Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 11, No.4, November : 2024

INVESTIGATE THE PERFORMANCE OF VARIOUS DEEP LEARNING METHODS FOR CLOUD WORK LOAD PREDICTION

K.Mohana Rao Research Scholar, Acharya Nagarjuna University, Assistant Professor, dept of CSE, prakasam engineering college, kandukur, autonomous, affiliated to JNTUK, mohanakalavakuri@gmail.com.
Dr. B. Basaveswara Rao Professor, University Computer Centre, Acharya Nagarjuna University, Guntur 522510, India Email: bobbabrao62@gmail.com
Dr.G.Neelima Department of Computer Science & Engineering, Acharya Nagarjuna University, Guntur-522510, India.
Dr.R.Vasantha Department of Computer Science & Engineering, Acharya Nagarjuna University, Guntur-522510, India.

ABSTRACT

Cloud workload prediction is essential for optimizing resource allocation, minimizing energy consumption, reducing SLA violations and ensuring better quality of service in cloud computing. Over the last two decades, several researchers have worked in the domain of cloud workload prediction, proposing traditional, statistical and machine learning methods. However, these methods struggle to capture complex temporal dependences, thus leading to suboptimal predictions. To overcome this problem, deep learning methods have been designed and implemented by researchers. But there is a research gap in determining which method is preferable among the existing deep learning techniques for extracting patterns (Short/Long terms) from time series data.Moreover, the method should be suitable for real-time implementation and demonstrate better performance in identifying patterns of both long-term and short-term dependencies. To address this gap, this study presents a comparison of five prominent state-of-the-art methods for forecasting cloud workloads: Recurrent Neural Network (RNN),Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), Multilayer Perceptron (MLP) and Gated Recurrent Unit (GRU).The models are evaluated on evolving time-series data from cloud monitoring and BitBrains datasets, using performance metrics such as MAE (Mean Absolute Error), RMSE (Root Mean Squared Error), MAPE (Mean Absolute Percentage Error), MSLE (Mean Squared Logarithmic Error) ,Accuracy and Computational time. The study analyzes the prediction performance for different time horizons (1, 5, 10, and 30 minutes), offering insights into the strengths and limitations of each model. The results highlight that while LSTM offers the highest prediction accuracy, it has higher training time and computational cost. In contrast, MLP and CNN exhibit faster training with slightly reduced accuracy. This study offers valuable insights into selecting models for cloud workload prediction, balancing accuracy, training efficiency, and computational complexity.

Keywords:
Cloud workload prediction, Deep learning, RNN, CNN, LSTM, MLP, GRU, Time-series forecasting, Resource management, Cloud computing.

1. INTRODUCTION

With the rapid expansion of cloud computing, efficient workload prediction has become essential for optimizing resource management, energy consumption, and maintaining service quality [1]. Cloud workloads are highly dynamic, varying in both size and frequency over time, which poses significant challenges for effective forecasting and resource provisioning. An accurate workload prediction model allows cloud service providers to efficiently allocate computational resources, reducing operational costs while ensuring smooth service delivery. Traditional statistical methods for time-series forecasting often struggle to capture complex temporal dependencies, leading to suboptimal

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 11, No.4, November : 2024

predictions [2].Therefore, modern deep learning models, which are well-suited for learning intricate patterns from time-series data, have emerged as a promising solution to these challenges [3][4].
This paper provides a comprehensive comparative study of five prominent deep learning models—Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), Multilayer Perceptron (MLP), and Gated Recurrent Unit (GRU)—to evaluate their effectiveness in forecasting cloud workloads [5]. Each of these models offers distinct advantages for time-series prediction. RNNs are capable of capturing sequential dependencies, but they suffer from vanishing gradient problems in long sequences [6]. LSTMs and GRUs address these limitations through advanced gating mechanisms, enhancing the learning of long-term dependencies [7]. CNNs, though traditionally known for image processing, have shown promising results in time-series tasks by identifying localized patterns. MLPs, being lightweight and easy to implement, provide fast predictions despite potential limitations in capturing complex temporal relationships.
The unpredictable and fluctuating nature of workloads in cloud environments necessitates predictive models that not only deliver high accuracy but also strike a balance between training efficiency and computational complexity [8]. This study evaluates these models using real-world cloud monitoring datasets, including the widely used BitBrains dataset. Key performance metrics such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and overall prediction accuracy are used to compare the models. Additionally, predictions are analyzed across multiple time horizons (1, 5, 10, and 30 minutes) to understand the adaptability of each model to short-term and long-term workload fluctuations.
The findings of this study highlight trade-offs between prediction accuracy and computational efficiency. For example, while LSTM offers the most accurate forecasts, it comes with higher computational costs and longer training times [9][10]. On the other hand, CNNs and MLPs provide faster training but with slightly lower accuracy. GRUs, which are computationally lighter than LSTMs, offer a middle ground between performance and efficiency. These insights can guide cloud providers in selecting appropriate models based on their specific operational needs, ensuring the right balance between resource management efficiency and prediction precision.
This work contributes to the growing body of research on workload prediction by providing a detailed comparison of state-of-the-art deep learning models [11]. It also offers practical guidance on model selection, helping cloud operators make informed decisions about deploying predictive systems for real-world cloud environments. Through this analysis, the paper not only demonstrates the advantages and limitations of different models but also establishes a foundation for future research in adaptive workload forecasting methods.
The primary research questions addressed by this study are:
- Which method is most capable of accurate workload prediction among the existing contemporary deep learning methods?
- Which method is suitable for forecasting short-term dependencies versus long-term dependencies considering time origin?
- Does the performance of these deep learning methods reflect unbiased handling of real-world datasets?

The main contributions of this work are as follows:
- A discussion on the requirements and challenges of load prediction in cloud environments.
- An explanation of five contemporary deep learning methods.
- The use of two real-world datasets with different time horizons for model evaluation.
- A detailed comparative study of these models across different time horizons, with insights into their advantages and disadvantages.
- Final conclusions and recommendations provided to guide cloud service providers.

The paper is organized as follows: Section 2 providesa brief overview of various research efforts in this area. The five deep learning methods utilized in the study are explained in section 3. In section 4, the datasets selected for the analysis are described. Section 5outlines the flow process of the methodology. Analysis based on experimental results is presented insection 6. Finally,section 7 concludes the paper and discusses the future scope.

2. LITERATURE REVIEW

Zhang et al [1] proposes a novel deep learning model based on canonical polyadic decomposition, which significantly reduces the number of parameters by converting weight matrices into a more compact format. Their focus is on how to train Deep learning models efficiently because it is challenging due to the large number of parameters involved. They introduced an efficient learning algorithm to optimize the training process. The model is applied to virtual machine workload prediction in cloud environments, with experiments using PlanetLab datasets. Results show that the proposed model outperforms other machine learning approaches in terms of training efficiency and prediction accuracy, highlighting its potential for providing predictive services in industry informatics. However, while the compression of parameters improves efficiency, the complexity of canonical polyadic decomposition may introduce additional computational overhead during the model design and training phase.

Dattatray et al [2] compares four prominent machine learning algorithms—LSTM, CNN, RNN, and MLP—for stock value prediction, using historical stock price data and metrics like MAE, RMSE, and precision. LSTM outperforms the others, followed by CNN, RNN, and MLP, but they note that the best algorithm depends on the specific data and desired accuracy. The LSTM model for stock prediction involves data preprocessing, feature selection, model construction, and training with hyperparameter tuning. Performance is evaluated using metrics like MAE and MSE, and predictions are made on new data, with continuous monitoring and adjustments for market changes. But their study lacks consideration of external factors like economic events that influence stock prices, limiting the model's real-world applicability. Additionally, the heavy reliance on hyperparameter tuning may reduce the generalizability of the results across different markets and time periods.

Kamal et al [3] proposes a novel solution, Federated Cloud Workload Prediction with Deep Q-Learning (FEDQWP), aimed at improving VM placement, energy efficiency, and SLA adherence in Federated Cloud Computing (FCC) environments. By leveraging deep Q-learning, the model efficiently extracts patterns from real-world workloads to optimize resource allocation for Cloud Service Providers (CSPs). The FEDQWP outperforms existing algorithms in key metrics, achieving superior CPU utilization (median value of 29.02), faster migration times (average 0.31 units), more tasks completed (699 on average), lower energy consumption (1.85 kWh), and fewer SLA violations (0.03). The main merit is its comprehensive approach, addressing both performance and energy efficiency, which enhances scalability and SLA preservation for CSPs. However, they have limited focus on the practical implementation challenges of deep learning models in real-world FCC environments and the potential complexity of integrating the model into existing cloud infrastructures. Additionally, the study does not deeply explore the cost implications or adaptation to diverse workloads.

Zaakki et al [4] analyses the effectiveness of DL models using data from the Parallel Workloads Archive, highlighting LSTM as the best-performing model. The analysis employs MAE and RMSE as performance metrics and emphasizes the importance of robust, unbiased data sources. The paper also addresses the gap in existing literature on DL-based workload prediction and offers a detailed overview of resource management, prediction models, and error metrics in cloud environments. But they have lack of attention to data quality and uniformity in existing works on workload prediction using deep learning (DL).

Dozdar et al. [5] provide a review on Deep sequential (DS) models that are extensively used for time series forecasting due to their ability to learn hidden patterns and retain information from previous time points. Unlike traditional statistical models, DS models like artificial neural networks (ANN), long short-term memory (LSTM), and temporal-convolutional neural networks (TCNN) excel in forecasting by capturing complex temporal dynamics. This review presents a comprehensive comparison of these models across various domains, focusing on their structure, activation functions, optimizers, and performance. Among the models, LSTM, especially in hybrid forms, is found to deliver the most accurate predictions. Additionally, the paper discusses challenges and future perspectives for the development of deep sequential models.

Van-Thang et al [6]. Provides a comparative study of Deep Learning models for predicting stock prices. In this they deploy six deep learning models (i.e., MLP, CNN, RNN, LSTM, GRU, and AE) to predict the closing price, one day ahead, of 20 different companies (i.e. 5 groups of 4) in the S&P 500 markets over the 7-years range (Jan 2015- August 2022). It helps us to deepen our understanding of how to use deep learning models in financial markets.

Mortezapouret al [7]. offers a comprehensive survey of various DL models, including Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Generative Models, Deep Reinforcement Learning (DRL), and Deep Transfer Learning. It analyzes the structure, applications, and limitations of each model while comparing six key DL models—CNN, Simple RNN, LSTM, Bidirectional LSTM, GRU, and Bidirectional GRU—using datasets such as IMDB, ARAS, and Fruit-360. The comparison highlights the performance of these models across different problem domains.

Wanget al. [8] focuses on LSTM (Long Short-Term Memory) for multi-step time series prediction. It compares two approaches: multi-step input and seq2vec, offering insights into their effectiveness. The study begins with an introduction to LSTM theory, followed by its application to a selected dataset. Results demonstrate that LSTM can effectively predict seasonal time series data across multiple steps. Lastly, the paper highlights the strengths and limitations of both methods.

W. Li and K. L. E. Law.[9] explores methodologies for modeling time series data and highlights advancements in time series forecasting using deep learning models. It discusses the challenges of constructing reliable prediction models due to the complexity and scale of time series data. The study reviews commonly used datasets, evaluation metrics, and essential architectures for forecasting. Different deep learning models are assessed on the same dataset using identical hardware to compare their performance. The SCINet model achieves the highest accuracy with the ETT energy dataset, demonstrating the relationship between model design and performance. Finally, the paper offers insights into future research directions in deep learning for time series forecasting.

S. Bansal and M. Kumar [10]. addresses the challenge of unpredictable user demand in cloud computing, which can degrade service performance and reduce revenues. It emphasizes the importance of accurately predicting mixed workloads to optimize resource allocation, job scheduling, and server efficiency. The proposed approach uses a deep learning-based model that combines neural networks to analyze historical workload data and forecast future demands. Real-world datasets are utilized to evaluate the model, demonstrating its effectiveness in accurately predicting workloads. The results show that the proposed solution enhances system efficiency and user satisfaction.

Workload prediction is vital for resource provisioning in cloud computing and its extensions, such as serverless and edge computing, to ensure efficient resource allocation. Accurate workload forecasting enhances performance, reduces costs, optimizes energy use, and maintains high-quality service. Yekta et al [11] provides a comprehensive review of cutting-edge machine learning (ML) and deep learning (DL) algorithms used in workload prediction across cloud and edge computing. It compares selected studies based on methods, predicted factors, accuracy metrics, and datasets, and presents a table to summarize common advantages and disadvantages. They concluded by discussing current challenges and future research directions.

3. METHODS

The contemporary deep learning methods are adopted for this comparative study are briefly discussed below.

3.1 RNN

Recurrent Neural Networks (RNNs) are ideal for processing sequential data but face challenges with vanishing gradients during long sequences. The hidden state at time **t** is updated using the following equation:

$h_t=\sigma(W_h\cdot[h_{t-1},x_t]+b_h)$

□x_t= input at time t
□h_{t-1} = hidden state from the previous step
□W_h = weight matrix
□b_h= bias vector
□σ= activation function (e.g., tanh or ReLU)

While RNNs are effective for short-term temporal patterns, they struggle to retain information over longer sequences, leading to performance degradation in complex cloud workload prediction.

3.2. CNN

Convolutional Neural Networks (CNNs) excel at detecting local patterns in time-series data through convolutional filters. However, they struggle with long-term dependencies. A typical convolution operation is expressed as:

$\text{Conv}\ (x_{t,}W)=W * x_t+b$

□x_t = input at time t
□W = convolutional filter (weight matrix)
□b = bias term

After the convolution, max pooling reduces dimensionality and extracts the most significant features:

$\text{pooling}\ (x_t)=\max(x_t)$

While CNNs are efficient for capturing local patterns in workloads (such as CPU and memory usage spikes), they are often paired with LSTMs or GRUs to handle longer-term dependencies.

3.3. LSTM

Long Short-Term Memory (LSTM) networks address the vanishing gradient problem through gated mechanisms that control the flow of information. Three key gates are used:

Forget Gate

$f_t=\sigma(W_f\cdot[h_{t-1},x_t]+b_f)$

Input Gate

$i_t=\sigma(W_i\cdot[h_{t-1},x_t]+b_i)$

Output Gate

$o_t=\sigma(W_o\cdot[h_{t-1},x_t]+b_o)$

The **cell state** is updated as:

$C_t=f_t\cdot C_{t-1}+i_t\cdot\tanh(W_C\cdot[h_{t-1},x_t]+b_C)$

□C_t = current cell state
□C_{t-1} = previous cell state

LSTMs can retain both recent trends and long-term patterns, making them suitable for workloads with complex temporal dependencies.

3.4. GRU

Gated Recurrent Units (GRUs) simplify LSTM architectures by combining the forget and input gates into a reset gate. This makes them more efficient for real-time prediction tasks.

Reset Gate

$r_t=\sigma(W_r * [h_{t-1},x_t]+b_r)$

Update Gate

$z_t=\sigma(W_z * [h_{t-1},x_t]+b_z)$

The **hidden state** is updated as:

$h_t=z_t * h_{t-1}+(1-z_t) * \tanh(W_h*[r_t * h_{t-1},x_t]+b_h)$

GRUs provide a good trade-off between computational efficiency and the ability to model temporal dependencies, making them suitable for low-latency cloud prediction.

3.5. MLP

Multi-Layer Perceptrons (MLPs) are feedforward networks that work well for non-sequential data. Each layer applies a linear transformation followed by a non-linear activation function. For example: $y=\mathrm{ReLU}\,(W_1 \cdot x+b_1)\cdot W_2+b_2$

- W_1 and W_2 = weight matrices
- b_1 and b_2 = biases
- ReLU = max (0, x)

While MLPs are useful as a baseline model, they struggle to learn complex temporal dependencies, making them less effective for time-series predictions.

Each model architecture provides distinct advantages:

- **RNNs** capture short-term dependencies but struggle with long-term memory.
- **CNNs** are efficient in detecting local patterns but need additional layers or models for long-term forecasting.
- **LSTMs** handle both short- and long-term dependencies, achieving high accuracy for complex workloads.
- **GRUs** offer a lighter, faster alternative to LSTMs for real-time predictions.
- **MLPs** are simple and efficient but limited in learning temporal patterns.

By selecting the appropriate model or combining them (e.g., CNN-LSTM hybrids), cloud systems can achieve accurate and scalable workload predictions to improve resource management and performance.

The training of different models (CNN, RNN, LSTM, GRU, and MLP) is based on historical workload data (e.g., CPU usage and memory consumption) to generate predictions for various time horizons, ranging from 1 minute to 30 minutes ahead. The mean squared error (MSE) serves as the primary loss function, which is minimized using backpropagation through time (BPTT)**.** For optimization, the Adam optimizer is employed to update model weights efficiently.

The gradients of the loss function Lwith respect to the model parameters W are computed iteratively as follows:

$$\frac{\partial L}{\partial W}=\sum_t \frac{\partial L}{\partial ht}\cdot\frac{\partial ht}{\partial W}$$

Each model is trained over multiple epochs to minimize the error, ensuring good generalization and optimal performance for cloud workloads. The analysis of the prediction performance and efficiency of each model across different time horizons, based on results from two datasets: Application crash rate from the Cloud Monitoring Dataset and CPU Usage% from the BitBrains Dataset is performed.

4. DATASET DESCRIPTION

Cloud monitoring datasets, like the one released by Microsoft, capture detailed telemetry signals from various cloud-based services and systems to monitor metrics such as crash rates. The **Application Crash Rate dataset** from cloud monitoring provides insights into the stability and performance of applications deployed in cloud environments. It includes metrics like the frequency of crashes, affected application versions, and user sessions impacted. This dataset is valuable for identifying patterns, diagnosing root causes, and proactively improving system reliability. By analyzing this data, organizations can enhance their application's resilience and user experience.This dataset is downloaded from github[14]. The application crash rate app2-09 has 1029 number of records.

Bitbrains is another popular datasetfrom the fast storage of kaggle[15], that consists of 1250 VMs. Data is taken from VM3 and VM22 of fast storage, each of which consist 8634 and 8635 records respectively. It is used for cloud resource prediction containing real-world workload traces. This dataset covers various performance metrics, including CPU utilization**,** memory usage**,** and network traffic (including ingress and egress rates), recorded across thousands of servers over extended periods. Bitbrains datasets are typically used by several researchers to analyzeresource management, predictive workload scheduling, burst patterns and periodic workloads**,** which are crucial for efficient cloud resource allocation. These datasets are widely used in academic and industrial research to design models for scaling virtual machines (VMs) dynamically and optimizing resource consumption.

5. METHODOLOGY

In this section, comparative analysis of five deep learning models forcloud workload prediction is carried out.It consists of various phases i.e., data preparation, data preprocessing, different model configurations and experimental setup. The phases are explained below and the flow process is depicted in Fig.1.

Data Preparation

- **Datasets:** The study leverages two time-series datasets, the *Cloud Monitoring dataset* and the *Bitbrains dataset*, chosen for their relevance in workload prediction tasks.
- **Data Splitting:** Each dataset was divided into three distinct sets:
  - **Training Set:** 70% of the data, used to train the models
  - **Validation Set:** 15% of the data, used to fine-tune model parameters
  - **Test Set:** 15% of the data, reserved for evaluating final model performance

Data Preprocessing

- **Scaling:** All data values were normalized to a [0, 1] range, which promotes training stability and accelerates model convergence.
- **Sequence Creation:** To capture time dependencies, a sliding window approach was applied to generate fixed-length input sequences for the models.

Each of the proposed models—RNN, CNN, LSTM, MLP, and GRU—offers distinct benefits for predicting cloud workloads based on metrics like CPU usage, memory consumption, disk I/O, and network traffic**.** These models are trained to predict future workloads by analyzing historical data, with the goal of enhancing resource management in cloud systems. The choice of the models emphasizes capturing both local patterns and temporal dependencies, which vary across different time-series data.

Model Configurations and Experimental setup

The five models are carefully constructed and fine-tuned by adjusting key hyper parameters, such as learning rate, dropout, epochs, and batch sizes, to balance predictive accuracy with training efficiency. Eacharchitecture is tailored to leverage specific strengths—like memory retention in RNNs or pattern extraction in CNNs—while ensuring robust generalization through optimization strategies. This systematic tuning process aims to enhance model performance while mitigating risks of overfitting and underfitting across varying workload scenarios.

RNN Based Model

- **Input:**Time-series data representing application crash rates (monitoring dataset).
- **RNN layers: T**wo simple RNN layers with 64, and 32 Units.
- **Dense layers:** 1 layer with 32 units (Hidden layer), 1 layer with 30 units (Output layer).
- **Learning rate:** 0.001.
- **Batch size:** 64.
- **Epochs:** 100.

CNN Based Model

- **Input:** Time-series data representing application crash rates (monitoring dataset).
- **CNN layers:** 2 layers with 64 and 128 filters, kernel size 3.
- **Pooling layers:** 2 max pooling layers with pool size 2.
- **Flatten Layer:** Flattens the output from the last convolution layer for the dense layers.
- **Dense layer:** 1 layer with 64 units (Hidden layer), 1layer with 30 units (output layer).
- **Learning rate:** 0.001.
- **Batch size:** 64.
- **Epochs:** 100.

LSTM Based Model

- **Input:**Time-series data representing application crash rates rates (monitoring dataset).
- **LSTM Layers:** 2 layers with 64 and 32 units.
- **Dropout layers:** 2 layers with 20% dropout to prevent overfitting.
- **Dense Layers:** 1 layer with 32 units (Hidden layer), 1 layer with 30 units (output layer).
- **Learning Rate:**0.0005.
- **Batch Size:**64.
- **Epochs:**20 (with early stopping to monitor validation loss, patience of 10).

MLP Based Model

- **Input:** Time-series data representing application crash rates (monitoring dataset).
- **Hidden Layers:**
  - Dense Layer 1:64 neurons withReLU activation to Learns patterns from the input data.
  - Dense Layer 2: 32 neurons with ReLU activation to Extract deeper relationships in the data.
  - Dense Layer 3: 16 neurons with ReLU activation to Provide further feature refinement.
- **Output Layer:**
  - Dense Layer with 30 neuronsand Linear activation
- **Learning rate:** 0.001.
- **Batch size:** 64.
- **Epochs:** 100.

GRU Based Model

- **Input:** Time-series data representing application crash rates (monitoring dataset).
- **GRU Layers:** 2 layers with 64 and 32 units, respectively.
- **Dropout Layers:** 2 layers with 20% dropout to prevent overfitting.
- **Dense Layers:** 1 layer with 32 units (hidden layer), 1 layer with 30 units (output layer).
- **Learning Rate:** 0.0005.
- **Batch Size:** 64.
- **Epochs:** 100

Experimental Setup

These experiments are conducted on Windows 11, 12th Gen Intel Core i5-12450H, 2.00 GHz, 16 GB RAM with the following hardware and software specifications:

- Hardware Configuration
  - Processor: 12th Gen Intel Core i5-12450H, 2.00 GHz
  - Memory: 16 GB RAM
- Software Environment
  - Programming Language: Python, Version 3.9 or later
  - Libraries: The experiments were conducted using TensorFlow/Keras for deep learning model development. Additional support libraries included:
    - NumPy, pandas, and scikit-learn for data preprocessing and manipulation
    - Matplotlib for data visualization

This setup enabled a systematic and efficient evaluation of multiple deep learning models (RNN, CNN, LSTM, MLP, and GRU) on both datasets. The primary goal was to assess the predictive accuracy of each model while maintaining computational efficiency on a standard hardware platform.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 11, No.4, November : 2024

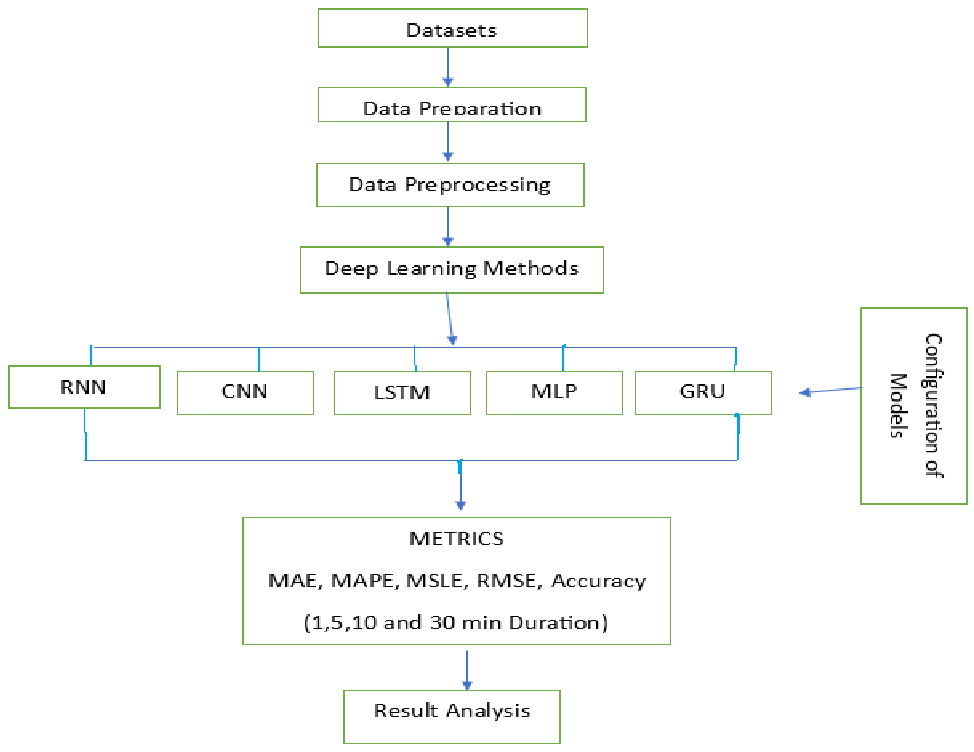

Fig 1: Flow Process of the comparative study of Deep Learning Methods for Cloud Work Load Prediction**.**

The above figure presents a comparative analysis of various deep learning models—RNN, CNN, LSTM, GRU, and MLP. The models were evaluated using cloud monitoring datasets representing application crash rates and CPU usage across multiple time horizons (1, 5, 10, and 30 minutes ahead). The experiments are conducted to assess each model's efficiency and performance using key metrics likeMAE, RMSE, MAPE, MSLE and Accuracy. Additionally, training time and prediction time were measured to determine the trade-offs between model accuracy and computational costs.

6. RESULTS AND ANALYSIS

The models are evaluated to elevate the efficiency of the models for prediction of workload through various metrics with different time horizons,1,5,10 and 30-minutes. The numerical results demonstrate the short-term and long-term prediction capabilities and responsiveness, which is critical for real-time systems. The following tables are depicted and graphs are presented basing on the numerical evaluations.

Table: 1 The Accuracies of predicting application crash rate-9 from cloud Monitoring dataset

Method	1-min	5-min	10-min	30-min
CNN	64.80%	63.94%	61.75%	64.82%
RNN	66.89%	66.40%	66.60%	67.04%
LSTM	75.93%	76.61%	76.89%	77.25%
GRU	66.88%	67.01%	66.58%	66.91%
MLP	69.52%	67.08%	68.64%	66.99%

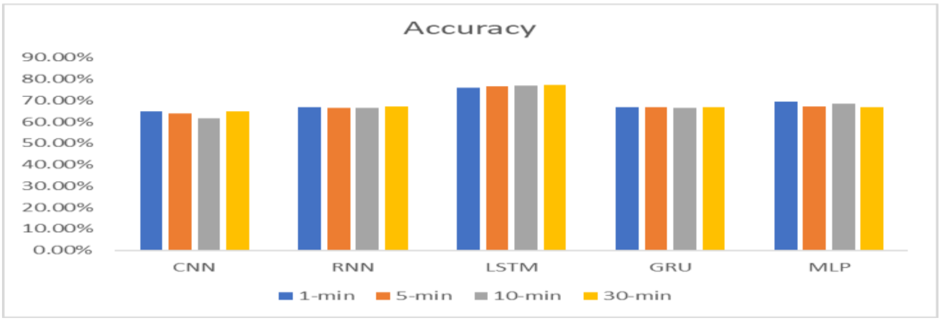

Figure2: The Effect of Accuracies for various deep learning methods of cloud monitoring dataset

From the above table, it is concluded that the accuracy of crash rate prediction is significantly influenced by changes in the time horizon. The differences in accuracies across the time horizons are 3.07 for CNN, 0.64 for RNN, 1.32 for LSTM, 0.33 for GRU, and 2.53 for MLP. GRU exhibits the least variation in accuracy, indicating it is highly consistent for both short-term and long-term predictions. In contrast, CNN shows the highest variation, though it still maintains high accuracy overall. These results suggest that while all methods perform well, GRU provides the most stable and reliable performance across different forecasting periods.

Table 2: MAE of predicting application crash rate-09 from cloud monitoring dataset.

Method	1-min	5-min	10-min	30-min
CNN	0.007	0.007	0.008	0.007
RNN	0.006	0.006	0.006	0.006
LSTM	0.008	0.007	0.008	0.008
GRU	0.005	0.006	0.006	0.005
MLP	0.006	0.006	0.006	0.007

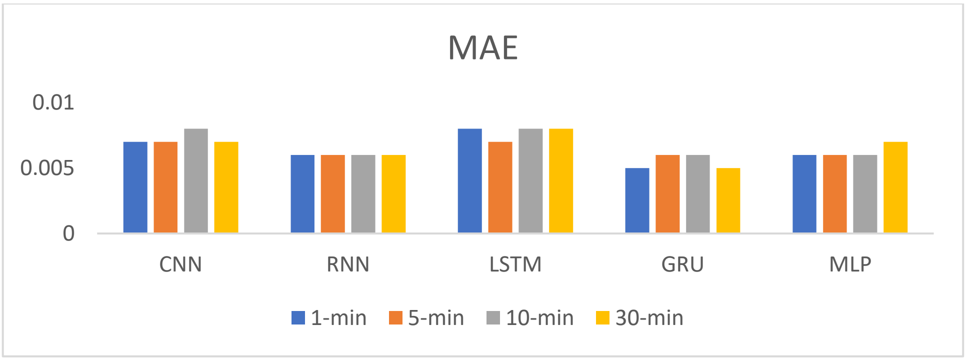

Figure 3: The Effect of MAE for various deep learning methods of cloud monitoring dataset

Table 2 shows that GRU achieves the lowest MAE of 0.001 across all time horizons, demonstrating superior accuracy and stability. RNN follows closely with consistent MAE values of 0.006, making it a reliable alternative. CNN and LSTM show slightly higher errors (0.007–0.008), with CNN displaying minor inconsistency at the 10-minute horizon. MLP performs similarly to RNN but shows a slight increase in error (0.007) for the 30-minute horizon. Overall, GRU stands out as the most accurate and stable method, with RNN offering comparable reliability.

Table 3: RMSE of predicting application crash rate-09 from cloud monitoring dataset.

Method	1-min	5-min	10-min	30-min
CNN	0.01	0.01	0.011	0.01
RNN	0.009	0.01	0.01	0.01
LSTM	0.012	0.011	0.011	0.012
GRU	0.007	0.008	0.008	0.008

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 11, No.4, November : 2024

MLP	0.008	0.009	0.009	0.009

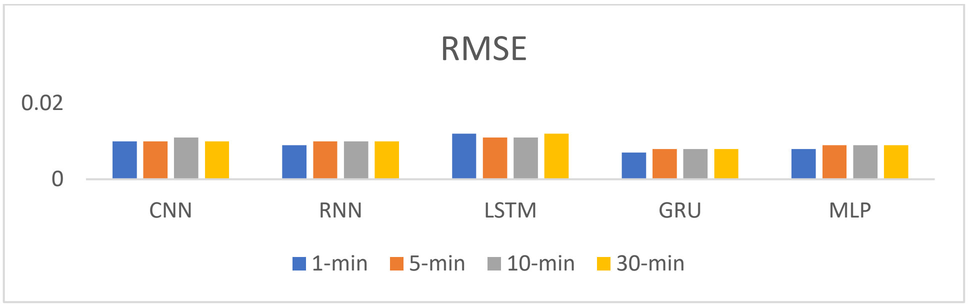

Figure 4: The Effect of RMSE for various deep learning methods of cloud monitoring dataset
Table 3 highlights the RMSE of various methods for predicting crash rate across different time horizons. GRU achieves the lowest RMSE values (0.007–0.008), indicating its high accuracy and stability over both short and long-term predictions. RNN performs well with consistent RMSE values (0.009–0.01) but slightly trails behind GRU. CNN maintains competitive RMSE values (0.01–0.011), though it shows a slight increase at the 10-minute horizon (0.011). LSTM records the highest RMSE (0.011–0.012), suggesting lower precision compared to other methods. MLP exhibits moderate RMSE (0.008–0.009) with stable performance across all horizons but is less accurate than GRU. Overall, GRU demonstrates the best performance, combining low RMSE with consistent predictions, while RNN and CNN provide competitive alternatives.

Table 4: MAPE of predicting application crash rate -09 from cloud monitoring dataset.

Method	1-min	5-min	10-min	30-min
CNN	0.322	0.313	0.303	0.306
RNN	0.264	0.267	0.263	0.266
LSTM	0.349	0.316	0.297	0.281
GRU	0.227	0.234	0.237	0.228
MLP	0.25	0.282	0.267	0.286

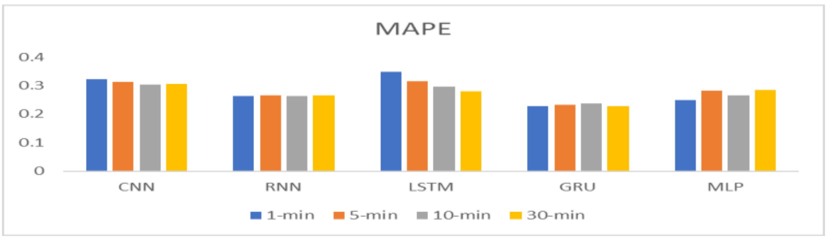

Figure 5: The Effect of MAPE for various deep learning methods of cloud monitoring dataset
Table 4 compares the MAPE of different methods for predicting the crash rate across various time horizons. GRU achieves the lowest MAPE (0.227–0.237), showcasing its superior accuracy and consistent performance. RNN also performs well, with MAPE values ranging from 0.263 to 0.267, indicating reliability across all horizons. CNN demonstrates slightly higher MAPE (0.303–0.322), with improved accuracy for longer-term predictions. LSTM shows the highest MAPE variation (0.281–0.349), with relatively lower accuracy for shorter horizons but improved performance as the horizon increases. MLP exhibits moderate MAPE values (0.25–0.286), maintaining consistent performance but not matching GRU or RNN. Overall, GRU stands out as the most accurate and reliable method for minimizing percentage error, followed closely by RNN.

Table 5: MSLE of predicting application crash rate-09 from cloud monitoring dataset.

Method	1-min	5-min	10-min	30-min
CNN	0.00010	0.00011	0.00011	0.00011
RNN	0.00008	0.00009	0.00008	0.00008

LSTM	0.00010	0.00012	0.00010	0.00015
GRU	0.00005	0.00006	0.00007	0.00007
MLP	0.00006	0.00007	0.00007	0.00008

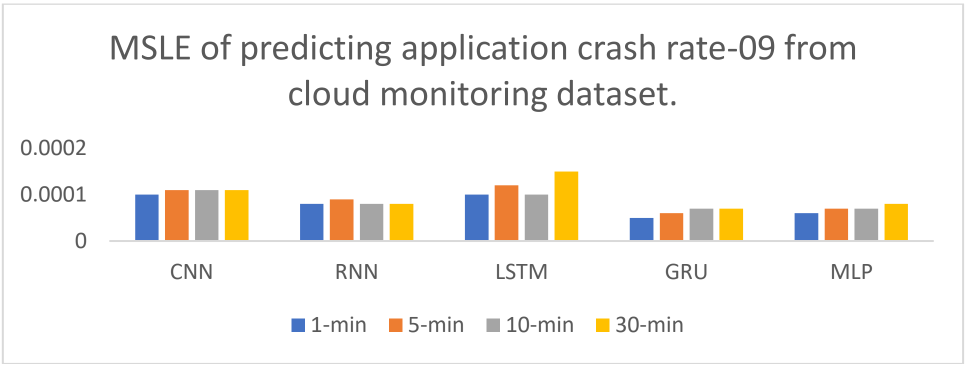

Figure 6: The Effect of MSLE for various deep learning methods of cloud monitoring dataset

Table 5 presents the MSLE for predicting the applicationcrash rate across different time horizons. GRU exhibiting consistently low error rates (0.00005 to 0.00007), making it the most stable and accurate. MLP and RNN show slightly higher MSLE values (0.00006–0.00008), maintaining stable performance but with minor variations across horizons. LSTM records the highest MSLE (0.00010–0.00015), indicating relatively lower precision, particularly for the 30-minute horizon (0.00015). Overall, CNN shows minimal variance but higher errors (0.00010 to 0.00011), compared to GRU, but it has good stability in diff time horizons.

Table 6: The Accuracy of CPU Usage% prediction on VM-22, Fast Storage of BitBrains Dataset

Method	1 min ahead	5 min ahead	10 min ahead	30 min ahead
CNN	89.67%	92.77%	91.99%	92.30%
RNN	91.96%	93.90%	92.78%	92.24%
LSTM	93.68%	94.00%	93.80%	94.96%
GRU	94.02%	92.95%	93.30%	94.01%
MLP	93.09%	92.04%	90.35%	93.92%

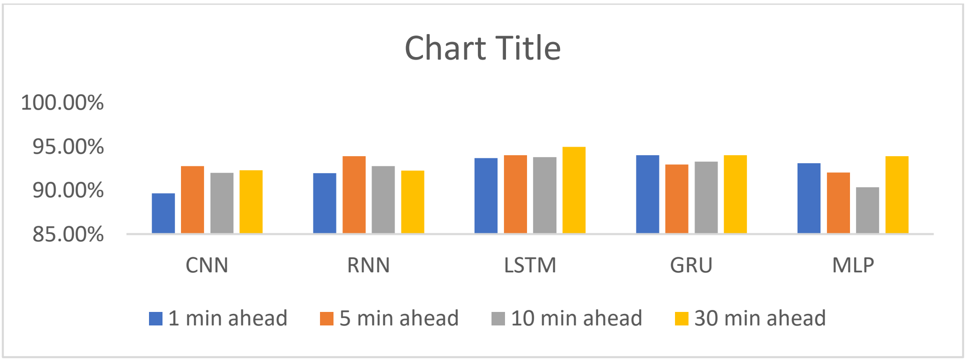

Figure 7: The Effect of Accuracies for various deep learning methods of BB VM-22 dataset

From the data, we can conclude that GRU exhibits the most consistent performance across different time horizons, with a minimal variation in accuracy of 1.07, indicating reliability for both short-term and long-term predictions. LSTM also demonstrates high accuracy with low variation of 1.28, showcasing robustness across different forecast periods. RNN maintains a consistent performance with moderate stability of 1.94. In contrast, CNN, despite being relatively stable, shows a slightly higher variation of 3.10. MLP has the highest variation in accuracy of 3.57 indicating a greater sensitivity to changes in the forecasting horizon. Overall, GRU and LSTM are the most reliable methods, while MLP exhibits the most variability.

Table 7: The MAE of CPU Usage% prediction on VM-22, Fast Storage of BitBrains Dataset

Method	1 min ahead	5 min ahead	10 min ahead	30 min ahead
CNN	0.147	0.127	0.126	0.125
RNN	0.115	0.113	0.112	0.112
LSTM	0.111	0.11	0.112	0.115
GRU	0.112	0.117	0.115	0.112
MLP	0.111	0.111	0.121	0.108

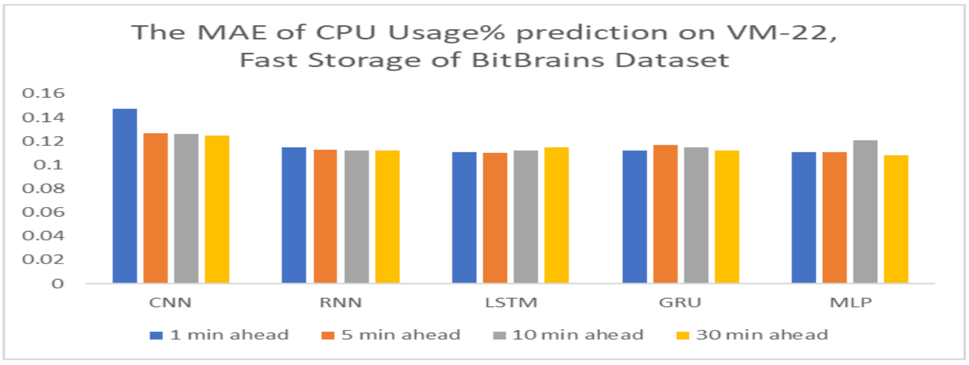

Figure 8: The Effect of MAE for various deep learning methods of BB VM-22 dataset

From the data, we can conclude that RNN exhibits the best overall performance in terms of mean absolute error (MAE) across all time horizons, with values ranging from 0.112 to 0.115, indicating strong reliability for CPU usage prediction. LSTM closely follows, demonstrating a consistent MAE from 0.111 at 1 minute to 0.115 at 30 minutes, reflecting stability in its predictions. GRU also shows reliable performance, with MAE values ranging from 0.112 to 0.117, indicating minimal variation. CNN displays slightly higher MAE values, varying from 0.125 to 0.147, which suggests a moderate level of accuracy. MLP presents some variability, with its MAE starting at 0.111, peaking at 0.121 at 10 minutes, and decreasing to 0.108 at 30 minutes. Overall, RNN and LSTM are the most reliable methods for CPU usage prediction, while CNN shows the highest MAE values.

Table 8: The RMSE of CPU Usage% prediction on VM-22, Fast Storage of BitBrains Dataset

Method	1 min ahead	5 min ahead	10 min ahead	30 min ahead
CNN	0.257	0.236	0.235	0.236
RNN	0.225	0.224	0.222	0.22
LSTM	0.22	0.221	0.222	0.226
GRU	0.221	0.222	0.221	0.222
MLP	0.221	0.218	0.247	0.22

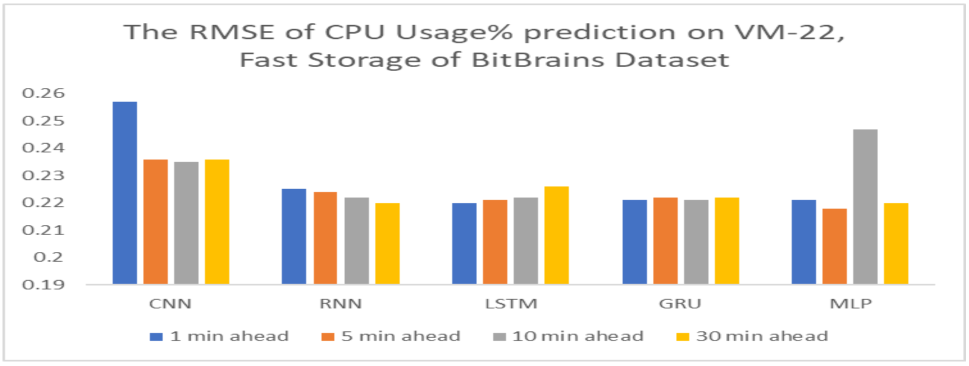

Figure 9: The Effect of RMSE for various deep learning methods of BB VM-22 dataset

From the data, we can conclude that RNN demonstrates the best overall performance in terms of root mean square error (RMSE) across all time horizons, with values ranging from 0.220 to 0.225,

Industrial Engineering Journal

ISSN: 0970-2555

Volume : 53, Issue 11, No.4, November : 2024

indicating strong reliability for CPU usage prediction. LSTM follows closely, showing consistent RMSE values from 0.220 at 1 minute to 0.226 at 30 minutes, reflecting stability in its predictions. GRU also exhibits solid performance, with RMSE values between 0.221 and 0.222, indicating minimal variation across different time horizons. CNN presents slightly higher RMSE values, ranging from 0.235 to 0.257, suggesting a moderate level of accuracy. MLP shows some variability, with its RMSE decreasing from 0.221 at 1 minute to 0.218 at 5 minutes, rising to 0.247 at 10 minutes, and then settling back at 0.220 at 30 minutes. Overall, RNN and LSTM are the most reliable methods for CPU usage prediction, while CNN exhibits the highest RMSE values.

Table 9: The MAPE of CPU Usage% prediction on VM-22, Fast Storage of BitBrains Dataset

Method	1 min ahead	5 min ahead	10 min ahead	30 min ahead
CNN	0.219	0.182	0.183	0.181
RNN	0.17	0.162	0.165	0.167
LSTM	0.162	0.156	0.164	0.164
GRU	0.162	0.176	0.171	0.162
MLP	0.162	0.165	0.176	0.152

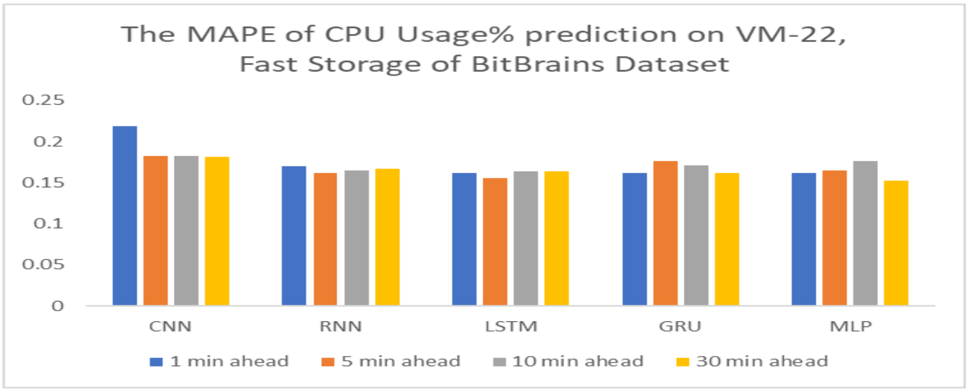

Figure 10: The Effect of MAPE for various deep learning methods of BB VM-22 dataset

From the data, we can conclude that LSTM demonstrates the best overall performance in terms of mean absolute percentage error (MAPE) across all time horizons, with values ranging from 0.156 to 0.164, indicating strong reliability for CPU usage prediction. RNN follows closely behind, showing consistent MAPE values from 0.170 at 1 minute to 0.167 at 30 minutes, reflecting stable predictions. GRU also exhibits reliable performance, with MAPE values between 0.162 and 0.176, indicating minimal variation. CNN presents slightly higher MAPE values, ranging from 0.181 to 0.219, suggesting a moderate level of accuracy. MLP shows some variability in performance, with its MAPE starting at 0.162, peaking at 0.176 at 10 minutes, and decreasing to 0.152 at 30 minutes. Overall, LSTM and RNN are the most reliable methods for CPU usage prediction, while CNN displays the highest MAPE values.

Table 10: The MSLE of CPU Usage% prediction on VM-22, Fast Storage of BitBrains Dataset

Method	1 min ahead	5 min ahead	10 min ahead	30 min ahead
CNN	0.016	0.013	0.013	0.013
RNN	0.011	0.011	0.011	0.01
LSTM	0.01	0.011	0.011	0.011
GRU	0.011	0.011	0.011	0.011
MLP	0.011	0.01	0.013	0.01

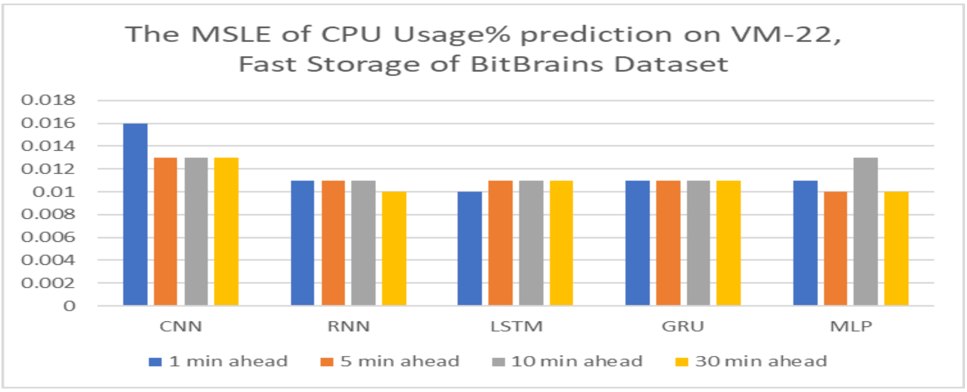

Figure 11: The Effect of MSLE for various deep learning methods of BB VM-22 dataset

From the data, we can conclude that RNN and GRU exhibit the best overall performance in terms of mean squared logarithmic error (MSLE) across all time horizons, consistently maintaining low values between 0.010 and 0.011, indicating strong reliability for CPU usage prediction. LSTM closely follows, demonstrating similarly low MSLE values ranging from 0.010 to 0.011, reflecting stability in its predictions. CNN presents slightly higher MSLE values, varying from 0.013 to 0.016, suggesting a moderate level of accuracy compared to the other methods. MLP shows some variability in its performance, with MSLE values fluctuating between 0.010 and 0.013. Overall, RNN and GRU are the most reliable methods for CPU usage prediction, while CNN displays the highest MSLE values.

From this we can conclude that The MLP is the fastest in both training and prediction, making it suitable for quick, straightforward tasks. CNN offers a good balance with moderate training and very low prediction time, suitable for tasks that need some level of complexity. LSTM, with the highest training time, is better suited for tasks that require deep sequence learning, and the GRU serves as a middle option, balancing complexity and efficiency.

Table 11: The Accuracy of CPU Usage% prediction on VM-3, Fast Storage of BitBrains Dataset

Method	1-min ahead	5-min ahead	10-min ahead	30-min ahead
CNN	96.53%	96.51%	96.44%	96.65%
RNN	96.54%	96.91%	96.99%	97.26%
LSTM	97.01%	97.20%	97.34%	98.35%
GRU	97.24%	97.45%	97.55%	97.61%
MLP	96.82%	96.73%	96.59%	96.72%

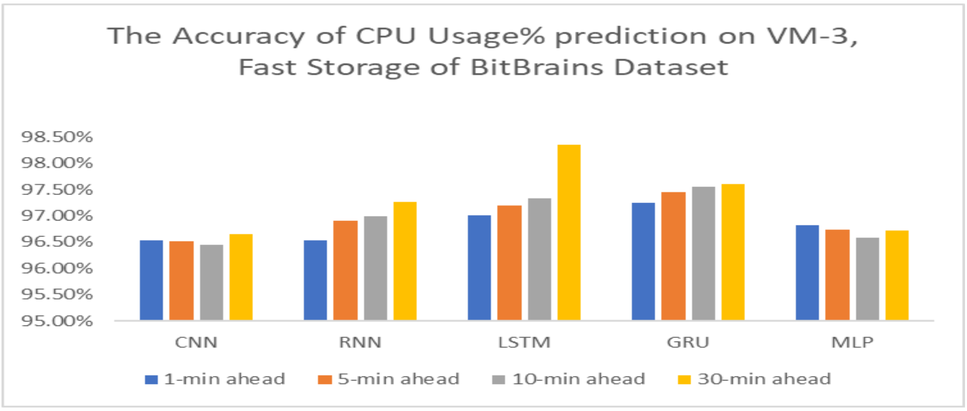

Figure 12: The Effect of Accuracy for various deep learning methods of BB VM-3 dataset

From the above data, it is observed that GRU shows the highest consistency and accuracy across all time horizons, ranging from 97.24% to 97.61%. LSTM also performs exceptionally well, with accuracies increasing from 97.01% to 98.35%, indicating strong reliability for both short-term and long-term predictions. RNN maintains high accuracy with a slight increase over time, ranging from

96.54% to 97.26%. CNN remains stable, with accuracies between 96.44% and 96.65%. MLP, while slightly lower, still shows consistent performance, with accuracies ranging from 96.59% to 96.82%. Overall, GRU and LSTM demonstrate superior performance and consistency, making them the most reliable methods for CPU usage prediction in this context.

Table 12: The MAE of CPU Usage% prediction on VM-3, Fast Storage of BitBrains Dataset

Method	1-min ahead	5-min ahead	10-min ahead	30-min ahead
CNN	0.114	0.119	0.113	0.12
RNN	0.094	0.104	0.105	0.115
LSTM	0.177	0.183	0.191	0.226
GRU	0.108	0.11	0.122	0.13
MLP	0.104	0.107	0.109	0.116

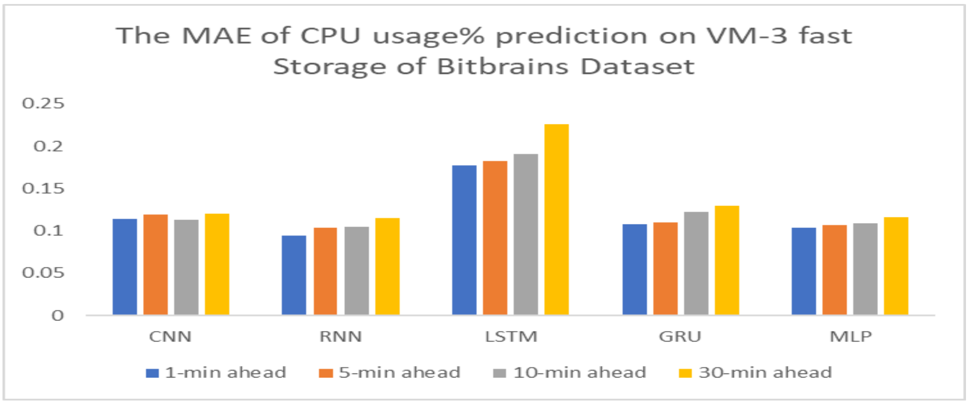

Figure 13: The Effect of MAE for various deep learning methods of BB VM-3 dataset

From the data, we can conclude that RNN exhibits the best overall performance in terms of mean absolute error (MAE) for CPU usage prediction across all time horizons, with values ranging from 0.094 to 0.115, indicating strong reliability and lower prediction error. MLP closely follows, demonstrating consistent MAE values between 0.104 and 0.116, suggesting moderate accuracy. CNN shows slightly higher MAE values, varying from 0.113 to 0.120, indicating a bit more error in its predictions. GRU also performs reasonably well, with MAE ranging from 0.108 to 0.130, reflecting stability, though it exhibits slightly higher errors than RNN and MLP. In contrast, LSTM has the highest MAE among the methods, with values ranging from 0.177 to 0.226, indicating greater variability and less accuracy in CPU usage predictions. Overall, RNN and MLP are the most reliable methods, while LSTM exhibits the highest MAE and, therefore, the least accuracy in predictions.

Table 13: The RMSE of CPU Usage% prediction on VM-3, Fast Storage of BitBrains Dataset

Method	1-min ahead	5-min ahead	10-min ahead	30-min ahead
CNN	0.353	0.356	0.341	0.343
RNN	0.232	0.268	0.3	0.327
LSTM	0.305	0.34	0.376	0.482
GRU	0.276	0.292	0.338	0.374
MLP	0.233	0.263	0.308	0.319

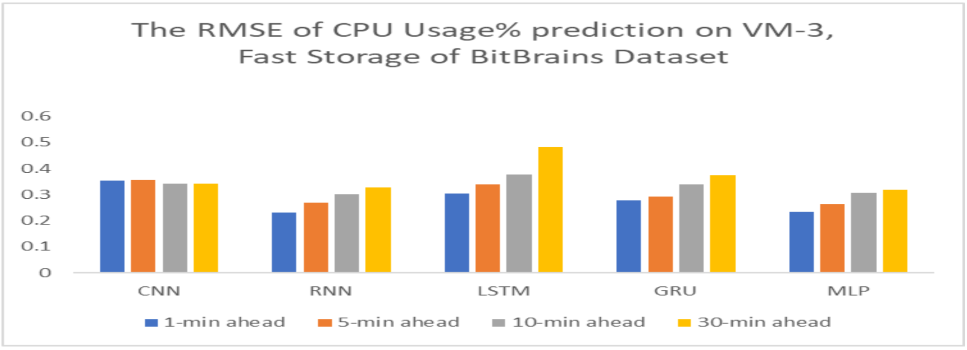

Figure 14: The Effect of RMSE for various deep learning methods of BB VM-3 dataset

From the data, we can conclude that MLP demonstrates the best overall performance in terms of root mean square error (RMSE) for CPU usage prediction across most time horizons, with values ranging from 0.233 to 0.319, indicating low prediction error. RNN follows closely, showing RMSE values between 0.232 and 0.327, reflecting strong reliability and consistent performance. GRU exhibits moderate performance, with RMSE ranging from 0.276 to 0.374, indicating slightly higher prediction error compared to RNN and MLP. CNN shows slightly higher RMSE values, varying from 0.341 to 0.356, suggesting moderate accuracy in predictions. LSTM has the highest RMSE among the methods, with values increasing from 0.305 to 0.482, indicating greater variability and less accuracy in CPU usage predictions as the time horizon increases. Overall, MLP and RNN are the most reliable methods, while LSTM exhibits the least accuracy in predictions due to its higher RMSE values.

Table 14: The MAPE of CPU Usage% prediction on VM-3, Fast Storage of BitBrains Dataset

Method	1-min ahead	5-min ahead	10-min ahead	30-min ahead
CNN	0.03	0.032	0.03	0.032
RNN	0.025	0.028	0.027	0.029
LSTM	0.051	0.052	0.053	0.059
GRU	0.028	0.028	0.031	0.032
MLP	0.028	0.029	0.029	0.03

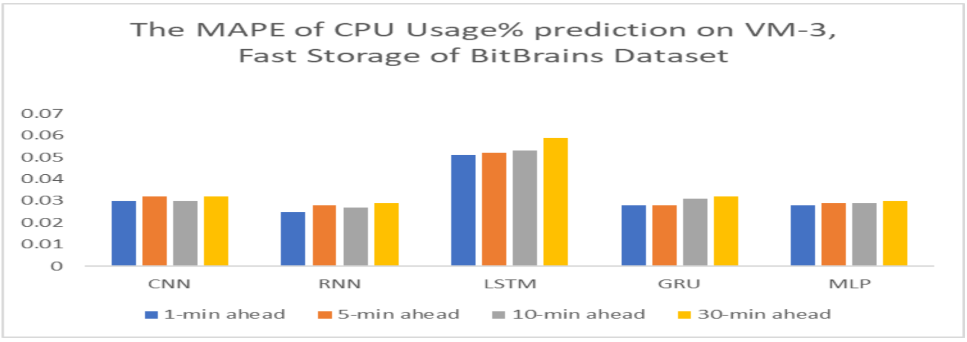

Figure 15: The Effect of MAPE for various deep learning methods of BB VM-3 dataset

From the data, we can conclude that RNN exhibits the best overall performance in terms of mean absolute percentage error (MAPE) for CPU usage prediction across all time horizons, with values ranging from 0.025 to 0.029, indicating a strong reliability in its predictions. CNN also performs relatively well, showing MAPE values between 0.03 and 0.032, which suggests stable accuracy but slightly higher error compared to RNN. GRU demonstrates comparable performance with MAPE values ranging from 0.028 to 0.032, reflecting moderate accuracy. MLP shows similar results, with MAPE between 0.028 and 0.030, indicating consistency in its predictions. In contrast, LSTM has the highest MAPE among the methods, with values ranging from 0.051 to 0.059, indicating greater variability and less accuracy in CPU usage predictions as the time horizon increases. Overall, RNN is the most reliable method for CPU usage prediction, while LSTM exhibits the least accuracy in predictions due to its higher MAPE values.

Industrial Engineering Journal

ISSN: 0970-2555

Volume : 53, Issue 11, No.4, November : 2024

Table 15: The MSLE of CPU Usage% prediction on VM-3, Fast Storage of BitBrains Dataset

Method	1-min ahead	5-min ahead	10-min ahead	30-min ahead
CNN	0.004	0.004	0.003	0.003
RNN	0.002	0.002	0.002	0.003
LSTM	0.003	0.005	0.005	0.008
GRU	0.002	0.003	0.003	0.004
MLP	0.002	0.002	0.003	0.003

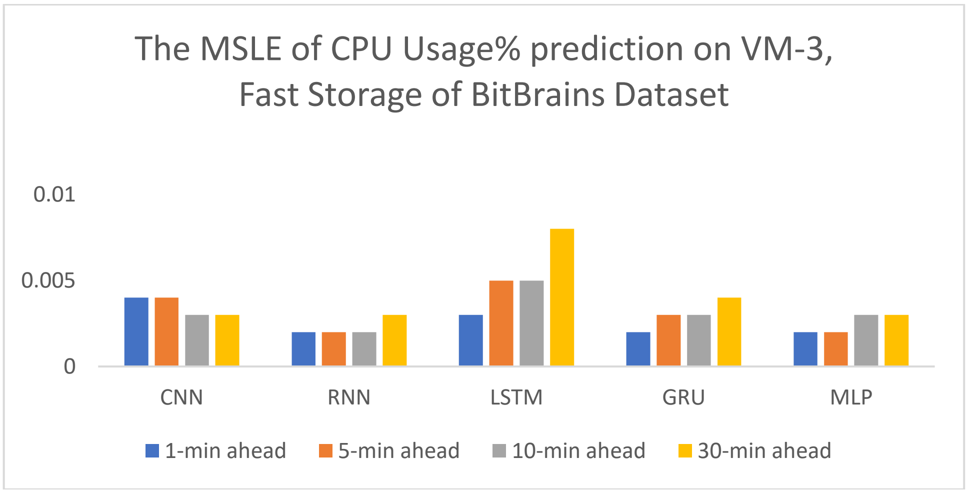

Figure 16: The Effect of MSLE for various deep learning methods of BB VM-3 dataset

From the data, we can conclude that RNN demonstrates the best performance in terms of mean squared logarithmic error (MSLE) for CPU usage prediction across various time horizons, with values consistently at 0.002 for the first three-time intervals and reaching 0.003 at the 30-minute mark. This indicates a strong reliability in its predictions with minimal error. GRU follows closely, showing MSLE values of 0.002 for the 1-minute interval, 0.003 for the 5 and 10-minute intervals, and 0.004 for the 30-minute interval, reflecting consistent accuracy. MLP matches GRU's performance with similar MSLE values across all time horizons. CNN also performs well, maintaining low MSLE values between 0.003 and 0.004, indicating stable predictions. LSTM exhibits slightly higher MSLE values, ranging from 0.003 to 0.008, suggesting a decrease in prediction accuracy as the time horizon extends. Overall, RNN is the most reliable method for CPU usage prediction based on MSLE, while LSTM shows the greatest variability in its predictive accuracy.

Computational costs:

Table 16: Effect of Computational Costs

Method	**Cloud Monitoring Data Set**	**BITBRAINS VM – 22**	**BITBRAINS VM - 3**
	Computational Time(sec)	Computational Time(sec)	Computational Time(sec)
CNN	14.44	65.69	64.57
RNN	18.45	370.14	146.35
LSTM	19.93	3153.385	3897.26
GRU	55.45	727.46	1154.34
MLP	13	15.58	14.67

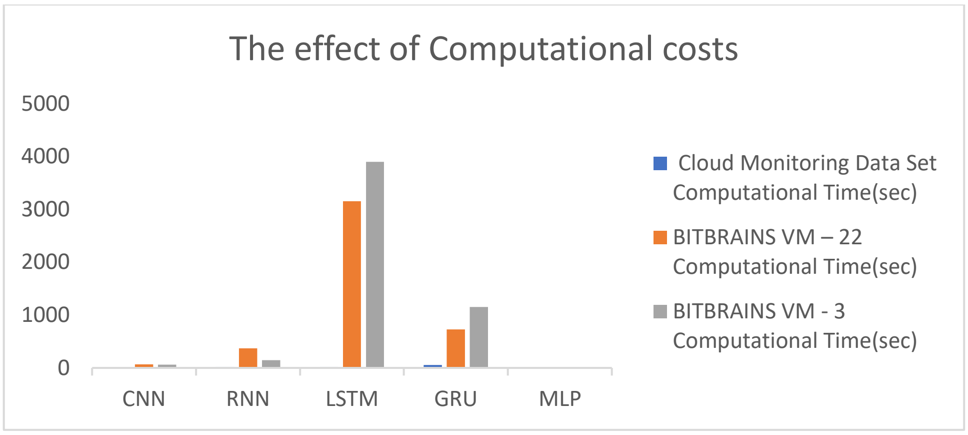

Fig: 17 The effect of Computational times of Various methods on two diff Datasets.

From the above table:16 highlights significant differences in computational time across methods and datasets. MLP consistently demonstrates lowest computational time, making it the fastest method when compared with all the other methods. Next to MLP, CNN also shows low computational time, maintaining competitive efficiency. RNN, while effective, requires significantly more time, particularly for certain datasets, indicating a higher computational cost. GRU offers moderate efficiency, with times that are higher than CNN and MLP but lower than RNN and LSTM. LSTM exhibits the highest computational time across all datasets, suggesting a considerable trade-off between its predictive performance and resource requirements. Overall, MLP stands out as the most computationally efficient method, while LSTM is the most resource-intensive.

Key Takeaways from Training and Prediction Results:

1. **CNN** models are efficient and provide fast predictions, making them suitable for real-time cloud applications. However, they tend to perform better for short-term predictions.
2. **RNN** models show balanced performance but require longer training and prediction times compared to CNN. They capture temporal dependencies better, making them a solid option for workload prediction.
3. **LSTM** networks are ideal for long-term forecasting, offering the highest accuracy at the expense of higher computation times. These models are suitable when prediction accuracy is prioritized over speed.
4. **GRU** models, while not as powerful as LSTM, achieve high accuracy with reduced training times, making them ideal for low-latency applications where both speed and accuracy are critical.
5. **MLP** models, though less effective for sequential data, perform surprisingly well in certain scenarios (e.g., Machine-22) and offer the fastest training times. They are useful for simpler prediction tasks.

7. CONCLUSION AND FUTURE SCOPES

This comparative study evaluates the effectiveness of RNN, CNN, LSTM, MLP, and GRU models for cloud workload prediction. The results show that each model has unique strengths and weaknesses, making them suitable for specific scenarios:

- **LSTM** consistently delivers the most accurate predictions, especially for longer time horizons. However, its high computational cost and training time may pose challenges for real-time applications.
- **MLP** and **CNN** provide competitive accuracy with faster training times, making them suitable for scenarios where rapid training and deployment are necessary.
- **GRU** offers a trade-off between performance and training efficiency, demonstrating effectiveness in shorter time horizons with moderate computational requirements.
- **RNN** models show balanced results but are outperformed by LSTM and GRU in terms of both accuracy and stability over time.

The study concludes that the choice of the model should depend on the use case, considering trade-offs between accuracy, training time, computational complexity, and prediction time. LSTM, although resource-intensive, remains the most accurate model for long-term predictions. MLP and CNN, on the other hand, are preferred for real-time, computationally light applications.

FUTURE SCOPES

This research opens several avenues for future exploration:

- Hybrid Models: Exploring hybrid models that combine the strengths of multiple architectures, such as CNN-LSTM or GRU-LSTM, could enhance prediction accuracy and efficiency.
- Transfer Learning: Investigate the use of transfer learning to adapt pre-trained models for workload forecasting in different cloud environments.
- Federated Learning Approaches: Integrate federated learning to enhance scalability and security by training models on distributed datasets without data sharing.
- Incorporation of External Factors: Extend the models to account for external factors like network traffic, seasonal variations, and economic events that impact workloads.
- Real-Time Adaptive Models: Develop adaptive models that can dynamically adjust to workload fluctuations and user demand in real-time, reducing prediction errors.
- Energy-Efficient Prediction Models: Focus on creating lightweight models that maintain high accuracy while minimizing energy consumption for green computing.
- Deployment in Edge and Serverless Computing: Explore the deployment of these models in serverless and edge computing environments, where resource constraints and latency requirements are crucial.

6. REFERENCES

[1].Qingchen Zhang, Laurence T. Yang, Zheng Yan, Zhikui Chen, and Peng Li "An Efficient Deep Learning Model to Predict Cloud Workload for Industry Informatics" DOI 10.1109/TII.2018.2808910, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS.

[2].Dattatray G. Takale, Aditya A. Wattamwar, Saksham S. Saipatwar, Harshwardhan V. Saindane, Tushar B. Patil "Comparative Analysis of LSTM, RNN, CNN and MLP Machine Learning Algorithms for Stock Value Prediction" JOURNAL OF Firewall Software and Networking (e-ISSN: 2584-1750) Double Blind Peer Reviewed Journal DOI: https://doi.org/10.48001/JoFSN.

[3]. Zaakki Ahamed, MaherKhemakhem and Kamal Jambi, Fathy Eassa, Fawaz Alsolami, Abdullah Basuhail "Deep Reinforcement Learning for Workload Prediction in Federated Cloud Environments" Sensors 2023, 23, 6911. https://doi.org/10.3390/ s23156911.

[4]. Zaakki Ahamed, Maher Khemakhem, Abdullah S, Al-Malaise Al-Ghamdi, Fathy Eassa, Fawaz Alsolami "Technical Study of Deep Learning in Cloud Computing for Accurate Workload Prediction" Electronics 2023, 12, 650. https://doi.org/10.3390/electronics12030650.

[5]. Dozdar Mahdi Ahmed, Masoud Muhammed Hassan, and Ramadhan J. Mstafa "A Review on Deep Sequential Models for Forecasting Time Series Data"Applied Computational Intelligence and Soft Computing Volume 2022, Article ID 6596397, 19 pages https://doi.org/10.1155/2022/6596397.

[6].Van-Thang Duong, Duc-Tuan-Anh Nguyen, Thi-Thu-Hang Pham, Van-Hau Nguyen and Van-Quoc Anh Le "Comparative Study of Deep Learning Models for Predicting Stock Prices" Proceedings of the Seventh International Conference on Research in Intelligent and Computing in Engineering pp 103-108. DOI: 10.15439/2022R02 ISSN 2300-5963 ACSIS, Vol. 33.

[7]. Mortezapour Shiri, Farhad & Perumal, Thinagaran& Mohamed, Raihani. (2023). A Comprehensive Overview and Comparative Analysis on Deep Learning Models: CNN, RNN, LSTM, GRU. 10.13140/RG.2.2.11938.81609.

[8]. Wang, Yupeng& Zhu, Shibing& Li, Changqing. (2019). Research on Multistep Time Series Prediction Based on LSTM. 1155-1159. 10.1109/EITCE47263.2019.9095044.

[9]. W. Li and K. L. E. Law, "Deep Learning Models for Time Series Forecasting: A Review," in IEEE Access, vol. 12, pp. 92306-92327, 2024, doi: 10.1109/ACCESS.2024.3422528. keywords: {Time series analysis;Forecasting;Predictivemodels;Autoregressiveprocesses;Electricity;Meteorology;Deeplearning;Performanceevaluation;Neuralnetworks;Transformers;Dataset;deeplearning;evaluationmetrics;neural network models;time series forecasting;Transformer models},

[10]. S. Bansal and M. Kumar, "Deep Learning-based Workload Prediction in Cloud Computing to Enhance the Performance," 2023 Third International Conference on Secure Cyber Computing and Communication (ICSCCC), Jalandhar, India, 2023, pp. 635-640, doi: 10.1109/ICSCCC58608.2023.10176790. keywords: {Cloud computing;Processorscheduling;Computationalmodeling;Neuralnetworks;Predictivemodels;Predictionalgorithms;Real-timesystems;CloudComputing;deeplearning;workloadprediction;neural networks},

[11].Mohammad Yekta and Hadi Shahriar Shahhoseini “A Review on Machine Learning Methods for Workload Prediction in Cloud Computing” 13th International Conference on Computer and Knowledge Engineering (ICCKE 2023), November 1-2, 2023, Ferdowsi University of Mashhad, Iran.

[12]. Masdari, M., Khoshnevis, A. “A survey and classification of the workload forecasting methods in cloud computing”. *ClusterComput* **23**, 2399–2424 (2020). https://doi.org/10.1007/s10586-019-03010-3.

[13]. Saxena, Deepika & Kumar, Jitendra & Singh, Ashutosh & Schmid, Stefan. (2023). Performance Analysis of Machine Learning Centered Workload Prediction Models for Cloud. IEEE Transactions on Parallel and Distributed Systems. PP. 1-18. 10.1109/TPDS.2023.3240567.

[14] https://github.com/microsoft/cloud-monitoring-dataset

[15] https://www.kaggle.com/datasets/gauravdhamane/gwa-bitbrains