EAI Endorsed Transactions
on Internet of Things
Research Article EAI.EU

# Developing a Deep Learning-Based Multimodal Intelligent Cloud Computing Resource Load Prediction System

Ruey-Chyi Wu[1*]

[1]Bachelor Degree Program of Digital Marketing, National Taipei University, Taipei, Taiwan

## Abstract

This study aims to predict the dynamic changes in critical cloud computing resource indicators, namely Central Processing Unit (CPU), Random Access Memory (RAM), hard disk (Disk), and network. Its primary objective is to optimize resource allocation strategies in advance to enhance overall system performance. The research employs various deep learning algorithms, including Simple Recurrent Neural Network (SRNN), Bidirectional Simple Recurrent Neural Network (BiSRNN), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU). Through experimentation with different algorithm combinations, the study identifies optimal models for each specific resource indicator. Results indicate that combining CNN, LSTM, and GRU yields the most effective predictions for CPU load, while CNN and LSTM together are optimal for RAM load prediction. For disk load prediction, GRU alone proves optimal, and BiSRNN emerges as the optimal choice for network load prediction. The training results of these models demonstrate R-squared values ($R^2$) exceeding 0.98, highlighting their high accuracy in predicting future resource dynamics. This precision facilitates timely and efficient resource allocation, thereby enhancing system responsiveness. The study's multimodal precise prediction capability supports prompt and effective resource allocation, further enhancing system responsiveness. Ultimately, this approach significantly contributes to sustainable digital advancement for enterprises by ensuring efficient resource allocation and consistent optimization of system performance. The study underscores the importance of integrating advanced deep learning techniques in managing cloud computing resources, thereby supporting the robust and sustainable growth of digital infrastructure.

**Keywords:** Cloud Computing, Deep Learning, Prediction, Bidirectional, CNN, LSTM, GRU.

Received on 8 June 2024, accepted on 22 June 2024, published on 19 07 2024



doi: 10.4108/eetiot.6296

* Corresponding author. Email: rueychyiwu@gmail.com

## 1. Introduction

In the dynamic environment of cloud computing, effective utilization and management of computational resources—particularly Central Processing Unit (CPU), Random Access Memory (RAM), hard disk (Disk), and network—are crucial for maintaining system performance and meeting the demands of modern applications. The variability and unpredictability of resource loads in cloud environments pose significant challenges to system administrators and developers, often leading to improper resource allocation and potential inefficiencies. With enterprises increasingly relying on cloud infrastructure, there is an urgent need for intelligent systems capable of predicting resource loads to optimize resource allocation in advance.

This study is motivated by the pressing need to address these challenges. Inefficient resource allocation can result in wasted computing power, increased costs, and decreased performance, negatively impacting operational efficiency and service quality for enterprises. As cloud-based services continue to expand, the pressure on system administrators to maintain optimal performance levels intensifies. This underscores the necessity of advanced predictive models

that can accurately forecast resource demands, enabling proactive adjustments and efficient resource utilization.

The objective of this research is to leverage deep learning techniques, using artificial neural network models trained on historical usage data, to predict future resource utilization. By enhancing system performance, businesses can proactively optimize resource allocation strategies, facilitating sustainable digital advancement. The study integrates various deep learning algorithms to handle time-series data and capture complex patterns within data sequences, including Simple Recurrent Neural Network (SRNN), Bidirectional SRNN (BiSRNN), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU). Exploration of different model combinations aims to identify the optimal models for predicting these multimodal resource loads.

Reliable prediction of future resource demands enables the system to promote proactive resource allocation, mitigate risks of over-provisioning or under-provisioning, and ultimately enhance overall system efficiency. This predictive capability is particularly valuable in environments with unstable resource demands, enabling enterprises to maintain high levels of service quality and operational efficiency.

## 2. Literature Review

This section delves into pertinent literature concerning a variety of topics including deep learning neural networks, cloud computing, and IoT (Internet of Things).

### 2.1. Application of IoT and Cloud Computing

The primary objective of IoT is to enable internet connectivity among objects, facilitating data collection, transmission, and analysis to achieve smarter, automated, and more efficient operations. It finds wide-ranging application, including smart homes, smart cities, industrial automation, agriculture, healthcare, transportation and logistics. When sensor processing systems utilize advanced communication technologies such as Bluetooth Low Energy, LoRA, ZigBee, Insteon, 3G, 4G, 5G, it defines a burgeoning technological field — the Internet of Things (Figure 1) [1].

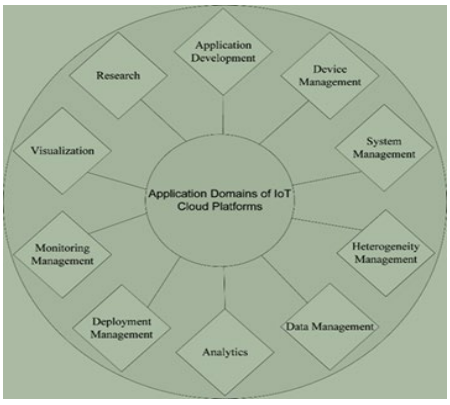



**Figure 1.** Application domains of IoT cloud platforms

For example, Pham et al. [2] introduced a novel algorithm aimed at enhancing the efficiency of cloud-based smart parking systems, leveraging IoT technology to develop a network architecture. This system assists users in automatically finding the most cost-effective parking spaces based on new performance metrics that consider distance and total available space in each parking lot to calculate the user's parking cost. Elements of a Smart IoT Parking System are illustrated in Figure 2.

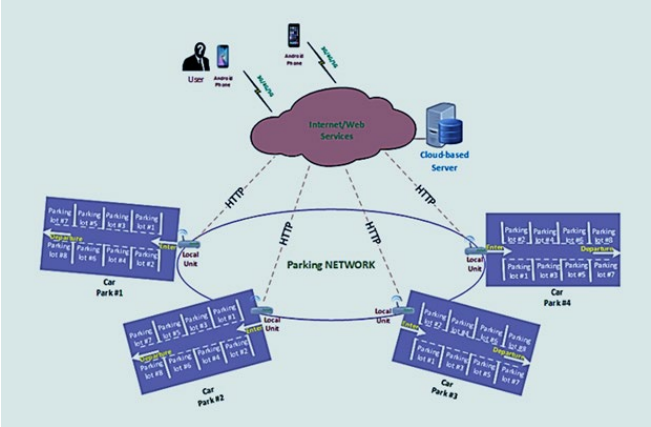



**Figure 2**. Elements of a Smart IoT Parking System

Al-Asaly et al. [3] proposed a method based on autonomic computing and deep learning, focusing on predicting virtual machine CPU usage to address user demand challenges. They adopted a Diffusion Convolutional Recurrent Neural Network (DCRNN) model, validating its effectiveness in predicting accuracy through experiments with real data, demonstrating an average absolute percentage error of 0.18 and a root mean square error of 2.40.

### 2.2. deep learning

Frank et al. [4] introduced several key deep learning methods, including Deep Feedforward Neural Networks (D-FFNN), CNNs, Deep Belief Networks (DBN), Autoencoders (AE), and LSTMs. These models constitute the foundational frameworks of current deep learning, essential tools for every data scientist to master. These fundamental frameworks can be flexibly combined, akin to LEGO bricks, to construct new application-specific network architectures. Therefore, a fundamental understanding of these network frameworks is crucial for preparing for the future development of artificial intelligence.

### 2.3. BiSRNN

Schuster et al. [5] mentioned extending conventional RNN to BiSRNN, which are not restricted by input information during training and can predict future frames. This is achieved by training simultaneously in both forward and backward time directions. In regression and classification

experiments on artificial data, the proposed structure outperformed other methods.

## 2.4. CNN

Chauhan et al. [6] elucidated that CNNs are a deep learning approach suitable for large datasets, which employ filters to perform convolution on input images, generating the desired output. This paper constructed a CNN model and evaluated its performance on image recognition and detection datasets. On the MNIST dataset, the model achieved an accuracy of 99.6%; on the CIFAR-10 dataset, real-time data augmentation and dropout processing were performed using a CPU. Figure 3 depicts the CNN architecture [7].

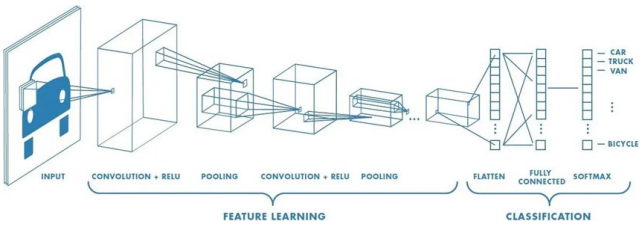



**Figure 3**. CNN Architecture

## 2.5. LSTM and GRU

Smagulova et al. [8] mentioned that LSTM units in neural networks can have multiple directions and dimensions. Commonly used are unidirectional and bidirectional LSTMs, where bidirectional LSTMs share inputs and outputs, enabling them to learn different features. Stacking multiple LSTMs generates hierarchical LSTMs capable of storing more information. Additionally, there are tree-structured LSTMs, where units reflect parent-child node information, similar to human speech. Unidirectional, bidirectional, and tree-structured LSTMs are depicted in Figure 4, Figure 5, and Figure 6, respectively.

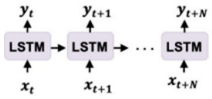



**Figure 4**. Unidirectional LSTM

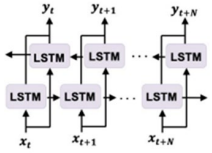



**Figure 5**. Bidirectional LSTM

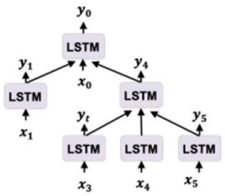



**Figure 6**. Tree-structured LSTM

LSTM was initially proposed by Hochreiter and Schmidhuber [9] in 1997. Since then, it has spawned various variants. Compared to regular RNNs, LSTM introduces input gates and forget gates to address the issues of vanishing gradients and exploding gradients, enabling it to capture long-term information and perform better in long sequence texts. The input and output structure of GRU is similar to that of regular RNNs, but its internal structure is similar to LSTM. The internal structure comparison between LSTM and GRU is shown in Figure 7 and Figure 8. LSTM and GRU are two popular variants of RNNs with long-term memory. In terms of model training speed, GRU processes the same dataset 29.29% faster than LSTM; in terms of performance, GRU outperforms LSTM in scenarios with long texts and small datasets, but lags behind LSTM in other scenarios [10].

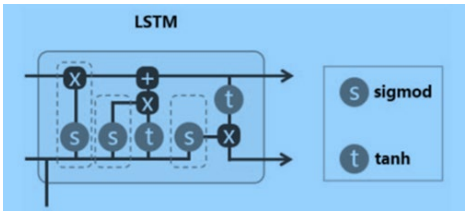



**Figure 7**. Internal Structure of LSTM

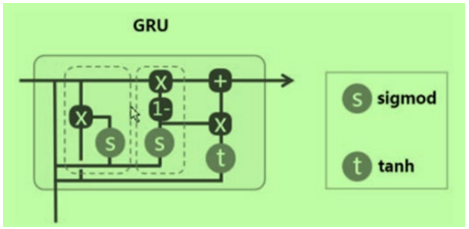



**Figure 8**. Internal Structure of GRU

## 3. Research Methodology

The following are the steps of this study, as depicted in Figure 9:

(i) Problem Definition: Explain the prediction of key indicators of resource operation environment required for managing virtualization platform.
(ii) Data Collection and Preprocessing: Gather data on CPU, RAM, disk, and network performance within cloud computing environments, subsequently refining and transforming the data in preparation for further analysis and modeling.
(iii) Evaluation of Optimal Deep Learning Algorithms: Evaluate various neural network architectures, including SRNN, BiSRNN, CNN, LSTM, GRU, CNN-LSTM, CNN-GRU, CNN-LSTM-GRU, etc.
(iv) Selection of Optimal Model Algorithm: Select the optimal-performing single or combination algorithm from the previous step.
(v) Deploy to Visualization Dashboard and Message Dispatch: Finally, transmit the time-series prediction

waveforms generated by the trained models to relevant stakeholders through a visualization dashboard system as proactive measures.

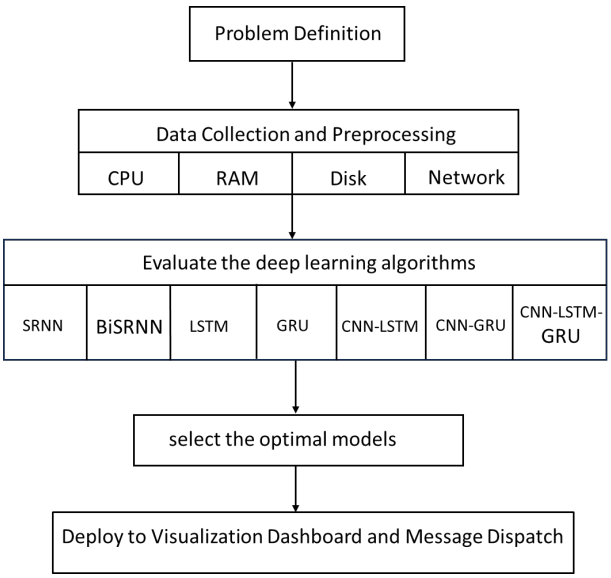



**Figure 9**. Research Framework Diagram

## 4. System Implementation and Evaluation

In this study, the system implementation process is outlined in the following subsections.

### 4.1. Problem Definition, Data Collection, and Preprocessing

This study investigates the prediction of future dynamic operations of key indicators, including CPU, RAM, Disk usage ratio, and Network usage, on the same cloud virtual machine. The data sources include:

(i) Environmental Monitoring Sensors:
- Temperature sensors: Monitor air and water temperatures.
- Humidity sensors: Measure the relative humidity of the air.
- Air quality sensors: Detect pollutant concentrations in the air, such as PM2.5 and CO2.

(ii) Mechanical Equipment Monitoring:
- Energy management systems: Monitor the energy consumption of factories or facilities, including electricity and water.
- Remote monitoring systems: Real-time monitoring and management of remote
- devices and infrastructure.

(iii) Vehicles and Traffic Systems:
- On-board Diagnostics System (OBD): Provides vehicle operational data, such as speed.
- Vehicle tracking devices: Provide vehicle location and driving route data.

(iv) Other Administrative and Business Information Management Systems.

### 4.2. Evaluate the optimal deep learning algorithms

This study utilizes the Python programming language, leveraging libraries such as NumPy, Pandas, Matplotlib, and Scikit-learn, tensorflow. The implementation is carried out on the Colab Notebooks cloud platform [11].

In this study, common regression algorithm evaluation metrics are adopted to measure CNN and RNNs.

(i) Mean Absolute Error (MAE) [12]:
- MAE is the average of the absolute errors between predicted values and actual values, used to measure the average accuracy of the model predictions.
- It represents the average absolute error, being sensitive to the magnitude of errors without being affected excessively by a few extreme values, making MAE better reflect prediction errors under normal circumstances.
- A smaller MAE indicates better prediction performance, with predicted values closer to actual values.

(ii) Root Mean Square Error (RMSE) [13]:
- RMSE measures the square root of the mean of the squared errors between predicted values and actual values.
- It is used to measure the deviation between predicted values and actual observed values, particularly sensitive to larger errors.
- A smaller RMSE indicates more accurate predictions.

(iii) Coefficient of determination (R-squared or $R^2$) [14]:
- R-squared indicates the extent to which the model explains the variability of the data, representing the goodness of fit of the model.
- The closer $R^2$ is to 1, the better the model fits the data.

To predict future CPU, RAM, Disk usage, and network usage, this study explores various methods during the model selection process, including SRNN, BiSRNN, CNN, LSTM, GRU, CNN-LSTM, CNN-GRU, and CNN-LSTM-GRU. The loss curve graphs of each trained model depict the validation set curves used to evaluate the training fit, while the test set curves demonstrate the model's generalization ability. In addition, metrics such as MAE, RMSE, and $R^2$ are used to compare the actual values of the training set, validation set, and test set. The ranking superiority order will be determined by the majority vote of three evaluation indicators.

#### 4.2.1. Evaluating and Selecting the Optimal Training Model for CPU

Figure 10 displays a partial data graph of the CPUs in the dataset. The MAE, RMSE, and $R^2$ for the training results of various models are presented in Table 1.

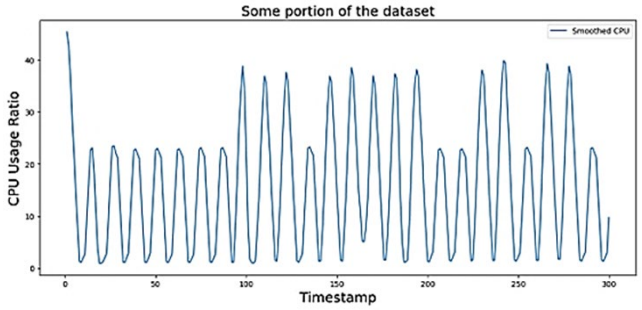



**Figure 10.** A portion of the data graph from the CPU in the dataset

Table 1. Test scores and rankings of various learning models for CPU

| CPU Models | MAE | RMSE | $R^2$ | Rank |
|---|---|---|---|---|
| SRNN | 2.55327 | 4.06997 | 0.92464 | 7 |
| BiSRNN | 2.13060 | 3.49882 | 0.94429 | 6 |
| LSTM | 1.32208 | 1.89668 | 0.98363 | 5 |
| GRU | 1.24383 | 1.89662 | 0.98363 | 4 |
| CNN-LSTM | 1.16324 | 1.77279 | 0.98570 | 2 |
| CNN-GRU | 1.24364 | 1.83286 | 0.98471 | 3 |
| CNN-LSTM-GRU | 0.75638 | 1.16523 | 0.99224 | 1 |

Based on the comparison of the MAE, RMSE, and $R^2$ results from the table above, where lower MAE and RMSE values are preferable and $R^2$ closer to 1 is better, it can be concluded that the fusion algorithm CNN-LSTM-GRU is the optimal integrated model for CPU prediction.

The training steps for the CNN-LSTM-GRU combination are as follows:

(i) Data Preparation: Process and prepare the time-series data, including loading data, preprocessing (such as normalization, padding, etc.), and splitting the dataset.
(ii) Constructing the CNN Part: Build a CNN to extract spatial features from the time-series data. This can consist of one or multiple convolutional layers, possibly with pooling layers in between to reduce the size of feature maps.
(iii) Constructing the LSTM and GRU Part: Stack LSTM and GRU layers on top of the CNN output to capture temporal dependencies. These RNN layers will handle the time-series data and remember previous states.
(iv) Model Integration: Combine the CNN and LSTM/GRU parts to build the entire model. Transform the output of the CNN into a sequential form and then feed it into the LSTM and GRU layers.
(v) Model Definition: Define the model's loss function, optimizer, and evaluation metrics. This depends on the task type and model architecture.
(vi) Model Compilation: Compile the model using the selected loss function, optimizer, and evaluation metrics.
(vii) Model Training: Train the model using the training dataset, optimizing the model's performance by adjusting hyperparameters and monitoring the training process.
(viii) Model Validation: Evaluate the model's performance using the validation set and adjust as necessary.
(ix) Model Testing: Evaluate the final model's performance using the test set. This can help determine the model's generalization ability and effectiveness in real-world applications.

Figure 11 illustrates the CNN-LSTM-GRU ensemble model's adeptness in fitting and generalization across the training, validation, and test datasets, complemented by real experimental data. Additionally, Figure 12 displays the future 48-hour prediction graph.

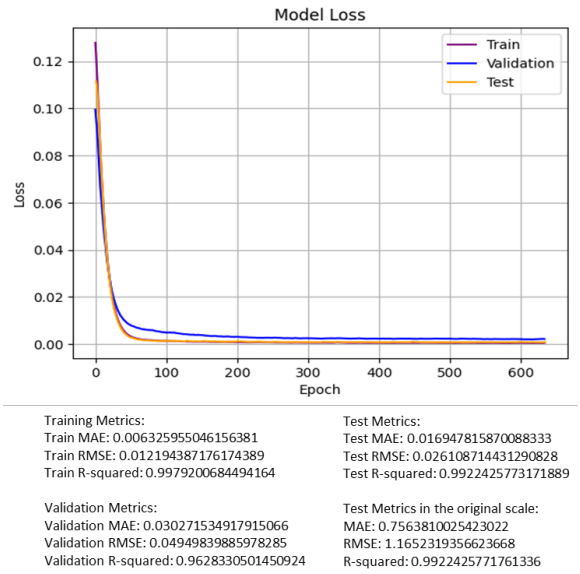



**Figure 11.** The CNN-LSTM-GRU ensemble model's performance

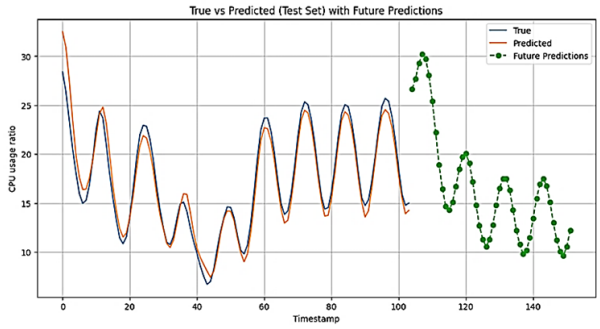



**Figure 12.** Future 48-Hour Prediction Graph of the CNN-LSTM-GRU Ensemble Model

## 4.2.2. Evaluating and Selecting the Optimal Training Model for RAM

Based on the RAM data in the dataset, as illustrated in Figure 13, and the training results of various models for RAM, including MAE, RMSE, and $R^2$ as shown in Table 2, the optimal training model for RAM can be evaluated and selected.

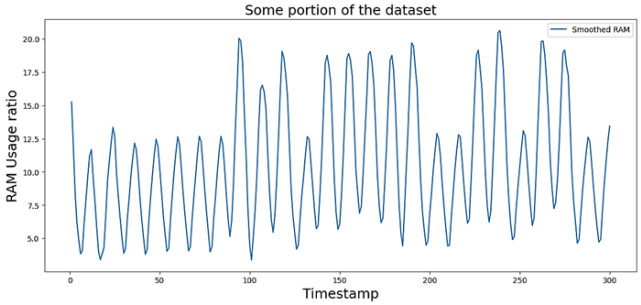



**Figure 13.** A portion of the data graph from the RAM in the dataset

Table 2. Test scores and rankings of various learning models for RAM

| RAM Models | MAE | RMSE | $R^2$ | Rank |
|---|---|---|---|---|
| SRNN | 1.52156 | 1.98511 | 0.89870 | 4 |
| BiSRNN | 1.40380 | 1.84866 | 0.91216 | 3 |
| LSTM | 1.73784 | 2.31335 | 0.87013 | 6 |
| GRU | 2.09211 | 2.88481 | 0.79519 | 7 |
| CNN-LSTM | 0.45580 | 0.59892 | 0.98291 | 1 |
| CNN-GRU | 1.29000 | 1.81435 | 0.91538 | 2 |
| CNN-LSTM-GRU | 2.18769 | 1.76545 | 0.88385 | 5 |

Based on the comparison of MAE, RMSE, and $R^2$ results from the table above, the CNN-LSTM fusion algorithm is determined to be the optimal integrated model for RAM prediction. CNN-LSTM is a model that combines Convolutional CNN and LSTM. The execution steps are as follows:

(i) Load the time series data.
(ii) Build the CNN part: Construct the CNN part of the model to process the spatial features of the data. This may include one or multiple convolutional layers and pooling layers to extract features and reduce the size of feature maps.
(iii) Build the LSTM part: Stack LSTM layers on top of the CNN output. The LSTM layers will handle the time series data and capture temporal dependencies within the sequences.
(iv) Combine the model: Integrate the CNN and LSTM parts to build the complete model. Typically, this involves transforming the output of the CNN into a sequential form and feeding it into the LSTM layers.
(v) Define the model: Define the model's loss function, optimizer, and evaluation metrics. This depends on the task type and model architecture.
(vi) Compile the model: Compile the model using the selected loss function, optimizer, and evaluation metrics.
(vii) Train the model: Train the model using the training dataset. Optimize the model's performance by adjusting hyperparameters and monitoring the training process.
(viii) Validate the model: Evaluate the model's performance using the validation dataset and make adjustments as necessary.
(ix) Test the model: Assess the final model's performance using the test dataset to determine its generalization ability and effectiveness in real-world applications.

Figure 14 illustrates the CNN-LSTM ensemble model's adeptness in fitting and generalization across the training, validation, and test datasets, complemented by real experimental data. Additionally, Figure 15 displays the future 48-hour prediction graph.

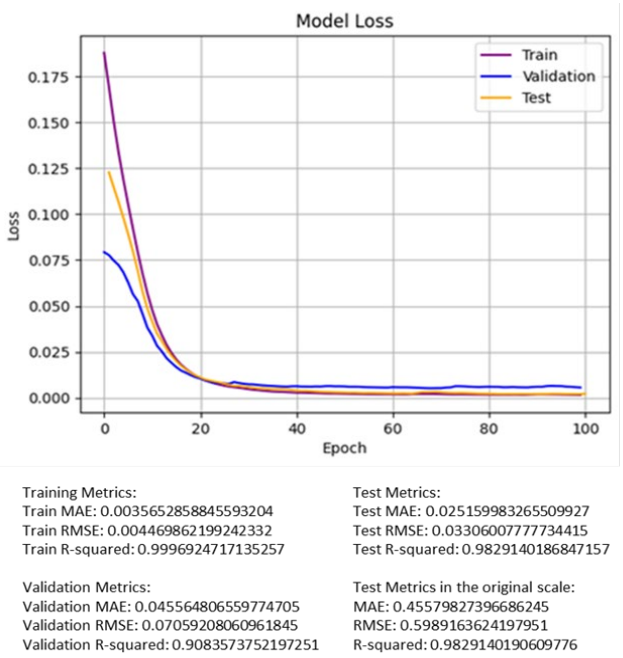



**Figure 14.** The CNN-LSTM ensemble model's performance

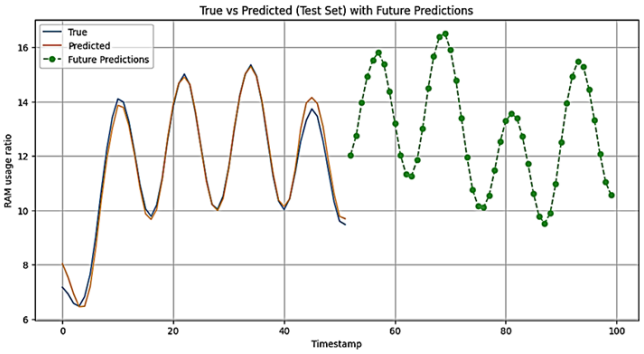



**Figure 15.** Future 48-Hour Prediction Graph of the CNN-LSTM Ensemble Model

### 4.2.3. Evaluating and Selecting the Optimal Training Model for Disk

Partial Disk data from the dataset is shown in Figure 16. The training results of various models for Disk, including MAE, RMSE, and $R^2$, are summarized in Table 3.

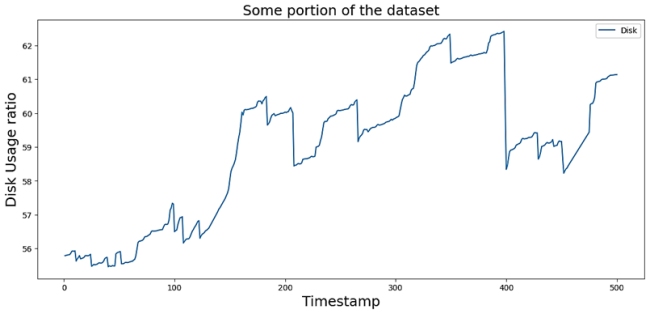



**Figure 16.** A portion of the data graph from the Disk in the dataset

Table 3. Test scores and rankings of various learning models for Disk

| Disk Models | MAE | RMSE | $R^2$ | Rank |
|---|---|---|---|---|
| SRNN | 0.10369 | 0.21837 | 0.98164 | 2 |
| BiSRNN | 0.12036 | 0.23122 | 0.98507 | 3 |
| LSTM | 0.13193 | 0.25252 | 0.98104 | 6 |
| GRU | 0.09742 | 0.14627 | 0.99456 | 1 |
| CNN-LSTM | 0.13778 | 0.25357 | 0.98229 | 5 |
| CNN-GRU | 0.12544 | 0.25395 | 0.98183 | 7 |
| CNN-LSTM-GRU | 0.13178 | 0.25357 | 0.98229 | 4 |

Comparing the results of MAE, RMSE, and $R^2$ for each model trained from the table above, it is evident that the GRU algorithm is the optimal choice for the Disk model. Here are the steps for implementing a model using GRU:

(i) Data Preparation: Load the time-series data.
(ii) Model Construction: Create a neural network model containing GRU layers. Design the model structure based on the nature of the problem and the characteristics of the data.
(iii) Model Definition: Define the model's loss function, optimizer, and evaluation metrics.
(iv) Model Compilation: Compile the model using the selected loss function, optimizer, and evaluation metrics.
(v) Model Training: Train the model using the training dataset. Optimize the model's performance by adjusting hyperparameters and monitoring the training process.
(vi) Model Validation: Evaluate the model's performance using the validation set. Adjustments may be needed to determine if the model is overfitting or underfitting.
(vii) Model Testing: Evaluate the final model's performance using the test set. This helps determine the model's generalization ability and performance in real-world applications.

Figure 17 illustrates the GRU model's adeptness in fitting and generalization across the training, validation, and test datasets, complemented by real experimental data. Additionally, Figure 18 displays the future 72-hour prediction graph.

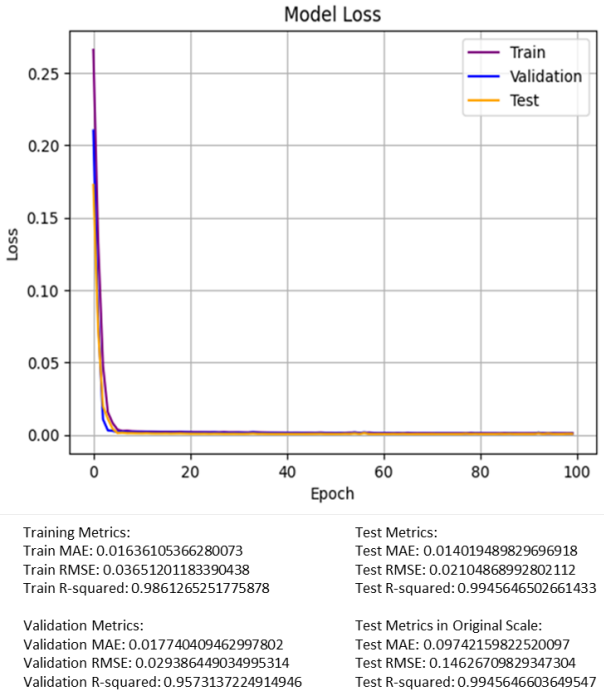



**Figure 17.** The GRU model's performance

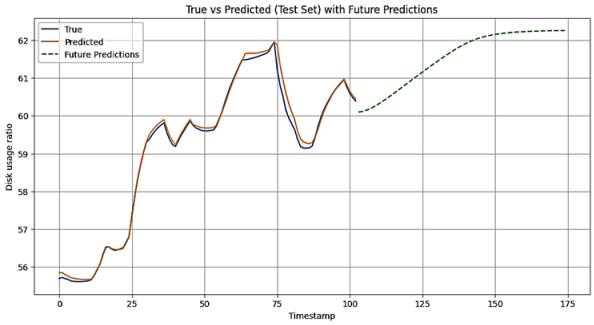



**Figure 18.** Future 72-Hour Prediction Graph of the GRU Model

### 4.2.4. Evaluation and Selection of the Optimal Training Model for Network

The partial Network data from the dataset is depicted in Figure 19. The training results of various models, including their MAE, RMSE, and $R^2$ scores, are summarized in Table 4.

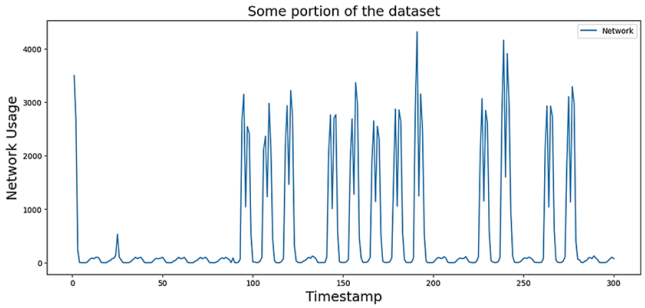



**Figure 19.** A portion of the data graph from the Network in the dataset

Table 4. Test scores and rankings of various learning models for Network

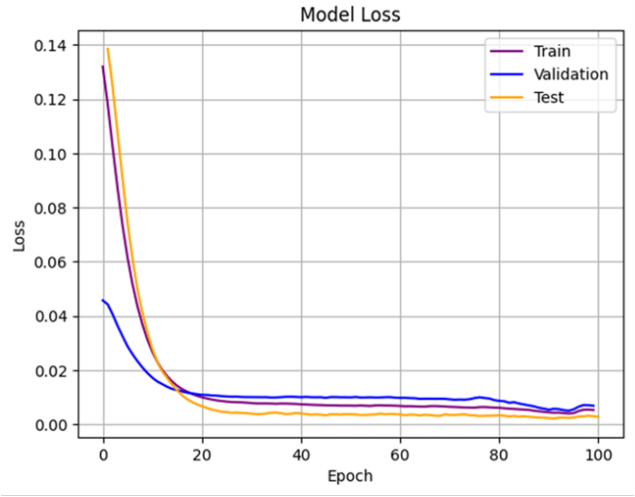
| Network Models | MAE | RMSE | $R^2$ | Rank |
|---|---|---|---|---|
| SiRNN | 158.99740 | 220.63885 | 0.97095 | 2 |
| BiSRNN | 121.20190 | 178.53639 | 0.98098 | 1 |
| LSTM | 328.98488 | 586.06498 | 0.42080 | 7 |
| GRU | 240.56327 | 500.67304 | 0.57729 | 6 |
| CNN-LSTM | 161.25082 | 375.45397 | 0.76229 | 4 |
| CNN-GRU | 254.68547 | 463.85682 | 0.63717 | 5 |
| CNN-LSTM-GRU | 160.64600 | 360.52747 | 0.78081 | 3 |

Based on the comparison of the MAE, RMSE, and $R^2$ scores of each model from the table, it is evident that the BiSRNN algorithm is the optimal choice for the Network model. This conclusion is drawn because smaller values of MAE and RMSE are preferable, and an $R^2$ value closer to 1 indicates better model performance.

The execution steps for the model using BiSRNN are as follows:

(i) Data Preparation: Load the time series data.
(ii) Model Building: Create a neural network model containing BiSRNN layers. BiSRNN can propagate information in both forward and backward directions, aiding in capturing complex patterns in sequence data.
(iii) Model Definition: Define the model's loss function, optimizer, and evaluation metrics. These choices depend on the task type and model architecture.
(iv) Model Compilation: Compile the model using the selected loss function, optimizer, and evaluation metrics.
(v) Model Training: Train the model using the training dataset. Optimize the model's performance by adjusting hyperparameters and monitoring the training process.
(vi) Model Validation: Evaluate the model's performance using the validation dataset. Adjustments may be needed to determine if the model is overfitting or underfitting.
(vii) Model Testing: Evaluate the final model's performance using the test dataset. This helps determine the model's generalization ability and performance in real-world applications.

Figure 20 illustrates the BiSRNN model's adeptness in fitting and generalization across the training, validation, and test datasets, complemented by real experimental data. Additionally, Figure 21 displays the future 48-hour prediction graph.




Training Metrics:
Train MAE: 0.03354570558448227
Train RMSE: 0.061859198188329294
Train R-squared: 0.9424593149476662

Validation Metrics:
Validation MAE: 0.040086379469593446
Validation RMSE: 0.07652956067552512
Validation R-squared: 0.8690130704557835

Test Metrics:
Test MAE: 0.028068991234713556
Test RMSE: 0.04134701061396016
Test R-squared: 0.9809780079891662

Test Metrics in the original scale:
MAE: 121.20190091010852
RMSE: 178.5363952221209
R-squared: 0.9809780072665757

**Figure 20.** The BiSRNN model's performance

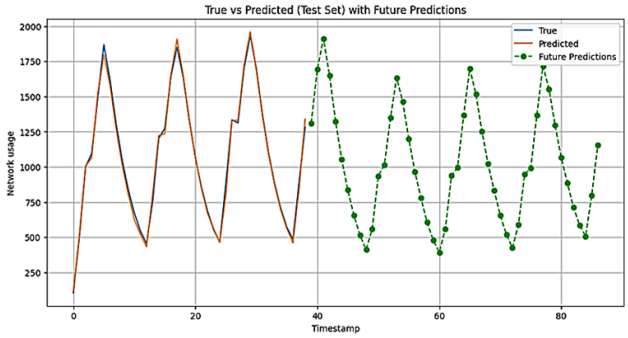



**Figure 21.** Future 48-Hour Prediction Graph of the BiSRNN Model

## 4.2.5. Summary of Implementation Results and Research Findings

The summary of the implementation results for the optimal models is shown in Table 5. The table reveals the following findings:

(i) CNN-LSTM-GRU is identified as the optimal combination model for predicting CPU load, achieving an $R^2$ value exceeding 0.99.
(ii) CNN-LSTM is identified as the optimal combination model for predicting RAM load, achieving an $R^2$ value exceeding 0.98.
(iii) GRU is identified as the optimal model for predicting disk load, achieving an $R^2$ value exceeding 0.99.
(iv) Bi-directional SRNN is identified as the optimal model for predicting network load, achieving an $R^2$ value exceeding 0.98.

Table 5. Summary of Optimal Models

| Resources | Optimal models | MAE | RMSE | $R^2$ |
|---|---|---|---|---|
| CPU | CNN-LSTM-GRU | 0.75638 | 1.16523 | 0.99224 |
| RAM | CNN-LSTM | 0.45580 | 0.59592 | 0.98291 |
| Disk | GRU | 0.09742 | 0.14627 | 0.99456 |
| Network | BiSRNN | 121.20190 | 178.53639 | 0.98098 |

## 4.3. Encapsulate into a unified model and deploy to a Visualization Dashboard and Message Dispatch

In the preceding section, this study presented a comprehensive process, as shown in Figure 22. Ultimately, the optimal models addressing various aspects of system performance were selected and integrated into a unified model, which is connected to the dashboard and messaging system for analysis and dissemination.

Deploying this prediction system to the visualization dashboard system and message delivery enhances the intuitiveness and comprehensibility of the predicted time series waveforms for CPU, RAM, disk usage ratios, and network usage. This not only provides a convenient way to showcase the predictive capabilities of the models but also promotes transparency and efficiency in the decision-making process. Furthermore, by leveraging various real-time messaging software to promptly deliver messages to relevant stakeholders, it lays the foundation for implementing response measures.

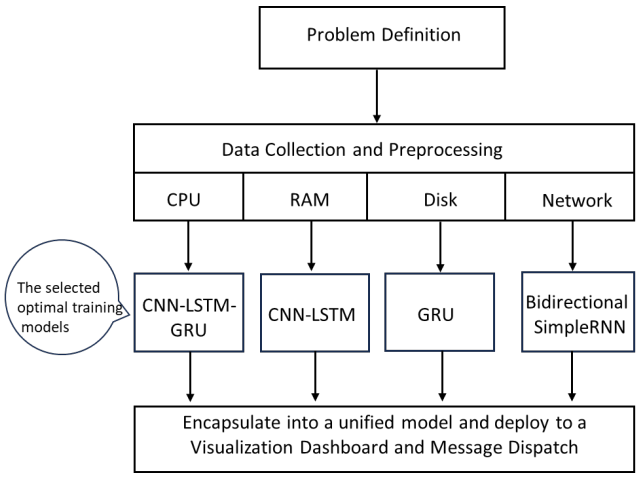



**Figure 22.** The Construction Process of the Final Model

# 5. Conclusion and Recommendations

This study has made significant contributions to resource load prediction in the field of cloud computing through the implementation of deep learning models. Tailored neural network model combinations for different types of resources have demonstrated the potential for highly accurate and effective resource management solutions. As cloud computing continues to evolve, implementing such intelligent systems will be crucial for maintaining optimal performance and supporting the growth and innovation of digital enterprises. The main contributions and future recommendations of this study are detailed in the following subsections.

## 5.1. Contributions

(i) High Accuracy Predictive Results: The models trained in this research demonstrated exceptionally high accuracy, with R-squared values ($R^2$) exceeding 0.98. This indicates the models' high precision in predicting future resource dynamics. Such precise predictive capability facilitates timely and efficient resource allocation, enhancing system responsiveness.

(ii) Enhancement of System Performance and Resource Management Efficiency: Through precise prediction and optimized resource allocation strategies, this study significantly enhanced the overall performance and operational efficiency of cloud computing systems. It reduced the risks of resource over-allocation or under-allocation, ensuring efficient system operations.

(iii) Support for Sustainable Digital Transformation: This research emphasized the integration of deep learning technologies in cloud computing resource management. This support not only ensures robust and sustainable growth of enterprise digital infrastructure but also enhances the efficiency of resource allocation, driving sustainable progress in digital transformation.

(iv) Facilitation of Proactive Resource Allocation Management: By leveraging historical usage data for deep learning-based predictions, this study provided insights into future resource utilization. It assisted enterprises in proactively optimizing resource allocation strategies, thereby improving service quality and operational efficiency, which is particularly valuable in environments with unstable resource demands.

## 5.2. Future Recommendations

(i) Model Application Expansion: While this study has validated the effectiveness of different combination models in predicting CPU, RAM, disk, and network loads, there is still a need to further explore and expand the applicability of these models in other resources and scenarios. For example, expanding the application scope to other hardware resources, environmental control resources, or different contexts can broaden the model's application value and practical range.

(ii) Smoothing Techniques: Introduce smoothing techniques into the model's prediction results to reduce the volatility of predicted values, thereby improving the stability and reliability of the prediction results. Methods such as moving averages, exponential smoothing, or filters can be used to smooth the prediction results, eliminating noise and fluctuations to make the results more interpretable and credible.

(iii) Multi-dimensional Correlation Models: Further research and develop multi-dimensional correlation models to comprehensively capture the complex relationships and interactions between resources. Such models can simultaneously consider the mutual influences among multiple resources, thereby improving prediction accuracy and model interpretability. Exploring the use of deep learning, graph neural networks, or other advanced techniques to construct multi-dimensional correlation models and combining domain knowledge and practical application scenarios for model design and optimization can achieve more precise and reliable resource prediction and management.

## References


[1] Partha Pratim Ray. A survey of IoT cloud platforms. Future Computing and Informatics Journal. 2016; (1-2):35-46. doi: 10.1016/j.fcij.2017.02.001

[2] Thanh Nam Pham, Ming-Fong Tsai, Duc Binh Nguyen, Chyi-Ren Dow, Der-Jiunn Deng. A Cloud-Based Smart-Parking System Based on Internet-of-Things Technologies. IEEE Access. 2015; 3:1581-1591. doi: 10.1109/ACCESS.2015.2477299

[3] Mahfoudh Saeed Al-Asaly, Mohamed A. Bencherif, Ahmed Alsanad, Mohammad Mehedi Hassan. A deep learning-based resource usage prediction model for resource provisioning in an autonomic cloud computing environment. Neural Computing and Applications. 2022; 34(13):10211-10228. doi:10.1007/s00521-021-06665-5

[4] Frank Emmert-Streib, Zhen Yang, Han Feng, Shailesh Tripathi, Matthias Dehmer. An introductory review of deep learning for prediction models with big data. Frontiers in Artificial Intelligence. 2020; Volume 3:4. doi:10.3389/frai.2020.00004

[5] Schuster, Mike. Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing. 1997;45(11):2673-2681. doi:10.1109/78.650093

[6] Rahul Chauhan, Kamal Kumar Ghanshala, Ramesh Chandra Joshi. Convolutional Neural Network (CNN) for Image Detection and Recognition. In:2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC); December 2018; Jalandhar, India. IEEE; 2018. p. 278-282. doi: 10.1109/ICSCCC.2018.8703316

[7] Sumit Saha. A Guide to Convolutional Neural Networks-the ELI5 way. Towards Data Science; 2018. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53/

[8] Kamilya Smagulova, Alex Pappachen James. A survey on LSTM memristive neural network architectures and applications. The European Physical Journal Special Topics. 2019; 228(10):2313-2324. doi: 10.1140/epjst/e2019-900046-x

[9] Sepp Hochreiter, Jürgen Schmidhuber. Long short-term memory. Neural computation. 1997; 9(8):1735-1780. doi: 10.1162/neco.1997.9.8.1735

[10] Shudong Yang, Xueying Yu, Ying Zhou. LSTM and GRU neural network performance comparison study: Taking yelp review dataset as an example. In 2020 International workshop on electronic communication and artificial intelligence (IWECAI); IEEE; 2020. p. 98-101. doi: 10.1109/IWECAI50956.2020.00027

[11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Andreas Müller, Joel Nothman, Gilles Louppe, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Édouard Duchesnay. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011;12:2825-2830. doi: 10.48550/arXiv.1201.0490

[12] Dennis Wackerly, William Mendenhall, Richard L. Scheaffer. Mathematical statistics with applications. Belmont, CA: Thomson Brooks/Cole; 2008.

[13] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. An introduction to statistical learning. New York: springer; 2013.

[14] Trevor Hastie, Robert Tibshirani, Jerome Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. New York: Springer; 2009.