents the original attribute. X_{min} and X_{max} refer to the minimum and maximum values of the attribute, respectively. This normalization process ensures that all attributes are scaled to a common range, facilitating fair comparison and interpretation across the dataset. MaxAbs Scaler is a technique scales each attribute by its maximum absolute value [26]. It is especially suitable for data that is centered or sparse. The transformation is given in (2).

$$X_{scaled} = \frac{X}{|\ X_{max}\ |} \quad (2)$$

where X_{scaled} represents the scaled attribute, X denotes the original attribute, and $|X_{max}|$ is the maximum absolute value of the attribute. This process ensures that each attribute is scaled proportionally to its maximum absolute value, facilitating accurate comparisons across the dataset.

2.5. Data Categorization

In this study, the process of data categorization follows data normalization. This involves organizing the normalized data into significant categories using TIPHON by ETSI [27]. TIPHON, which stands for Telecommunications and Internet Protocol Harmonization Over Networks, is a standard aimed at harmonizing the delivery of various services over IP networks. It is effective in categorizing network data because it provides a structured approach to classify data based on predefined QoS parameters. Each data record is assigned to a category based on its attributes, with categories defined by patterns and distributions observed in the data. This approach ensures consistency and comparability across different datasets and experiments.

The result is a categorized dataset, with each record associated with a category, ready for subsequent cluster validation and clustering. Utilizing TIPHON for data categorization ensures the effective organization of data into meaningful groups, facilitating pattern identification during clustering. Based on the level of congestion, the data was categorized using the inverse of QoS category in the TIPHON Standard. The TIPHON categorization process involves classifying data into different grades, such as Excellent, Good, Medium, and Low, based on parameters like packet loss and delay. Table 1 provides a detailed description of the TIPHON Standard categories used in this study. Figure 2 further illustrates the functions used to invert the TIPHON classification based on packet loss and delay values.

Table 1. TIPHON Standard

Parameter	Grade	QoS Category
Packet Loss	0%	Excellent
	3%	Good
	15%	Medium
	25%	Low
Delay	<150ms	Excellent

Parameter	Grade	QoS Category
	150ms-300ms	Good
	300ms-450ms	Medium
	>450ms	Low

To implement the inverse categorization, the following functions are used to classify packet loss and delay values into their respective QoS categories.

```
def inverted_tipon_packet_loss(packet_loss):
    if packet_loss == 0:
        return "Low"
    elif packet_loss <= 3:
        return "Medium"
    elif packet_loss <= 15:
        return "Good"
    else:
        return "Excellent"

def inverted_tipon_delay(delay):
    if delay < 150:
        return "Low"
    elif delay <= 300:
        return "Good"
    elif delay <= 300:
        return "Good"
    elif delay <= 450:
        return "Excellent"
```

Figure 2. TIPHON Python Code

2.6. Cluster Validation

Cluster validation is an essential step in the data analysis pipeline, particularly in unsupervised learning tasks such as clustering. It involves assessing the quality of the clusters formed and determining the optimal number of clusters. In this study, we employ three widely-used cluster validation techniques: Gap Statistic [28], DBI [29], [30], and Elbow Method [31]. We use these methods because they offer robust and complementary perspectives on cluster quality, specifically tailored to the context of network congestion analysis. The Gap Statistic assesses the adequacy of clustering solutions by comparing the total intra-cluster variation across different values of (k) (number of clusters) with their expected values under a null reference distribution. Typically, the optimal number of clusters is identified where the gap statistic reaches its maximum.

The DBI serves as a metric for evaluating clustering algorithms. It quantifies the average similarity between each cluster and its most similar counterpart, where similarity is computed as the ratio of within-cluster distances to between-cluster distances. A lower DBI score indicates clusters that are more distinct and less dispersed, reflecting a superior clustering outcome. The Elbow method offers a heuristic approach for determining the optimal number of clusters in a dataset. By plotting the explained variation against the number of clusters, the method identifies the "elbow" of the curve, indicating the point at which additional clusters provide diminishing returns in explaining the data variance, thus suggesting an appropriate cluster count. These validation techniques provide a quantitative measure of the quality of the clusters and help in determining the optimal number of clusters. The outcome of this step is a validated set of clusters ready for further analysis. By employing these robust validation methods, we ensure that our clustering results accurately reflect the complex dynamics of network congestion, thereby enhancing the reliability of subsequent analyses and insights drawn from the data.

2.7. K-Means (Clustering Method)

K-Means is a widely-used method in the field of machine learning for partitioning a dataset into distinct clusters [32], [33]. The algorithm operates on the principle of minimizing the within-cluster sum of squares (WCSS), which is the sum of the squared distances between each data point in a cluster and the centroid of that cluster. In this study, we apply the K-Means algorithm to the categorized and normalized dataset, forming distinct clusters that are then used in the subsequent deep learning classification step. The optimal number of clusters 'k' is determined based on the results

of the cluster validation methods, namely the Gap Statistic, DBI, and Elbow method [34]. These methods provide a quantitative measure of the quality of the clusters and help in determining the optimal number of clusters.

2.8. Training Configuration

The training configuration for our models encompasses the initialization of the model architecture and optimizer, followed by data partitioning into training, validation, and test sets. The models trained include BPNN, CNN, LSTM, CNN-LSTM hybrids, and LSTM-CNN hybrids. The training process is executed over 3000 epochs, where each epoch involves performing a training step, validating the model, and recording validation metrics. Figure 3 provides a detailed illustration of the overall model training and validation process, including the flow of data and performance evaluation across different models.

```
train_models(algorithms=[BPNN, CNN, LSTM, CNN_LSTM, LSTM_CNN]):
        # Initialize model architecture and optimizer
        initialize_model()
        initialize_optimizer()

        # Split data into training, validation, and test sets
        split_data()

        # Training loop for 3000 epochs
        for epoch in range(3000):
            perform_training_epoch()
            validate_model()
            save_validation_metrics()

        # Evaluate model with test accuracy
        evaluate_model()
```

Figure 3. Model Training & Validation Python Code

This comprehensive training regimen ensures the iterative refinement of model parameters, aimed at optimizing performance. Upon completion of the training epochs, the models are evaluated based on test accuracy to determine their effectiveness and generalizability.

2.9. Deep Learning Classification

Following the clustering process, the next step is deep learning classification. In this study, we employ several deep learning models using PyTorch Framework, including BPNN, CNN, LSTM, and combinations of these models i.e., CNN-LSTM, and LSTM-CNN. We chose BPNN for its simplicity and effectiveness in handling non-linear relationships, CNN for its ability to capture spatial dependencies and patterns in the data, and LSTM for its strength in modeling temporal dependencies and sequence prediction. The hybrid models, CNN-LSTM and LSTM-CNN, were selected to leverage the advantages of both spatial and temporal feature extraction, providing a more comprehensive approach to network congestion prediction.

BPNN is a type of artificial neural network that uses a supervised learning method for training. It is known for its efficiency in solving complex nonlinear problems. The architecture of the BPNN model used in this study is shown in Table 2. The learning rate for the BPNN model is set to the default value of 0.001 in PyTorch. Table 2 presents the structure of the model, which includes two hidden layers (FC1 and FC2), each followed by a Rectified Linear Unit (ReLU) activation function. The output layer (FC3) functions without an activation function, implying a classification task.

Table 2. BPNN Architecture

Layer (Fully Connected)	Input Dimension	Output Dimension	Activation Function
FC1	10 & 3 (QoS)	5	ReLU
FC2	5	5	ReLU
FC3	5	Cluster Size	None

CNNs are a type of deep learning neural networks, predominantly used for visual imagery analysis [35]. They are utilized in image and video recognition, recommendation systems, and natural language processing. The design of the CNN model employed in this research is depicted in table 3. The learning rate for the CNN model is set to the default value of 0.001 in PyTorch.

Table 3 displays the model architecture, comprising a convolutional layer (conv1) followed by a max pooling layer (pool), as well as two fully connected layers (FC1 and FC2). The Rectified Linear Unit (ReLU) activation function is applied after the convolutional layer and the initial fully connected layer. The output layer (FC2) does not utilize an activation function, assuming a classification task.

Table 3. CNN Architecture

Layer Type	Layer Name	Input Channels	Output Channels	Kernel Size	Stride	Padding	Activation Function
Conv2d	conv1	10 & 3 (QoS)	32	1	1	0	ReLU
MaxPool2d	pool	-	-	1	1	-	-
Linear	fc1	32	128	-	-	-	ReLU
Linear	fc2	128	Cluster Size	-	-	-	None

LSTM [28] networks represent a variant of recurrent neural networks specifically designed to capture order dependence in sequence prediction tasks. This characteristic proves advantageous in time series forecasting, particularly for multivariate or multiple input scenarios, where traditional linear techniques may struggle to accommodate complexities. Refer to Table 4 for a depiction of the LSTM model architecture utilized in this research. The learning rate for the LSTM model is set to the default value of 0.001 in PyTorch. Table 4 displays the model's architecture, which includes an LSTM layer succeeded by a fully connected layer (fc). The LSTM layer internally uses a mix of hyperbolic tangent (tanh) and sigmoid activation functions. The output layer (fc) operates without an activation function, indicating a classification task.

Table 4. LSTM Architecture

Layer Type	Layer Name	Input Dimension	Output Dimension	Activation Function
LSTM	lstm	10 & 3 (QoS)	5	Tanh/Sigmoid
Linear	fc	5	Cluster Size	None

CNN-LSTM and LSTM-CNN: These are hybrid models of CNN and LSTM, engineered to harness the advantages of both models. The CNN-LSTM model capitalizes on the CNN's proficiency in handling spatial data along with LSTM's capacity for temporal data processing. On the other hand, the LSTM-CNN model employs LSTM for temporal data processing and CNN for spatial data processing. Table 5 illustrates the CNN-LSTM model architecture, comprising a convolutional layer (conv1), subsequent max pooling layer (pool), a fully connected layer (FC1), an LSTM layer, and another fully connected layer (FC2). ReLU activation function is employed following the convolutional and first fully connected layers. The LSTM layer utilizes a blend of hyperbolic tangent (tanh) and sigmoid activation functions internally [36]. The output layer (FC2) operates without an activation function, presuming a classification task.

Table 5. CNN-LSTM Architecture

Layer Type	Layer Name	Input Dimension	Output Dimension	Activation Function
Conv2d	conv1	10 & 3 (QoS)	32	ReLU
MaxPool2d	pool	-	-	-
Linear	fc1	32	128	ReLU
LSTM	lstm	128	hidden_size	Tanh/Sigmoid
Linear	fc2	hidden_size	Cluster Size	None

Table 6 outlines the engineering of the LSTM-CNN show, which incorporates an LSTM layer taken after by a convolutional layer (conv1), a max pooling layer (pool), and two completely associated layers (FC1 and FC2). The Amended Straight Unit (ReLU) actuation work is utilized after the convolutional layer and the starting completely associated layer. Inside the LSTM layer, a combination of hyperbolic digression (tanh) and sigmoid enactment

capacities is utilized. The yield layer (FC2) works without an enactment work, beneath the presumption of a classification errand.

Table 6. LSTM-CNN Architecture

Layer Type	Layer Name	Input Dimension	Output Dimension	Activation Function
LSTM	lstm	10 & 3 (QoS)	32	Tanh/Sigmoid
Conv1d	conv1	32	64	ReLU
MaxPool1d	pool	-	-	-
Linear	fc1	64	128	ReLU
Linear	fc2	128	Cluster Size	None

2.10. Ensemble Learning

Ensemble learning is a powerful machine learning paradigm where multiple models are trained to solve the same problem and combined to get better results. In this study, we use an ensemble method known as Majority Voting [37]. In majority voting [38], each model in the ensemble votes for a class label, and the class label that gets the majority of votes is predicted as the final output. If the ensemble consists of models of varying performance, the majority voting method can lead to improved predictive accuracy over individual models. The principle behind majority voting is that it leverages the wisdom of the crowd. While individual models may have weaknesses, when many models are combined, their strengths can collectively outweigh their weaknesses, leading to a more robust and accurate prediction. The ensemble of models in this study includes the BPNN, CNN, LSTM, and combinations of these models such as CNN-LSTM and LSTM-CNN. Majority voting was chosen for its simplicity and effectiveness. It combines predictions from multiple models, enhancing overall accuracy and robustness by reducing individual model variance. This method is particularly suitable for network congestion prediction, where diverse models capture different data patterns, leading to more reliable results.

2.11. Evaluation Metric

In our study, we focus on utilizing validation loss, validation accuracy, and test accuracy as the primary evaluation metrics for deep learning models. By consistently monitoring validation loss and accuracy throughout training, we offer transparency regarding the model's convergence and generalization capabilities. Additionally, test accuracy serves as the ultimate measure of the model's ability to generalize to new, unseen data. For ensemble evaluation, we expand the assessment to include metrics like Matthews Correlation Coefficient (MCC) [39], Cohen's Kappa [40], and Hamming Loss [41]. Validation loss is used to measure the model's ability to generalize beyond the training data, ensuring it is not overfitting. Accuracy provides a straightforward measure of the overall correctness of the model's predictions. MCC is chosen for its ability to handle imbalanced datasets and provide a balanced measure that considers true and false positives and negatives. Cohen's Kappa is included to measure the agreement between predicted and actual labels, adjusting for the possibility of random chance. Hamming Loss is used to quantify the fraction of incorrect labels, which is particularly useful in multi-class and multi-label classification problems. These metrics offer nuanced evaluations, particularly beneficial for ensemble approaches and datasets with complex class distributions, and provide a comprehensive understanding of the ensemble's predictive performance and its robustness in predicting network congestion across diverse scenarios.

3. Results and Discussion

The results are derived from the various stages of the research flow, including data preparation, normalization, categorization, cluster validation, clustering, deep learning classification, and ensemble learning. Each of these stages contributes to the final outcome, providing insights into the effectiveness of the proposed methodology. Utilizing Wireshark and the Packet Capture (PCAP) API, network traffic was captured and stored in a PCAP file. As depicted in figure 4, a Python script was employed to transform these files into Excel, yielding key metrics such as packet number, file size, data size, capture duration, data byte rate, data bit rate, average packet size, average packet rate,

 ISSN 2723-6471

throughput, packet loss, delays, packet loss percentage, and delay in milliseconds. The total recorded data amounted to 255,148 records.

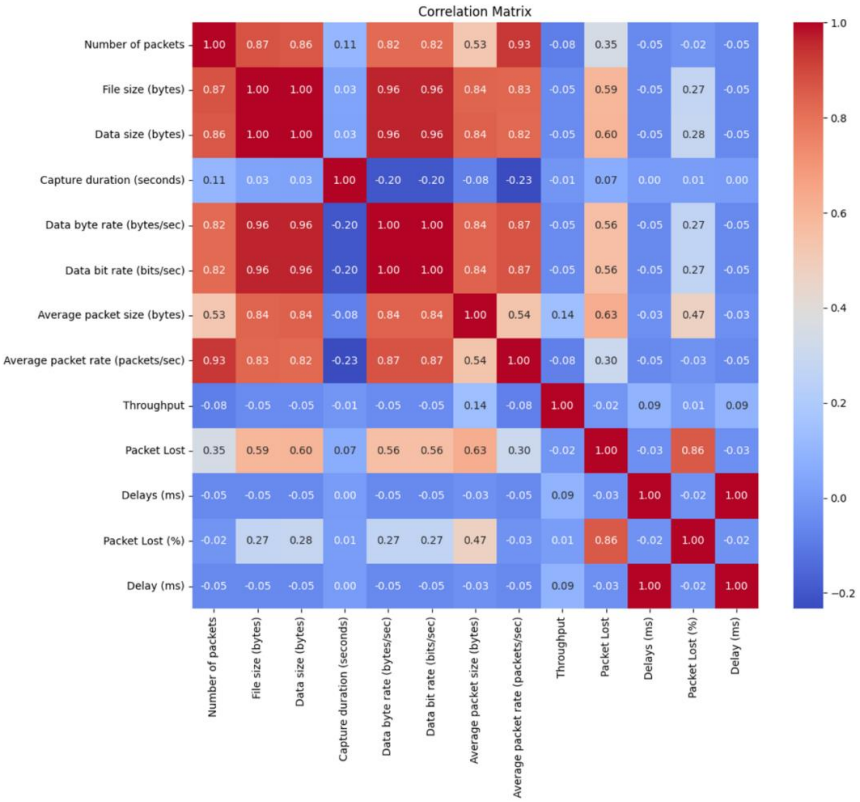


Figure 4. Correlation Matrix of Features

In the pursuit of a comprehensive understanding of the network parameters, we have undertaken an extensive data collection and analysis process. These parameters, which include metrics such as the number of packets and delays (measured in milliseconds), provide critical insights into the operational efficiency of network systems. To encapsulate these elements in a cohesive manner, we introduce a correlation matrix as in Figure 4 that outlines the interrelationships between various network parameters. Each cell within this matrix is color-coded, representing correlation coefficient values that range from -1 to 1. This visual representation offers an intuitive understanding of the interactions between each parameter and the intensity of their correlations. In the ensuing sections, we will explore these interactions in greater depth, discussing their implications for network performance and potential optimization strategies. The insights derived from this matrix will serve as a foundation for targeted interventions aimed at enhancing network system efficiency and reliability. Following this, we present a statistical summary of these network parameters in the form of a table. Table 7 presents a comprehensive statistical summary of the network parameters used in this research. Each row represents a different statistical measure (count, mean, standard deviation, minimum, 25th percentile, median, 75th percentile, maximum) for each of the network parameters. This tabular representation allows for a more granular understanding of the data set, complementing the insights gleaned from the Correlation Matrix (figure 2). It provides a tangible representation of the key metrics, thereby facilitating a deeper understanding of the network's operational efficacy. The subsequent sections will delve into the implications of these statistics for network performance and the potential strategies for optimization.

Table 7. Descriptive Dataset

	Mean	Standard deviation	Min	Max
Number of packets	1.26E+04	4.16E+03	11	5.04E+04
File size (bytes)	9.69E+06	4.59E+06	1.54E+03	4.49E+07

Data size (bytes)	9.27E+06	4.47E+06	840	4.33E+07
Capture duration (seconds)	0.065277778	0.11	0.001	1
Data byte rate (bytes/sec)	9.92E+06	4.84E+06	921	4.33E+07
Data bit rate (bits/sec)	7.94E+07	3.87E+07	7.37E+03	3.46E+08
Throughput	0.07	0.53	0.01	114
Packet Loss	1.39E+03	1.23E+03	0	1.23E+04
Packet Loss (%)	11.08	9.17	0	100
Delay (ms)	179	3.30E+03	0	6.62E+05

3.1. Cluster Validation

Table 8 elucidates the outcomes of cluster validation, employing diverse methods and metrics. Each row delineates a distinct method (Min-Max 10 Level, Min-Max 3 QoS Level, MaxAbs 10 Level, MaxAbs 3 QoS Level), while each column signifies a different metric (DBI for $k = 3$, $k = 4$, and $k = 5$, Gap Optimal, Elbow Optimal).

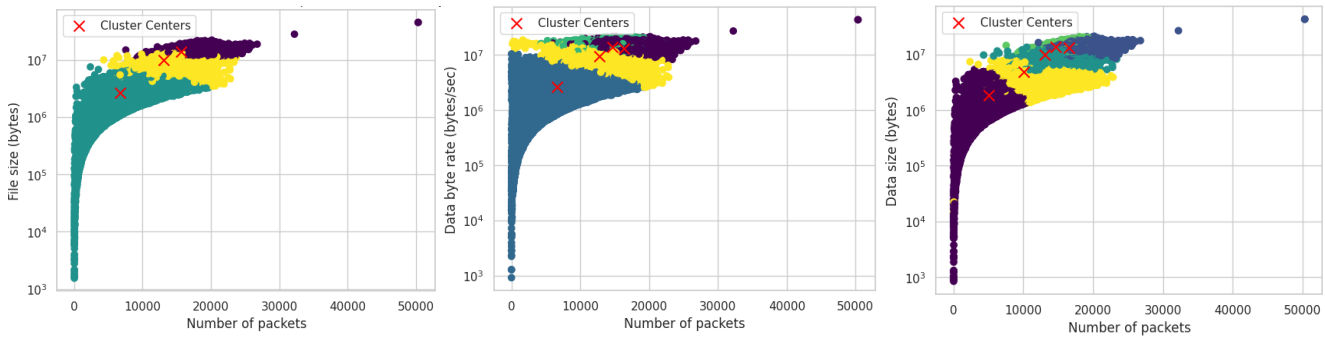
Table 8. DBI Cluster Validation

	DBI $k = 3$	**DBI k=4**	**DBI k=5**	**Gap Optimal**	**Elbow Optimal**
Min-Max 10 Level	0.47	0.51	0.54	5	3
Min-Max 3 QoS Level	0.45	0.49	0.39	5	4
MaxAbs 10 Level	0.47	0.51	0.54	5	3
MaxAbs 3 QoS Level	0.45	0.49	0.39	4	4

The DBI method indicates that the optimal cluster number for a 10-level dataset is 3 clusters, and for the QoS level, it is 5 clusters, aligning with the elbow method's results. However, the statistical gap method presents different results, suggesting that the optimal cluster number for 10 levels is 5 clusters. Interestingly, a discrepancy is observed between min-max and maxAbs normalization at the QoS level. The Min-Max QoS Level exhibits optimal clusters at 5 clusters, whereas the MaxAbs QoS Level presents an optimal cluster of 4 clusters. This highlights the nuanced differences that can emerge depending on the normalization method employed.

3.2. K-Means Analysis

After conducting several analyzes on 6 cluster combinations (3, 4, and 5 clusters against Normalized Min-Max & MaxAbs, at 10 Level features dataset), all combinations of the 10 Level dataset have fair data distribution and suitability for the clusters center when comparing "Number of Packet" with "File Size", "Data Size", "Data byte rate" as shown in the figure 5, which represents the above statement and all 6 cluster combinations.

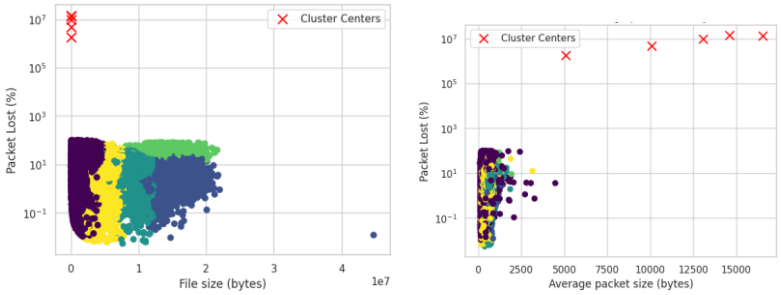


(a) (b) (c)

Figure 5. 10 Level Features K-Means Clustering: Packet vs (a) File Size (3 Clusters, Min-Max), (b) Data Byte Rate (4 Clusters, MaxAbs), (c) Data Size (5 Clusters, Min-Max)

 ISSN 2723-6471

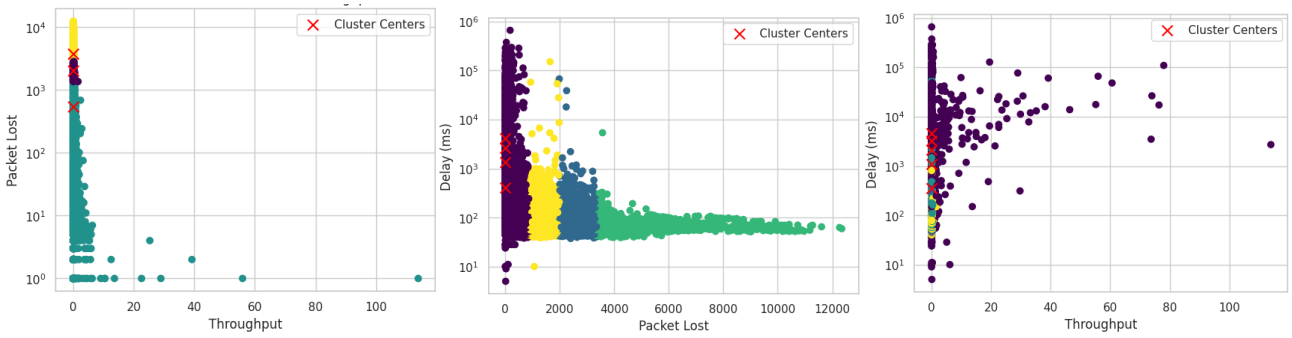
In the case of other feature combinations, there exists a data distribution that, while still exhibiting a pattern, does not align with the central cluster. Additionally, there is a data distribution that lacks any discernible pattern. These distributions are depicted in figure 6.



(a) (b)

Figure 6. 10 Level Features K-Means Clustering: (a) File Size vs Packet Loss (5 Clusters, Min-Max), (b) Avg Packet Size vs Packet Loss (4 Clusters, MaxAbs)

In the other six combinations (3, 4, and 5 clusters for Normalized Min-Max & MaxAbs, on 3 QoS Level datasets), it was observed that all combinations of QoS Level datasets exhibit a patterned data distribution and align with the central cluster only when packet loss is juxtaposed with throughput. However, there is a patterned data distribution that does not coincide with the central cluster when comparing packet loss with delay. Furthermore, there is an absence of any discernible pattern when examining the combination of Delay and Throughput features. These observations are visualized in figure 7 representing all 6 QoS Level combinations. Based on the aforementioned observations, it can be inferred that the clustering outcomes for the 10 Level dataset, for both Min-Max and MaxAbs and across all cluster combinations, indicate that K-Means attempts to cluster the dataset based on the Number of Packets and features associated with packet size. Conversely, the clustering outcomes for the 3 Level QoS dataset, for both Min-Max and MaxAbs and across all cluster combinations, reveal that K-Means aims to cluster the data based on Packet Loss.

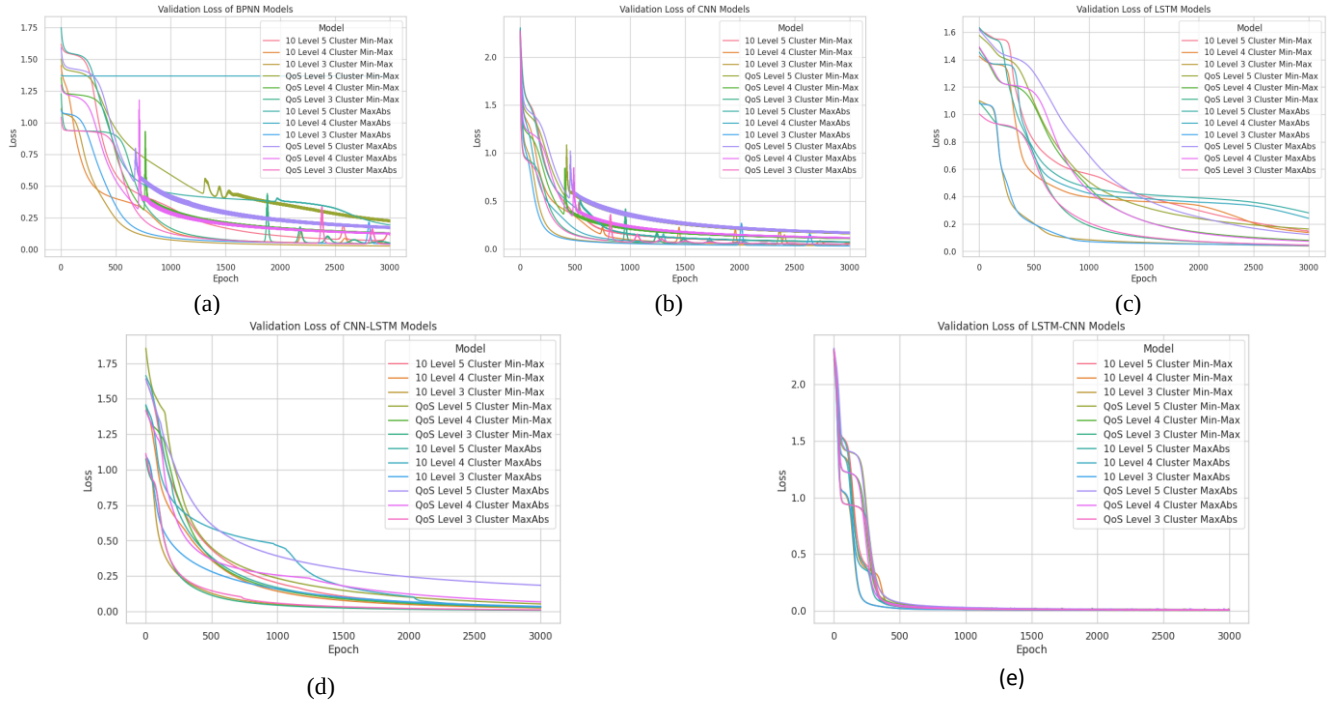


(a) (b) (c)

Figure 7. 3 QoS Level Features K-Means Clustering: (a) Throughput vs Packet (3 Clusters, Min-Max), (b) Packet Loss vs Delay (4 Clusters, MaxAbs), (c) Throughput vs Delay (5 Clusters, Min-Max)

ISSN 2723-6471

3.3. Classification Analysis



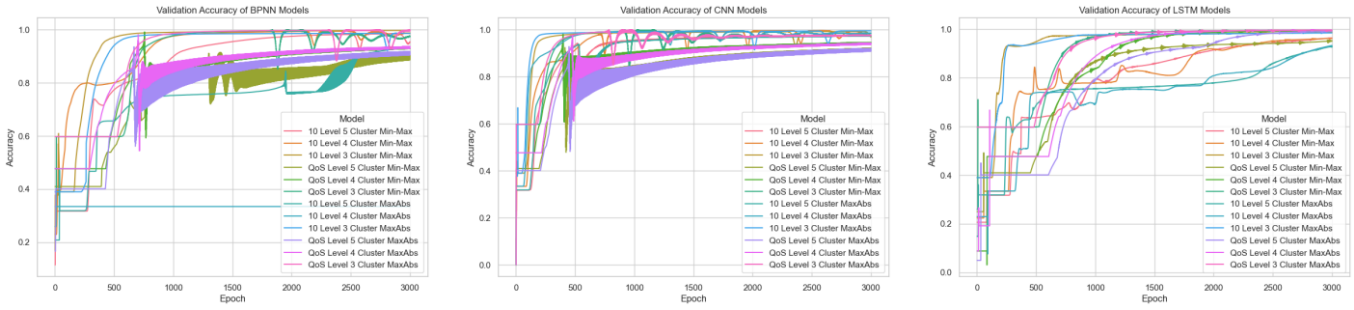
(a) (b) (c)

(d) (e)

Figure 8. Validation Loss of (a) BPNN, (b) CNN, (c) LSTM, (d) CNN-LSTM, (e) LSTM-CNN models

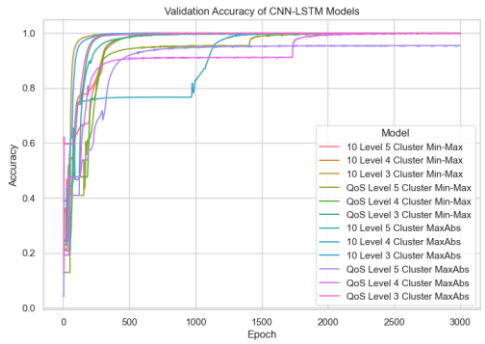
In this study, the following training configurations are employed: (1) The dataset is partitioned into three subsets for the training process: the training set comprises 60% of the total data, and the validation set makes up 20%. (2) For the model testing phase, 20% of the data is utilized as the test set. (3) Every algorithm employs Adam Optimization with a learning rate of 0.0001. (4) This study documents validation loss and validation accuracy over a span of 3000 epochs. Figure 8 presents the validation loss metric. The LSTM-CNN algorithm generates models that exhibit a high degree of effectiveness and a very low level of loss stability across all data combinations. In contrast, it can be observed that BPNN, CNN, and LSTM yield unstable losses and relatively high final losses.

The data combination that demonstrates the highest level of stability and the best final loss across all algorithms is represented by the 10 Level model with 3 Clusters, which is normalized by MaxAbs. On the other hand, the combinations of QoS Level with 5 Clusters Min-Max and MaxAbs, as well as 4 Clusters with MaxAbs, particularly in the BPNN and CNN algorithms, display the opposite characteristics.

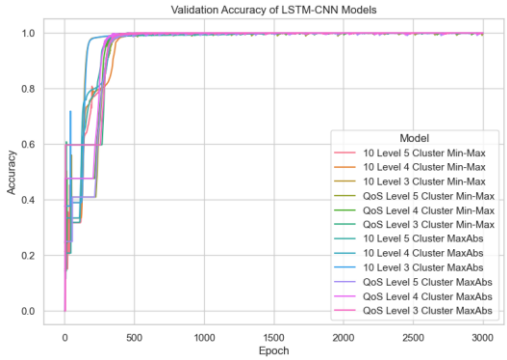


(a) (b) (c)

ISSN 2723-6471



(d)



(e)

Figure 9. Validation Accuracy of (a) BPNN, (b) CNN, (c) LSTM, (d) CNN-LSTM, (e) LSTM-CNN Models

As depicted in figure 9, similar to the validation loss metric, the models trained in this study demonstrate that the LSTM-CNN algorithm generates models that exhibit the highest level of accuracy stability among the tested deep learning algorithms. Conversely, the models produced by the BPNN, CNN, and LSTM algorithms display accuracy instability during the training process. An intriguing observation is made with the BPNN algorithm, in the 10 Level model with 4 Clusters normalized by MaxAbs. This combination shows the poorest performance in terms of loss and accuracy, but it does not appear as deficient when compared to other algorithms. Upon completion of the training process for all algorithm and data combinations, the accuracy results are presented in table 9 using the prepared test set. It is revealed that the CNN-LSTM and LSTM-CNN algorithms achieve superior performance compared to the other algorithms.

Table 9. Comparison of Models Across Different Levels and QoS Levels

	BPNN	CNN	LSTM	CNN-LSTM	LSTM-CNN
3k Min-Max 10 Level	0.98	0.97	0.99	1.00	1.00
4k Min-Max 10 Level	0.99	0.98	0.96	1.00	1.00
5k Min-Max 10 Level	0.97	0.97	0.96	1.00	1.00
3k MaxAbs 10 Level	0.97	0.99	0.98	1.00	1.00
4k MaxAbs 10 Level	0.99	0.99	0.93	1.00	1.00
5k MaxAbs 10 Level	0.94	0.97	0.93	1.00	1.00
3k Min-Max 3 QoS Level	1.00	0.97	0.99	1.00	1.00
4k Min-Max 3 QoS Level	0.94	0.95	0.99	1.00	1.00
5k Min-Max 3 QoS Level	0.94	0.92	0.95	1.00	1.00
3k MaxAbs 3 QoS Level	0.96	0.98	0.99	1.00	1.00
4k MaxAbs 3 QoS Level	0.94	0.94	1.00	1.00	1.00
5k MaxAbs 3 QoS Level	0.86	0.92	0.99	0.95	1.00

After all models have been trained, the models will be combined based on each dataset combination and produce 12 ensemble models with final results determined using the majority voting method which has been evaluated in table 10.

Table 10. Comparison of Metrics Across Different Levels and QoS Levels

	Accuracy	MCC	Cohen's Kappa	Hamming Loss
3k Min-Max 10 Level	1.00	0.99	0.99	0.00
4k Min-Max 10 Level	1.00	1.00	1.00	0.00
5k Min-Max 10 Level	0.99	0.99	0.99	0.01

3k MaxAbs 10 Level	1.00	1.00	1.00	0.00
4k MaxAbs 10 Level	0.99	0.99	0.99	0.01
5k MaxAbs 10 Level	1.00	1.00	1.00	0.00
3k Min-Max 3 QoS Level	1.00	1.00	1.00	0.00
4k Min-Max 3 QoS Level	1.00	1.00	1.00	0.00
5k Min-Max 3 QoS Level	1.00	0.99	0.99	0.00
3k MaxAbs 3 QoS Level	1.00	1.00	1.00	0.00
4k MaxAbs 3 QoS Level	1.00	1.00	1.00	0.00
5k MaxAbs 3 QoS Level	0.98	0.98	0.98	0.02

Table 10 presents a comparative analysis of various metrics, including Accuracy, MCC, Cohen's Kappa, and Hamming Loss, across different configurations involving $3k$, $4k$, and $5k$ Min-Max and MaxAbs normalization levels, as well as corresponding QoS levels. The results demonstrate consistently high performance across all metrics, with Accuracy, MCC, and Cohen's Kappa values nearing or at 1.00, indicating excellent 100% predictive capability and agreement for most configurations. Notably, the result of Hamming Loss remains minimal at 0.00, reflecting nearly misclassification rates possibility. The slight variations observed in the $5k$ Min-Max and $5k$ MaxAbs QoS levels suggest marginally reduced performance in these specific scenarios. Overall, the metrics indicate robust model performance across various levels and normalization strategies. It can also be seen that all combinations of highly recommended to using the QoS level dataset in term of features, and 3 clusters have the most consistent evaluation results from all experiments in terms of the number of clusters.

4. Conclusion

This study leveraged deep learning and ensemble learning to predict network congestion effectively. Beginning with meticulous data collection and preprocessing, it delved into correlation analysis, exploratory data analysis, cluster validation, and classification. Findings revealed nuanced insights into network parameter relationships and optimal clustering strategies. K-Means analysis highlighted patterns, with clustering focusing on key features like packet numbers and losses. Classification analysis showcased superior stability and accuracy of LSTM-CNN. Ensemble learning further enhanced predictions, yielding twelve robust models with high accuracy across diverse dataset configurations. Furthermore, it is preferably recommended to use the QoS level dataset and the combinations of 3 clusters due to the most consistent evaluation results compared to other combination experiments. In essence, this research demonstrates the power of deep learning and ensemble methods in forecasting network congestion, offering valuable strategies for enhancing network efficiency and reliability in the face of escalating traffic demands.

5. Declarations

5.1. Author Contributions

Conceptualization: P.; Methodology: P. and R.R.R.; Software: R.R.R; Validation: P., A.P.W., and A.N.; Formal Analysis: R.R.R.; Investigation: H.D.; Resources: P.; Data Curation: P.; Writing—Original Draft Preparation: H.D. and R.R.R.; Writing—Review and Editing: P., A.P.W., H., A., and A.N.; Visualization: H.D.; Supervision: A.P.W., A.N., A., and H.; Project Administration: A.P.W; Funding Acquisition: P., All authors have read and agreed to the published version of the manuscript.

5.2. Data Availability Statement

The data presented in this study are available on request from the corresponding author.

5.3. Funding

The authors received financial support for the research and publication of this article from Universitas Muslim Indonesia.

5.4. Institutional Review Board Statement

Not applicable.

5.5. Informed Consent Statement

Not applicable.

5.6. Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] L. Zhu, B. Li, and H. Zhang, "ARP spoofing forensics based on network data flow", *in Third International Conference on Computer Communication and Network Security (CCNS 2022), 2022*, vol. 12453, no. 1245307, pp. 34–39.

[2] L. Chettri and R. Bera, "A Comprehensive Survey on Internet of Things (IoT) Toward 5G Wireless Systems," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 16–32, Jan. 2020.

[3] A. Narayanan, "A variegated look at 5G in the wild," *in Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, vol. 2021, no. August, pp. 610–625, 2021.

[4] A. R. Md Jamil and N. Nower, "A Comprehensive Analysis of Reward Function for Adaptive Traffic Signal Control," *Knowl. Eng. Data Sci.*, vol. 4, no. 2, p. 85 –96, Dec. 2021.

[5] R. M. Baazeem, "Cybersecurity: BotNet Threat Detection Across the Seven-Layer ISO-OSI Model Using Machine Learning Techniques," *Comput. Informatics*, vol. 42, no. 5, pp. 1060–1090, 2023.

[6] R. Swami, M. Dave, and V. Ranga, "Detection and Analysis of TCP-SYN DDoS Attack in Software-Defined Networking," *Wirel. Pers. Commun.*, vol. 118, no. 4, pp. 2295–2317, Jun. 2021.

[7] X. Chen, X. Chen, Y. Huang, J. Lin, Y. Wu, and Y. Chen, "TCP1 increases drug resistance in acute myeloid leukemia by suppressing autophagy via activating AKT/mTOR signaling," *Cell Death Dis.*, vol. 12, no. 11, pp. 1058-1069, Nov. 2021.

[8] S. Saha, A. Haque, and G. Sidebottom, "An Empirical Study on Internet Traffic Prediction Using Statistical Rolling Model," *2022 Int. Wirel. Commun. Mob. Comput. IWCMC 2022*, vol. 2022, no. 7 pp. 1058–1063, 2022.

[9] L. Mei, "Realtime mobile bandwidth prediction using LSTM neural network and Bayesian fusion," *Comput. Networks*, vol. 182, no. March, pp. 107515-107528, 2020.

[10] M. Labonne, C. Chatzinakis, and A. Olivereau, "Predicting bandwidth utilization on network links using machine learning," *2020 Eur. Conf. Networks Commun*, vol. 2020, no. June, pp. 242–247, 2020.

[11] M. Yeo, 'Flow-based malware detection using convolutional neural network', *in 2018 International Conference on Information Networking (ICOIN)*, vol. 2018, no. January, pp. 910–913, 2018.

[12] A. A. Salih and M. B. Abdulrazaq, "Combining Best Features Selection Using Three Classifiers in Intrusion Detection System," *2019 Int. Conf. Adv. Sci. Eng. ICOASE 2019*, vol. 2019, no. April pp. 94–99, 2019.

[13] M. A. Alsheikh, D. Niyato, S. Lin, H. -p. Tan and Z. Han, "Mobile big data analytics using deep learning and apache spark," *in IEEE Network*, vol. 30, no. 3, pp. 22-29, 2016.

[14] L. Wang, J. Wang, Z. Liu, J. Zhu, and F. Qin, "Evaluation of a deep-learning model for multispectral remote sensing of land use and crop classification," *Crop J.*, vol. 10, no. 5, pp. 1435–1451, Oct. 2022.

[15] A. Munandar, W. Maulana Baihaqi, and A. Nurhopipah, "A Soft Voting Ensemble Classifier to Improve Survival Rate Predictions of Cardiovascular Heart Failure Patients," *Ilk. J. Ilm.*, vol. 15, no. 2, pp. 344–352, Aug. 2023.

[16] S. Y. Prasetyo, G. Z. Nabiilah, Z. N. Izdihar, and S. M. Isa, 'Pneumonia detection on x-ray imaging using softmax output in multilevel meta ensemble algorithm of deep convolutional neural network transfer learning models', *International Journal of Advances in Intelligent Informatics*, vol. 9, no. 2, pp 319–330, 2023.

[17] K. Trang and A. H. Nguyen, "A Comparative Study of Machine Learning-based Approach for Network Traffic

Classification," *Knowl. Eng. Data Sci.*, vol. 4, no. 2, pp. 128–137, Jan. 2022.

[18] A. P. Wibawa, "Mean-Median Smoothing Backpropagation Neural Network to Forecast Unique Visitors Time Series of Electronic Journal," *J. Appl. Data Sci.*, vol. 4, no. 3, pp. 163–174, Sep. 2023.

[19] A. Krishnaswamy Rangarajan and H. K. Ramachandran, "A fused lightweight CNN model for the diagnosis of COVID-19 using CT scan images," *Automatika*, vol. 63, no. 1, pp. 171–184, Jan. 2022.

[20] A. W. Saputra, A. P. Wibawa, U. Pujianto, A. B. Putra Utama, and A. Nafalski, "LSTM-based Multivariate Time-Series Analysis: A Case of Journal Visitors Forecasting," *Ilk. J. Ilm.*, vol. 14, no. 1, pp. 57–62, Apr. 2022.

[21] H. Hanafi, A. Pranolo, Y. Mao, T. Hariguna, L. Hernandez, and N. F. Kurniawan, "IDSX-Attention: Intrusion detection system (IDS) based hybrid MADE-SDAE and LSTM-Attention mechanism," *Int. J. Adv. Intell. Informatics*, vol. 9, no. 1, pp. 121–135, Mar. 2023.

[22] A. Halbouni, T. S. Gunawan, M. H. Habaebi, M. Halbouni, M. Kartiwi, and R. Ahmad, "CNN-LSTM: Hybrid Deep Neural Network for Network Intrusion Detection System," *IEEE Access*, vol. 10, pp. 99837–99849, 2022.

[23] A. S. Sams and A. Zahra, "Multimodal music emotion recognition in Indonesian songs based on CNN-LSTM, XLNet transformers," *Bull. Electr. Eng. Informatics*, vol. 12, no. 1, pp. 355–364, Feb. 2023.

[24] ETSI, "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON); General aspects of Quality of Service (QoS)," *Etsi Tr 101 329 V2.1.1*, vol. 1, no. 1, pp. 1–37, 1999.

[25] G. S. Nijaguna, J. A. Babu, B. D. Parameshachari, R. P. de Prado, and J. Frnda, "Quantum Fruit Fly algorithm and ResNet50-VGG16 for medical diagnosis," *Appl. Soft Comput.*, vol. 136, no. 1, pp. 110055–110068, Mar. 2023.

[26] K. Dashdondov, S.-M. Lee, and M.-H. Kim, "OrdinalEncoder and PCA based NB Classification for Leaked Natural Gas Prediction Using IoT based Remote Monitoring System," *Advances in Intelligent Information Hiding and Multimedia Signal Processing*, vol. 2, pp. 252–259, 2021.

[27] I. M. A. P. J. Dashdondov, "Video Streaming QoS Analysis with TIPHON Standard NDNts," *in 2023 9th International Conference on Wireless and Telematics (ICWT)*, vol. 2023, no. July, pp. 1–4, 2023.

[28] S. Özarpacı, B. Kılıç, O. C. Bayrak, A. Özdemir, Y. Yılmaz, and M. Floyd, "Comparative analysis of the optimum cluster number determination algorithms in clustering GPS velocities," *Geophys. J. Int.*, vol. 232, no. 1, pp. 70–80, Sep. 2022.

[29] A. Shokouhmand and N. Tavassolian, "Fetal Electrocardiogram Extraction Using Dual-Path Source Separation of Single-Channel Non-Invasive Abdominal Recordings," *IEEE Trans. Biomed. Eng.*, vol. 70, no. 1, pp. 283–295, Jan. 2023.

[30] R. Herdianto, P. Setyosari, D. Kuswandi, A. P. Wibawa, A. Nafalski, and I. M. P. Pradana, "Indonesian education: A future promise," *Int. J. Educ. Learn.*, vol. 4, no. 3, pp. 202–213, Dec. 2022.

[31] T. Yan, S.-L. Shen, and A. Zhou, "Identification of geological characteristics from construction parameters during shield tunnelling," *Acta Geotech.*, vol. 18, no. 1, pp. 535–551, Jan. 2023.

[32] L. Gotsev, 'Cluster analysis and ensemble transfer learning for COVID-19 classification from computed tomography scans', *International Journal of Advances in Intelligent Informatics*, vol. 8, no. 2, pp. 135–150, 2022.

[33] T. Widiyaningtyas, I. Hidayah, and T. B. Adji, "Recommendation Algorithm Using Clustering-Based UPCSim (CB-UPCSim)," *Computers*, vol. 10, no. 10, pp. 123-132, Oct. 2021.

[34] S. R. Jabir, Purnawansyah, H. Darwis, H. Lahuddin, A. Faradibah, and A. W. M. Gaffar, "Evaluation of Tourism Object Rating Using Naïve Bayes, Support Vector Machine, and K-Means for Business Intelligence Application in Indonesia Tourism," *in 2024 18th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, vol. 2024, no. January, pp. 1–8, 2024.

[35] D. F. Laistulloh, A. N. Handayani, R. A. Asmara, and P. Taw, "Convolutional Neural Network in Motion Detection for Physiotherapy Exercise Movement," *Knowl. Eng. Data Sci.*, vol. 7, no. 1, pp. 27–39, May 2024.

[36] A. Pranolo, "Exploring LSTM-based Attention Mechanisms with PSO and Grid Search under Different Normalization Techniques for Energy demands Time Series Forecasting," *Knowl. Eng. Data Sci.*, vol. 7, no. 1, pp. 1–12, Apr. 2024.

[37] Y. I. Sulistya, E. T. Br Bangun, and D. A. Tyas, "CNN Ensemble Learning Method for Transfer learning: A Review," *Ilk. J. Ilm.*, vol. 15, no. 1, pp. 45–63, Apr. 2023.

[38] L. Menshawy, A. H. Eid, and R. F. Abdel-Kader, "Ensemble deep models for covid-19 pandemic classification using chest x-ray images via different fusion techniques," *Int. J. Adv. Intell. Informatics*, vol. 9, no. 1, pp. 51 –65, Mar. 2023.

[39] W. Wang, Y. Pei, S.-H. Wang, J. manuel Gorrz, and Y.-D. Zhang, "PSTCNN: Explainable COVID-19 diagnosis using PSO-guided self-tuning CNN," *Biocell Off. J. Soc. Latinoam. Microsc. Electron.*, vol. 47, no. 2, pp. 373 –384, 2023.

[40] D. Chicco, M. J. Warrens, and G. Jurman, "The Matthews Correlation Coefficient (MCC) is More Informative Than Cohen's Kappa and Brier Score in Binary Classification Assessment," *IEEE Access*, vol. 9, no. May, pp. 78368–78381, 2021.

[41] T. M. Usman, Y. K. Saheed, D. Ignace, and A. Nsang, "Diabetic retinopathy detection using principal component analysis multi-label feature extraction and classification," *Int. J. Cogn. Comput. Eng.*, vol. 4, no. June pp. 78–88, Jun. 2023.