Journal of Theoretical and Applied Information Technology
15th September 2024. Vol.102. No. 17

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

OPTIMIZING CLOUD RESOURCE UTILIZATION THROUGH MACHINE LEARNING FORECASTING

ANGELICA KAYLEE WIESI[1], JOSHUA SAMUAL[1], MOHAMMED AMIN ALMAIAH[2], AITIZAZ ALI[1], TAYSEER ALKHDOUR[3], ROMEL AL-ALI[4], THEYAZN H.H. ALDHYANI[5] AND RAMI SHEHAB[3]

[1]School of Technology Asia Pacific University of Technology & Innovation Kuala Lumpur, Malaysia
[2]King Abdullah the II IT School, University of Jordan, Amman 11942, Jordan.
[3]College of Computer Science and Information Technology, King Faisal University, Al-Ahsa 31982, Saudi Arabia
[4]Associate Professor, The National Research Center for Giftedness and Creativity, King Faisal University, Saudi Arabia
[5]Applied college in Abqaiq, King Faisal University, P.O. Box 400, Al-Ahsa 31982, Saudi Arabia.

tp064902@mail.apu.edu.my, talkhdour@kfu.edu.sa

ABSTRACT

This research introduces a resource utilization prediction tool tailored for dynamic and seasonal workloads in cloud environments. Traditional prediction methods often fall short in accuracy due to the constantly changing nature of cloud resources and workloads. To address this gap, the research proposes a machine learning-centric approach aimed at enhancing prediction accuracy, thereby promoting sustainability, energy savings, and improved user experience in line with SDG 7: Affordable and Clean Energy. The approach begins with data collection and preprocessing, employing techniques such as Fourier Series and Lag Features to capture temporal patterns. Three machine learning models—Gated Recurrent Unit (GRU), Long Short-Term Memory (LSTM), and Random Forest—are developed, trained, and evaluated using metrics like MAE, RMSE, and MAPE. Hyperparameter tuning is conducted to optimize model performance and minimize overfitting. The best-performing model, identified as the one-step GRU, is then deployed using Streamlit and AWS EC2, with User Acceptance Testing (UAT) ensuring it meets performance standards. This comprehensive approach demonstrates significant improvements in prediction accuracy and resource management, contributing to more efficient and sustainable cloud computing practices.

Keywords: *Resource Utilization; Lstm; Gru; Random Forest; Machine Learning; Sustainability*

1. INTRODUCTION

The rapid growth of the digital age has transformed how businesses and individuals interact with technology, with cloud computing emerging as a critical innovation. Cloud computing offers scalable, flexible, and cost-effective solutions, allowing access to services like software, platforms, and infrastructure on a pay-as-you-go basis [1]. This shift has revolutionized commercial strategies and fostered digital growth [2]. However, the dynamic nature of cloud environments presents unique challenges, particularly in resource management [3]. Unlike traditional on-premises systems with fixed resource allocation, cloud computing requires continuous adjustment due to unpredictable user demands [4]. This creates a risk of over-provisioning, leading to wasted resources and increased costs, or under-provisioning, which can result in poor service quality and unmet service level agreements (SLAs) [5]. Therefore, efficient and accurate resource utilization prediction is essential for optimizing cloud resource allocation, minimizing costs, and enhancing environmental sustainability.

To address these challenges, various resource utilization prediction techniques, including statistical methods, machine learning models, and deep learning architectures, have been developed. Traditional methods like ARIMA have limitations in handling the non-linear and dynamic nature of cloud workloads, necessitating more adaptive and accurate approaches [6]. Machine learning models, such as weighted quadratic random forests, genetic algorithms, and artificial neural networks (ANNs), have shown promise in improving prediction accuracy, but there is still room for enhancement [7]. The complexity of cloud resources, especially with the increasing influence of digitalization, demands a more dynamic method capable of adapting to real-

time demands and effectively handling seasonal and non-seasonal workload patterns [8]. The evolution of cloud computing requires prediction models that can accurately forecast resource utilization, particularly for CPU usage, to ensure cost-effectiveness, optimal service delivery, and user satisfaction.

The aim of this project is to develop a machine learning-based model to optimize cloud resource utilization prediction, focusing on accurately forecasting dynamic cloud workloads, including factors like seasonality and trends. By conducting a comparative analysis of various models, the project seeks to enhance the accuracy of resource utilization predictions, targeting specific performance metrics such as Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE). The anticipated benefits of this research include significant cost savings, improved operational efficiency, and enhanced customer satisfaction by ensuring SLAs are met and minimizing service disruptions. The insights gained from accurate predictions will enable businesses to make data-driven decisions, gain a competitive edge, and focus on innovation, ultimately contributing to sustainable and cost-effective cloud services.

2. OVERVIEW OF RESOURCE UTILIZATION IN CLOUD COMPUTING

This section provides a comprehensive overview of resource management and utilization in cloud computing. It explores the vital factors influencing resource consumption, and the challenges associated with effective allocation and utilization in cloud computing. Furthermore, it explores the application of machine learning techniques for accurate resource utilization prediction, a vital component of modern cloud management practices

A. Resource Management and Utilization in Cloud Computing

Cloud computing provides a scalable and flexible environment where users can access virtualized resources like memory, storage, network, and CPU on demand. This technology's inherent advantages, such as scalability, agility, and cost-effectiveness, make it a vital component of modern IT infrastructure. However, these benefits come with the challenge of managing these resources efficiently to satisfy both cloud providers and users. Resource management (RM) in cloud computing is crucial in this context, involving the acquisition, allocation, and monitoring of virtualized resources to ensure optimal performance and user satisfaction. Effective RM is key to balancing functionality, cost, and performance, ensuring the scalability, quality of service (QoS), and cost-effectiveness that cloud computing promises [9]. It focuses on capacity allocation, energy optimization, load balancing, and maintaining QoS to minimize downtime and ensure fast response times [10].

The dynamic nature of cloud computing presents unique challenges in RM, especially compared to traditional static data centers. Resource provisioning and allocation must be adaptable to fluctuating user demands to avoid violating Service Level Agreements (SLAs) and ensure customer satisfaction (Srivastava & Kumar, 2020). The unpredictability of cloud workloads, driven by varying application demands, often leads to an imbalance in resource utilization within data centers [11]. For instance, studies have shown that average CPU utilization in data centers is only about 17.76%, with memory utilization at 77.93%, indicating a significant underutilization of resources [12]. This imbalance not only results in inefficient resource usage but also leads to excessive energy consumption, contributing to a higher carbon footprint [13]. Data centers, as reported by the International Energy Agency (IEA) [14], were responsible for approximately 1% of global greenhouse gas emissions in 2020, underscoring the environmental impact of inefficient resource management.

To mitigate these challenges, resource management strategies in cloud computing need to evolve beyond traditional approaches. Conventional RM processes, which focus on allocating resources based on current demand, often lead to under-provisioning or over-provisioning, where tasks may be left incomplete due to a lack of resources, or resources may be wasted, incurring unnecessary costs [15]. A more effective approach involves predictive resource management, which uses historical data to anticipate future workloads and adjust resource allocation accordingly [16]. This predictive approach enhances resource efficiency, reduces latency, and ensures better alignment with actual resource needs, thereby preventing underutilization and overutilization [17]. By accurately forecasting future resource requirements, cloud providers can optimize resource usage, lower costs, and reduce the environmental impact, ultimately achieving a more sustainable and cost-effective cloud computing environment.

B. Resource Utilization Prediction Techniques

In cloud computing, resource utilization often faces imbalances that can lead to inefficiencies like over-provisioning or under-provisioning of resources. To address these challenges, cloud resource management increasingly relies on predictive techniques that leverage historical data to anticipate future demands. By accurately predicting future resource usage, cloud environments can

dynamically allocate resources, optimizing performance and cost efficiency.

Machine learning (ML), a subfield of artificial intelligence, plays a crucial role in these predictive approaches [18]. ML algorithms enable systems to learn from historical data and improve their predictions over time, much like how humans learn from experience [19]. The history of ML dates to the 1950s with Arthur Samuel's work on checker-playing programs, which marked the beginning of computers learning from data [20]. Over the decades, ML has expanded its influence across various industries, providing data-driven insights and enhancing decision-making processes in fields like healthcare, finance, and education.

ML can be broadly categorized into several types: supervised, unsupervised, reinforcement, and semi-supervised learning. Supervised learning is highly relevant for resource utilization prediction in cloud environments [21]. In supervised learning, algorithms are trained on labeled data—input data paired with the correct output—allowing the model to learn by comparison and correction [22]. As the model processes more data, it refines its predictions, making it increasingly accurate over time [22]. Supervised learning is further divided into classification, which deals with discrete labels, and regression, which handles continuous labels like numerical values [23]. For instance, regression is commonly used in stock price prediction, where the data points are continuous and ordered [22].

Unsupervised learning, another key ML approach, is used when the data lacks labels. Instead, the model identifies patterns and relationships within the raw data, making it particularly useful for clustering tasks where similar data points are grouped together [22]. While both supervised and unsupervised learning are powerful, traditional ML models often assume that data points are independent of one another, which can be a limitation in certain contexts [24].

This proceeds to the importance of time series analysis, particularly in cloud computing environments where resource usage data is inherently sequential. Unlike traditional ML models that treat data points as independent, time series analysis recognizes the temporal dependencies between data points [25]. In cloud environments, resource usage data such as CPU and memory consumption is collected over time, making time series analysis an essential tool for capturing patterns like trends, seasonality, and cyclical variations.

Time series data is unique in that each data point is influenced by its predecessors, allowing for the identification of patterns that span across different time frames [25]. These patterns can include long-term trends, such as the overall increase or decrease in resource usage, as well as short-term seasonal patterns, like the daily fluctuation in resource demand. Understanding these components is crucial for accurate prediction and resource management in cloud environments.

For instance, Recurrent Neural Networks (RNNs) are a type of ML model designed to handle sequential data by retaining information from previous inputs to inform current predictions [26]. This ability to look back at past data makes RNNs particularly effective for time series forecasting, where understanding the order of information is key [26].

In summary, the integration of machine learning, particularly supervised learning, with time series analysis offers a robust framework for predicting resource utilization in cloud environments. By leveraging the strengths of both approaches, cloud resource management can proactively anticipate and respond to fluctuating workloads, ensuring optimal resource allocation and minimizing inefficiencies. This combination of predictive techniques and time series analysis represents a significant advancement in the ability to manage cloud resources effectively, aligning with the dynamic and ever-evolving nature of cloud computing environments.

C. *Machine Learning Models*

Support Vector Machines (SVMs) are machine learning models that seek to find an optimal hyperplane to separate data into distinct categories with the widest margin [27]. This clear boundary allows for immediate classification without calculating probabilities. SVMs are advantageous for their ability to generalize and avoid overfitting, particularly when handling non-linear data using the kernel trick [27]. However, SVMs are computationally intensive, particularly with large datasets. In cloud environments, Support Vector Regression (SVR), a variant of SVM, has been used to optimize resource allocation [28]. Although SVR offers accurate predictions and handles both linear and non-linear data effectively, it also requires significant computational resources [28].

Random Forest is an ensemble learning model that combines multiple decision trees to improve prediction accuracy [29]. It is particularly effective when dealing with datasets with many features and fewer data points, offering built-in error estimates [29]. However, Random Forest struggles to capture linear relationships within the data [29]. In cloud computing, Random Forest has been applied to enhance resource management and has demonstrated high accuracy in CPU, memory, and disk usage predictions [29]. Despite its strengths, the integration of Random Forest into complex systems can be

challenging, and security concerns in resource allocation need to be addressed [29].

Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Network (RNN) designed to capture long-term dependencies in sequential data, such as time series [30]. LSTMs are well-suited for tasks requiring the retention of past information to recognize trends and patterns, but they are complex to implement and require substantial computational resources [30]. LSTM has been widely used in forecasting CPU utilization in cloud environments, showing superior accuracy compared to other models like ARIMA [28]. However, the complexity and resource demands of LSTM make it less interpretable and harder to deploy [28].

Gated Recurrent Unit (GRU) networks are a more streamlined version of RNNs, designed to address the vanishing and exploding gradient problems commonly found in standard RNNs [25]. GRUs use gating mechanisms to selectively retain or forget information, making them effective at capturing complex temporal patterns in sequential data [25]. GRUs have been successfully applied to cloud resource utilization predictions, demonstrating strong performance in workload prediction and energy consumption estimation [25]. Despite their effectiveness, GRUs, like LSTMs, require significant computational resources and are complex to implement [25].

Statistical methods, such as Autoregressive Integrated Moving Average (ARIMA) and its seasonal variant SARIMA, are traditional approaches to time series forecasting [24]. These methods focus on understanding the underlying mechanisms generating the data, making them highly interpretable and useful for capturing linear trends and seasonality [24]. However, statistical methods struggle with non-linear data and may provide inaccurate predictions in such cases [24]. While they are simpler and less prone to overfitting compared to machine learning models, the need to select appropriate models for different patterns can make them less practical in dynamic environments.

3. EXISTING SYSTEMS IN RESOURCE UTILIZATION IN CLOUD COMPUTING

Nawrocki et al. [26] proposed a dynamic, data-driven approach for cloud resource utilization prediction aimed at improving cost-effectiveness and environmental sustainability. This system adapts prediction algorithms to the nature of the data, overcoming the limitations of static methods that often result in under provisioning or overprovisioning of resources. By building a knowledge base from past data and selecting the best prediction model for each workload, the system ensures higher accuracy. However, the limited dataset scope and significant computational resources required for this approach are potential drawbacks.

Al-Asaly et al. [27] introduced a deep learning-based model using Diffusion Convolutional Recurrent Neural Networks (DCRNN) to address the challenge of fluctuating workloads in cloud environments. This model, tested on real-world CPU usage data from Planet Lab, demonstrated superior accuracy compared to conventional methods, as evidenced by lower MAPE and RMSE values. The DCRNN model effectively captures complex, non-linear workload patterns and adapts to changes in workload automatically. However, its dependency on extensive computational resources and the quality and quantity of the dataset are limitations.

Anupama et al. [28] proposed a hybrid approach combining SARIMA and LSTM models to tackle overprovisioning and under provisioning in cloud computing. SARIMA is adept at forecasting seasonal patterns, while LSTM handles non-linear and complex workloads. Tested on real-world data from Bitbrains cloud, this approach achieved lower MAE and MAPE, offering better accuracy and the ability to cover both seasonal and non-seasonal workloads. However, the complexity of combining two techniques and the focus on only CPU and memory usage are noted drawbacks.

Nashold & Krishnan [29] compared SARIMA and LSTM models for predicting CPU usage in cloud environments. Using data from Microsoft Azure, the study found that SARIMA performed better for long-term predictions, while LSTM excelled in short-term forecasts. The research highlighted that SARIMA struggles with dynamic data, whereas LSTM is limited in forming long-term dependencies. Despite some limitations in dataset scope and model implementation, the study provided valuable insights into the strengths of each model in cloud resource prediction.

Borkowski et al. [21] focused on enhancing cloud resource utilization by predicting resource needs with Artificial Neural Networks (ANN). Using data from over 3 million build process records, the ANN model was trained and validated, showing a significant reduction in prediction error. The study demonstrated the effectiveness of ANN in improving resource management but acknowledged the limitation of relying heavily on historical data, which might not always be available.

Valarmathi & Kanaga Suba Raja [22] combined Ensemble Random Forest (eRF) and LSTM models to improve CPU utilization prediction accuracy in cloud environments. Using a dataset from Alibaba, the study showed that the eRF-LSTM model

outperformed conventional methods, achieving enhanced accuracy and reduced training time. However, the scope was limited to CPU utilization, leaving out other resources like memory and storage that could also impact overall cloud resource management. This approach contributes to the ongoing research by providing an innovative model with improved prediction accuracy.

4. EXPERIMENTAL SETUP

This section will define the methodology, data gathering, and data collection process. A well-designed setup ensures the validity, reliability, and reproducibility of the study's findings. This section will outline the specific procedures and techniques employed in this research to collect and prepare the data for subsequent analysis and modeling.

A. Methodology

In developing a resource utilization prediction system, choosing the right methodology is crucial. The Waterfall methodology follows a linear, step-by-step process, moving through phases like requirement gathering, design, development, testing, and deployment in sequence [7]. This method is advantageous for projects with well-defined requirements, as it encourages clear planning [7]. However, its rigidity can be a drawback when changes or new insights arise, as revisiting previous phases is difficult [11].

On the other hand, the agile methodology is more flexible and iterative, breaking the project into smaller cycles that allow for continuous feedback and improvements [5]. This adaptability is particularly useful for projects that require ongoing stakeholder input, such as resource utilization prediction [5]. Agile's iterative process enables early prototyping and quick issue resolution, though it can also lead to extended timelines and higher costs due to the evolving project scope [5].

Given the need for continuous improvement and stakeholder collaboration in the resource utilization prediction system, the agile methodology is more suitable. The development process under Agile includes phases like requirements gathering, where stakeholder needs are identified; design, where appropriate prediction models are selected; and construction, where the system is developed and tested iteratively. Deployment and testing occur in each iteration, with user acceptance testing towards the end. Finally, feedback from stakeholders is gathered to refine the system in subsequent iterations, ensuring it meets user expectations and operational standards.

B. Data Gathering

The author employed interviews and observations to gather data and validate user requirements for a resource utilization prediction system in cloud environments. Interviews were conducted with two cloud industry professionals to gain insights into challenges, unpredictability of workloads, monitoring processes, and the impact of predictive tools on cloud resource management decisions. Key challenges identified include unpredictable workload spikes, difficulties in accurate cost and resource estimation during the design phase, and limitations of current tools in handling dynamic environments. Auto scaling was highlighted as a crucial feature for managing sudden traffic surges, though it primarily reacts rather than predicts. The professionals emphasized the need for ongoing monitoring and optimization to prevent over or under-provisioning.

Observations were used to analyze the training process of the prediction models. By plotting data and examining metrics like MAE and MAPE, the author assessed the impact of different time intervals on model accuracy. The observations helped identify patterns such as seasonality, which are crucial for accurate prediction.

From the analysis, user requirements were inferred: tools that can accurately predict resource utilization, particularly in dynamic environments, and the need for continuous optimization to balance short-term and long-term resource planning. The insights gathered underscore the importance of advanced predictive tools that can differentiate between regular patterns and fluctuating demands.

C. Data Collection

This research utilizes the dataset titled "AzureReadings_at_a_timestamp.csv," publicly available on LeadingIndiaAI's GitHub repository [10]. Stored in a comma-separated values (CSV) format, this dataset allows for convenient use by various software programs.

The data specifically focuses on CPU utilization metrics gathered from Microsoft Azure, a prominent cloud provider [10]. These metrics offer a comprehensive perspective on resource consumption patterns within virtual machines (VMs) [10]. Designed to capture the dynamic nature of cloud resources, the dataset is meticulously sampled at a high frequency of 5 minutes [10]. This granular sampling interval provides a detailed picture of the data, enabling researchers to observe even short-term fluctuations in resource demands [10].

The values within the dataset are not explicitly labeled as Hertz (Hz) or percentage. However, considering the presence of values reaching a

million, it's highly improbable for them to represent either unit. The most likely scenario is that the values are scaled. This scaling serves a dual purpose: protecting sensitive information and ensuring data usability. Raw CPU utilization data might pose privacy concerns; therefore, scaling the data achieves a balance between security and functionality, as the scaled values still accurately reflect the underlying trends in the raw data.

5. DESIGN AND IMPLEMENTATION

Effective model development starts on a solid foundation built through thorough data understanding and preparation. This section will focus on preparing the data into a suitable format for model training and evaluation. An exploratory data analysis will initiate the process to gain insights into data characteristics, identify potential issues, and extract vital features. Furthermore, the data undergoes preprocessing to handle missing values, outliers, and inconsistencies, ensuring data quality and reliability. The reprocessing also applies the features found throughout EDA. The clean and prepared dataset proceed to go through model building that will explore various algorithms and techniques to develop a robust predictive model

A. Data Understanding

TABLE I. *Variables Table*

Variables	Description
timestamp	Timestamp with an interval of 5 minutes
min cpu	This metric indicates the lowest CPU usage recorded during the five-minute interval, providing information about potential idle periods or low-demand phases.
max cpu	This metric represents the peak CPU usage observed within the five-minute interval, providing insights into the upper bound of resource demand during that period.
avg cpu	This metric captures the mean CPU utilization across the entire five-minute interval, offering a representative value of resource consumption.

This section describes the variables included in the CSV files and their corresponding descriptions. The timestamp column will serve as the index for the dataset. The three main columns will be analyzed through exploratory data analysis (EDA) to identify patterns and create new features while removing unnecessary information.

Before proceeding to model development, a comprehensive understanding of the data is important. An exploratory data analysis (EDA) is critical in uncovering hidden patterns, trends, and anomalies within the dataset. This examination aims to extract valuable insights that will inform subsequent modeling efforts. These are the key steps that are involved in the EDA:

1. **Import Libraries**: Essential Python libraries for data processing, such as Pandas and Matplotlib, were imported to facilitate the analysis.
2. **Data Loading**: The dataset was loaded using the Pandas library. The 'timestamp' column was converted to a datetime format and set as the index to enable time-based operations. The initial inspection of the dataset confirmed successful loading and revealed that the dataset had 8640 rows and memory usage of approximately 270 KB.
3. **Time Series Visualization**: A line graph was plotted to visualize the CPU utilization metrics ('min_cpu', 'max_cpu', and 'avg_cpu'). The visualization revealed periodic patterns with daily and weekly cycles, along with an upward trend in 'avg_cpu' and 'max_cpu'. Occasional extreme spikes suggested the presence of anomalies.
4. **Statistical Analysis**: Descriptive statistics were generated, showing that the dataset had high mean values and large standard deviations, indicating significant variation in CPU usage. This variability suggested the presence of dynamic workloads and potential outliers.
5. **Distribution Plots**: The distribution of CPU usage for each column was visualized using histograms and KDE curves. The 'min_cpu' showed a normal distribution, while 'max_cpu' and 'avg_cpu' were slightly right skewed. The 'avg_cpu' was deemed the most stable and representative metric for modelling.
6. **Correlation Matrix**: A correlation matrix and heatmap were generated, revealing strong positive correlations between 'min_cpu', 'max_cpu', and 'avg_cpu'. The high correlation supported the use of 'avg_cpu' for prediction as it effectively represents overall CPU utilization.
7. **Feature Filtering**: Based on the analysis, only the 'avg_cpu' column was selected for further modeling. The dataset was reloaded, and unnecessary columns were filtered out.
8. **Null Value Check**: The dataset was checked for null values in the 'avg_cpu' column, and none were found.

9. **Data Shape Verification**: The shape of the dataset was confirmed, ensuring the correct number of rows and columns were present after filtering.
10. **Anomaly Detection**: Anomalies were detected using the Interquartile Range (IQR) method. Points above 1.5 million in 'avg_cpu' were identified as outliers and marked as anomalies.
11. **Anomaly Interpolation**: The anomalies were handled by interpolating the missing values, filling the gaps with estimated values based on surrounding data points. The impact of this interpolation was visualized to ensure the time series remained consistent.
12. **Time Series Decomposition**: The time series was decomposed into trend, seasonality, and residual components using multiplicative decomposition. The analysis revealed a slight upward trend and strong daily seasonality, which could be useful for feature engineering.
13. **Moving Average Calculation**: A moving average was calculated over a 24-hour window to smooth out fluctuations. The trend analysis indicated a non-linear, polynomial trend, which could be important for the model to learn.
14. **Seasonal Subseries Plot**: A seasonal subseries plot was created to visualize hourly patterns across different days. The plot highlighted differences in CPU usage between weekdays and weekends, suggesting the influence of business day effects.
15. **Month Season Plot**: A monthly seasonal plot was generated to observe hourly patterns across the month. This plot reinforced the presence of distinct hourly patterns, indicating that incorporating time-based features could improve model performance.
16. **Period gram Time Series**: A period gram was used to analyse the frequency content of the data. The dominant frequency was identified as daily, supporting the earlier findings of strong daily seasonality.
17. **Lag Plot, ACF, and PACF**: Lag plots, along with Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) analyses, were conducted to explore the relationship between current and past values. The analysis determined that a lag of 36 (equivalent to 3 hours) was optimal for retaining relevant data while minimizing decay.

Overall, the EDA provided a comprehensive understanding of the CPU utilization dataset, revealing key patterns, correlations, and potential features for modeling. The insights gained from this analysis will guide the next steps in model development and optimization.

B. Data Preprocessing

The preprocessing of the CPU utilization dataset involved several carefully considered steps to ensure the data was ready for modelling. The process began with importing the necessary libraries for data manipulation, visualization, and model building, setting the foundation for the subsequent tasks. The dataset, containing 'timestamp' and 'avg_cpu' columns, was then loaded. These specific columns were chosen because 'avg_cpu' had a strong correlation with both 'min_cpu' and 'max_cpu,' making it a suitable representative for CPU utilization. The timestamp data was parsed into date time format and set as the index, and the dataset was resampled to maintain consistent 5-minute intervals, ensuring uniformity in the time series data.

Next, the dataset was checked for missing values, revealing none, which indicated a clean data set. However, the dataset required further scrutiny for anomalies. Using the Interquartile Range (IQR) method, anomalies were identified and flagged in a new column. These anomalies, likely representing significant deviations from the norm, were then addressed by replacing the corresponding 'avg_cpu' values with NaNs. These missing values were subsequently interpolated, a method chosen to smooth out the data while maintaining the integrity of the time series.

To capture the underlying trend in the data, polynomial regression was employed. A new feature, 'time_numeric,' was created to map each data point in the time series to a numerical value, facilitating trend analysis. The resulting trend was then added to the dataset as a new column. Reproducibility was ensured by setting a fixed seed for random operations, a critical step for maintaining consistency in the results across multiple iterations, particularly important in machine learning experimentation.

Recognizing the presence of periodic patterns in the data, Fourier series were generated to capture the seasonality. This step involved creating sine and cosine components to reflect daily and weekly cycles, providing the model with additional context about recurring patterns in the data. Additionally, lag features were created for the 'avg_cpu' column to

capture the influence of past values on current observations. This was informed by prior analysis, which indicated that lags up to 36 periods could be relevant. Rows with missing values, resulting from this lagging process, were removed to maintain data reliability.

The data was then normalized using MinMaxScaler, a crucial step for models like GRU and LSTM, which are sensitive to the scale of input features. The normalization process scaled the features to a [0, 1] range, ensuring that all features contributed equally during the modelling process. To prepare the data for sequential tasks, a sliding window of size 50 was created. This allowed the model to learn patterns and dependencies within the data over time, by structuring the data into sequences.

Finally, the dataset was split into training, validation, and testing sets, with a 70%, 20%, and 10% distribution, respectively. This split was done for both features and target variables, with 'avg_cpu' as the target and all other features contributing to the model's predictions. These pre-processing steps collectively ensured that the dataset was well-prepared for effective modelling, with a focus on capturing trends, seasonality, and temporal dependencies, while maintaining data integrity and consistency.

C. *Model Building*

The model building process explored three models: Gated Recurrent Unit (GRU), Long Short-Term Memory (LSTM), and Random Forest, each with specific configurations and approaches to prediction and evaluation.

The GRU model was first implemented as a baseline, with no hyper parameter tuning. It consisted of two GRU layers with 64 and 32 hidden units, respectively, and a dropout rate of 20% to prevent overfitting. The final layer was a Dense layer with one neuron, intended for one-step-ahead predictions. The model was trained and evaluated using Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and accuracy metrics. The baseline performance was strong, prompting further exploration into multi-step predictions. The GRU model was then extended to predict 12-time steps ahead by adjusting the Dense layer to output 12 values. Finally, hyper parameter tuning using Random Search optimized the GRU model's performance by adjusting hidden units, dropout rates, and learning rate, with early stopping implemented to prevent overfitting.

The LSTM model followed a similar structure to the GRU model, with two LSTM layers of 64 and 32 hidden units and a 20% dropout rate. The model was compiled using the Adam optimizer and a loss function of Mean Squared Error (MSE). After training and validation, the LSTM model's performance was evaluated using the same metrics as the GRU model. Hyper parameter tuning was also conducted using Random Search, optimizing the number of hidden units, dropout rates, and the optimizer used. The best parameters were determined to improve the LSTM model's performance.

The Random Forest model was defined and trained for regression with an initial configuration of 100 decision trees (n_estimators). The model was trained, and predictions were made on the training, validation, and testing sets. Manual hyperparameter tuning was performed to optimize the model further, adjusting parameters such as the number of trees, maximum depth, minimum samples required to split a node, and minimum samples per leaf. The best-performing configuration was identified based on evaluation metrics.

Overall, the model building process included a thorough exploration of GRU, LSTM, and Random Forest models. Each model was initially implemented as a baseline and then optimized through hyperparameter tuning. The GRU model demonstrated strong baseline performance, which was enhanced further for multi-step predictions. The LSTM and Random Forest models were also fine-tuned to achieve optimal results, with their performance evaluated and compared across various metrics.

6. MODELS RESULT AND DISCUSSION

The discussion section provides a comprehensive analysis of the various models used in the research, focusing on their performance in predicting CPU utilization. Among the models examined, the Gated Recurrent Unit (GRU) models, both one-steps ahead and 12-step ahead, Long Short-Term Memory (LSTM) models, and Random Forest models were evaluated based on their Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and accuracy metrics.

Journal of Theoretical and Applied Information Technology
15th September 2024. Vol.102. No. 17

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A. Models Evaluation and Comparison

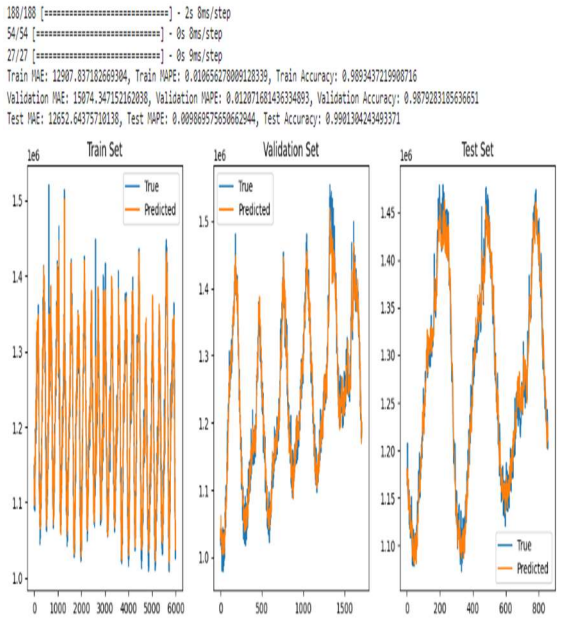

Fig. 1. GRU output line plot.

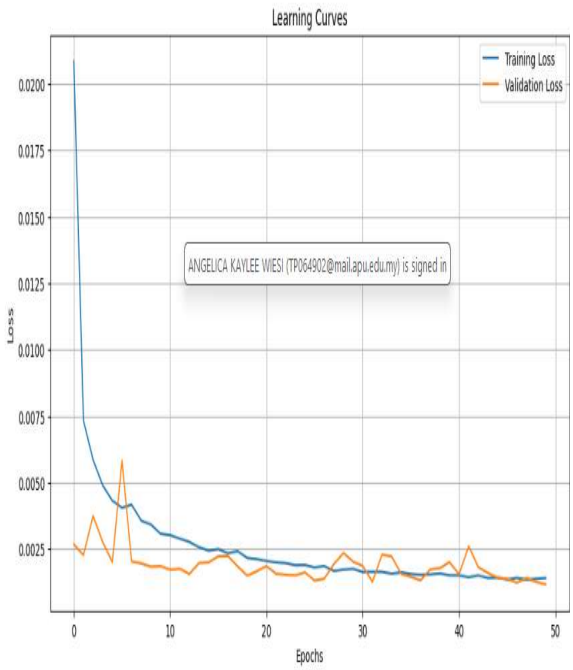

Fig. 2. GRU learning curves.

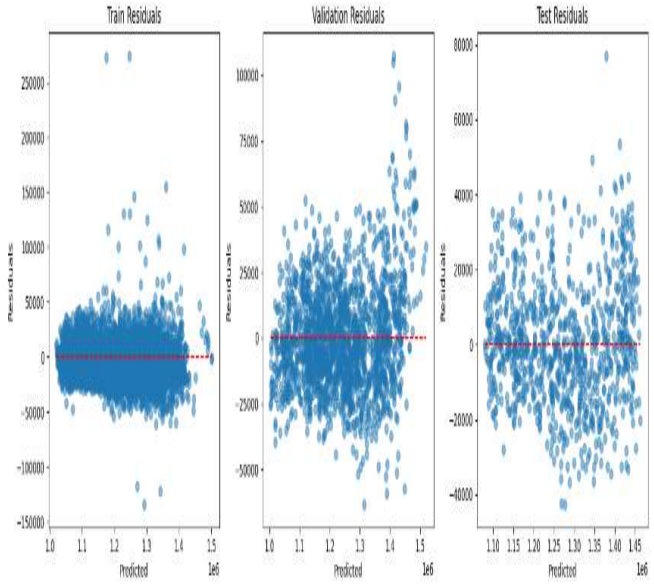

Fig. 3. GRU residual analysis.

		MAE	MAPE	Accuracy
Final One-Step GRU Model	Train	0.011	0.01	98.90%
	Validation	0.012	0.012	98.70%
	Test	0.009	0.009	99%

Fig. 4. GRU metrics outputs.

The final one-step ahead GRU model exhibited the strongest performance across all evaluation metrics. This model demonstrated a high degree of accuracy, especially on the test set, where it achieved a remarkable 99% accuracy. The analysis suggests that the model generalizes well to unseen data, though there is a slight indication of overfitting, as seen from the comparison between the training and test set performance. The learning curves and residual analysis supported this observation, showing that while the model performed well, there is still minimal overfitting that might benefit from further refinement.

Journal of Theoretical and Applied Information Technology
15th September 2024. Vol.102. No. 17

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

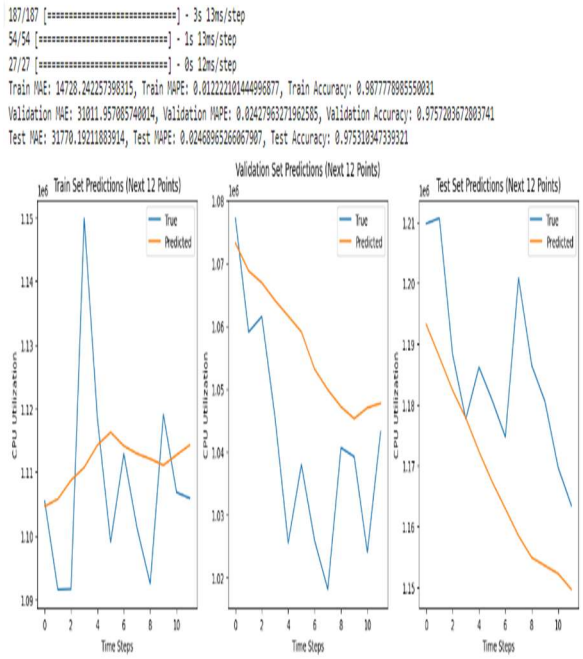

Fig. 5. GRU base 12-steps ahead line plot.

Base 12-Step GRU Model	Train	0.012	0.012	98.70%
	Validation	0.025	0.024	97.50%
	Test	0.024	0.025	97.50%

Fig. 6. GRU base 12-steps ahead metrics outputs.

On the other hand, the base 12-step ahead GRU model did not perform as well as the one-step model. It displayed significantly higher error rates and lower accuracy. The discussion points out that this could be attributed to the longer sequence length, which makes capturing long-term patterns more challenging. Moreover, the preprocessing steps for the 12-step model were the same as those used for the one-step model, including window size and lag features. However, since these models are designed to capture different sequential lengths—short-term versus long-term—the preprocessing steps might need to be reassessed to better accommodate the needs of a 12-step ahead prediction model.

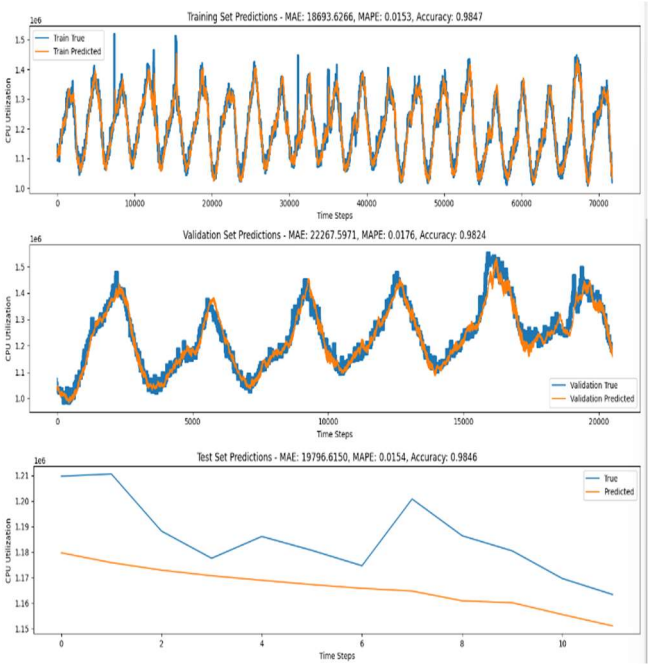

Fig. 8. GRU final 12-steps ahead line plots.

Final 12-Step GRU Model	Train	0.016	0.015	98.40%
	Validation	0.018	0.018	98.20%
	Test	0.016	0.015	98.40%

Fig. 9. GRU final 12-steps ahead metrics outputs.

After tuning, the 12-step GRU model showed some improvement, particularly in reducing overfitting. Despite these enhancements, it still struggled with handling longer sequences effectively and continued to exhibit higher error rates compared to the one-step GRU model. The discussion suggests that further experimentation with preprocessing and feature engineering could help improve the performance of the 12-step model.

Journal of Theoretical and Applied Information Technology
15th September 2024. Vol.102. No. 17

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

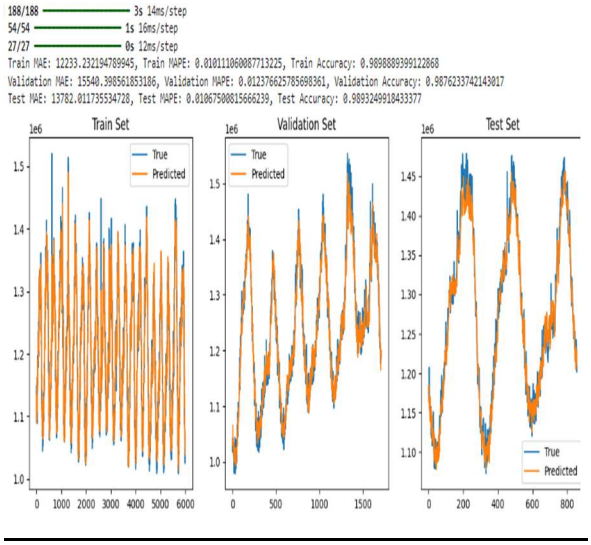

Fig. 10. LSTM base line plots.

		MAE	MAPE	Accuracy
Base LSTM Model	Train	0.01	0.01	98.90%
	Validation	0.012	0.012	98.70%
	Test	0.011	0.011	98.9%

Fig. 11. LSTM base metrics outputs.

The base LSTM model performed well on both the training and test sets, with a slight decrease in performance on the validation set. Although the model demonstrated good overall performance, it exhibited a minor degree of overfitting, as indicated by the difference in MAE and MAPE between the training and testing sets. This slight gap suggested the need for further tuning to enhance the model's performance.

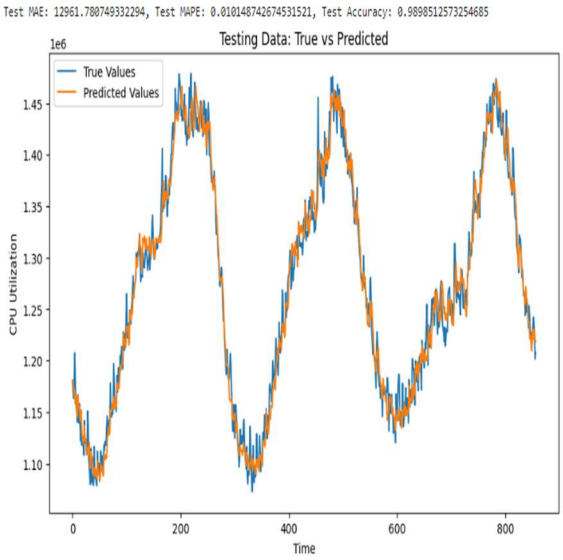

Fig. 12. LSTM final line plot.

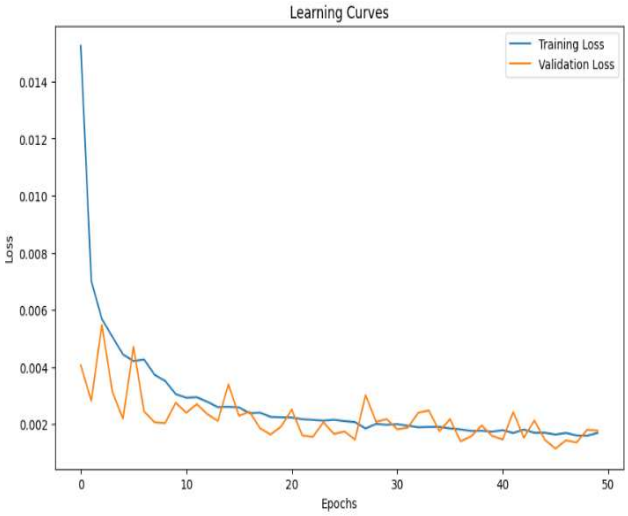

Fig. 13. LSTM final learning curve.

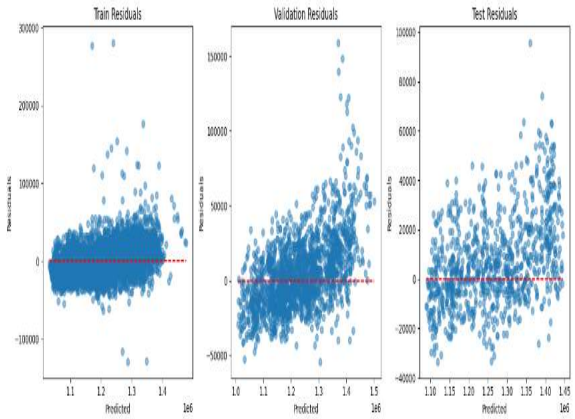

Fig. 14. LSTM final residual analysis.

Final LSTM Model	Train	0.011	0.011	98.90%
	Validation	0.012	0.012	98.70%
	Test	0.01	0.01	98.90%

Fig. 15. LSTM final metrics outputs.

The final tuned LSTM model showed a very slight improvement over the base model. Although there were some changes in the metrics, these changes were minimal after rounding, and overall, the model continued to exhibit strong training performance with reasonable generalization to unseen data. The learning curve analysis revealed that while the model learned the data effectively, there were some fluctuations in the validation loss, indicating room for improvement, possibly through the use of regularization techniques. The residual analysis also highlighted some patterns and clusters in the validation and test sets, suggesting that further tuning or feature engineering could enhance the model's ability to capture these patterns more effectively.

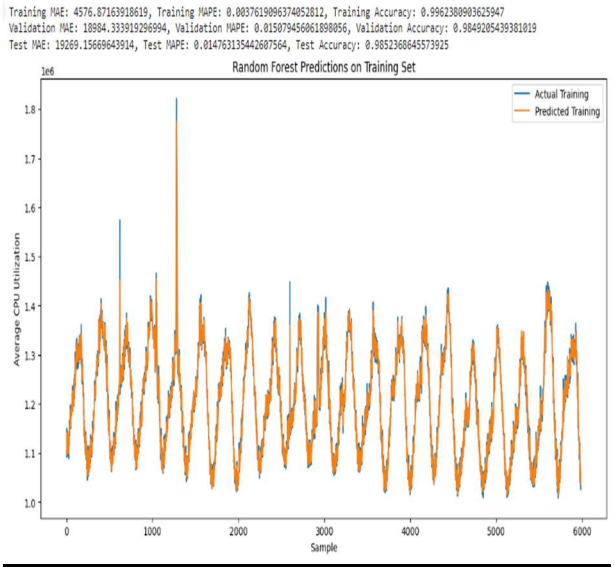

Fig. 16. Random Forest base Training Set line plot.

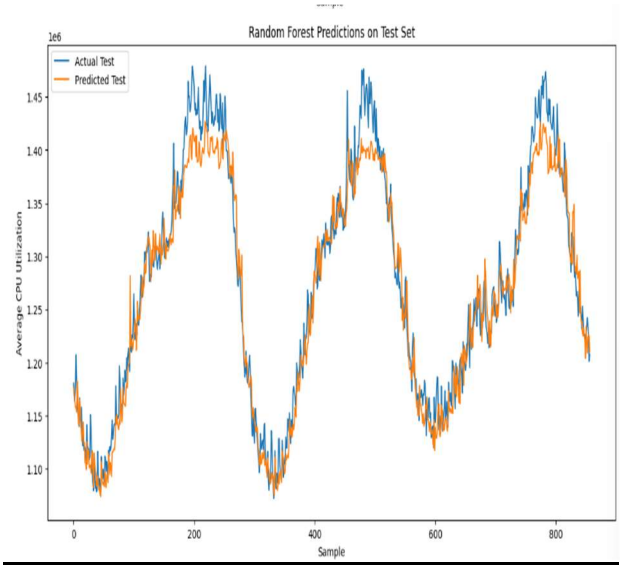

Fig. 17. Random Forest base Testing Set line plot.

		MAE	MAPE	Accuracy
Base Random Forest Model	Train	0.004	0.003	99.60%
	Validation	0.015	0.015	98.40%
	Test	0.015	0.015	98.50%

Fig. 18. Random Forest base metrics outputs.

The Random Forest model initially showed strong performance on the training set with very low MAE and MAPE values. However, there was a significant gap between the training and

validation/testing metrics, indicating a severe overfitting issue. The discussion highlights that this overfitting could be mitigated through hyperparameter tuning.

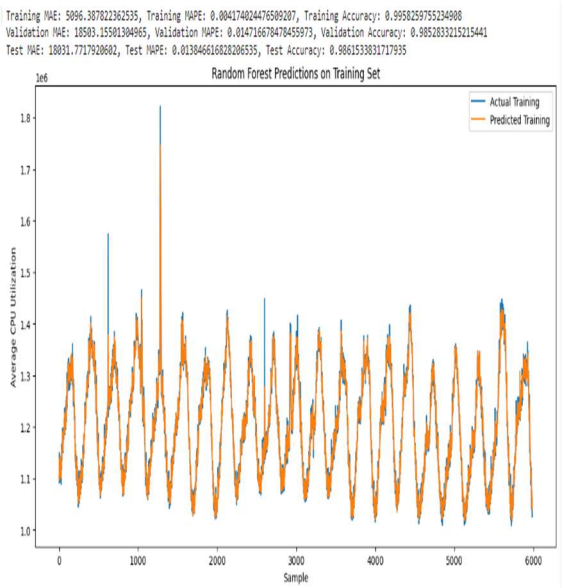

Fig. 19. Random Forest final Training Set line plot.

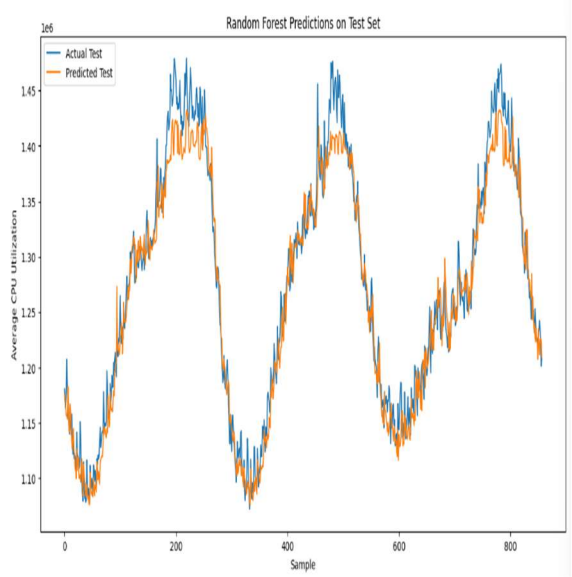

Fig. 20. Random Forest final Testing Set line plot.

Final Random Forest	Train	0.004	0.004	99.50%
	Validation	0.015	0.015	98.50%
	Test	0.014	0.014	98.60%

Fig. 21. Random Forest final metrics outputs.

After tuning, the Random Forest model showed an improved performance with reduced overfitting, as evidenced by the decreased MAE on the validation and test sets. Despite this improvement, the Random Forest model continued to struggle with handling time series data, showing higher error rates compared to the GRU and LSTM models. The discussion suggests that the Random Forest model may require additional features or more complex preprocessing steps to better capture the patterns in the time series data. Additionally, the model's architecture, which is less suited for sequential data compared to GRU and LSTM models, might be inherently limited in its ability to effectively predict dynamic time series data.

		MAE	MAPE	Accuracy
Final One-Step GRU Model	Train	0.011	0.01	98.90%
	Validation	0.012	0.012	98.70%
	Test	0.009	0.009	99%
Final 12-Step GRU Model	Train	0.016	0.015	98.40%
	Validation	0.018	0.018	98.20%
	Test	0.016	0.015	98.40%
Final LSTM Model	Train	0.011	0.011	98.90%
	Validation	0.012	0.012	98.70%
	Test	0.01	0.01	98.90%
Final Random Forest	Train	0.004	0.004	99.50%
	Validation	0.015	0.015	98.50%
	Test	0.014	0.014	98.60%

Fig. 22. Tuned model outputs.

The final comparison across all models clearly identifies the final one-step GRU model as the superior performer, with the best generalization to unseen data and minimal overfitting. The final 12-step GRU model, while improved from its baseline version, still fell short in comparison to the one-step GRU model, particularly in handling long-term sequences. The final LSTM model came close to the performance of the one-step GRU model, showing strong generalization and minimal overfitting, but it still lagged slightly behind in overall accuracy and error rates. The Random Forest model, even after tuning, remained the weakest performer in this study, likely due to its limitations in handling time series data as effectively as the GRU and LSTM models.

The final section also compares these models to those used in other research studies. The models in this research, particularly the GRU and LSTM models, demonstrated superior performance, with lower MAE and MAPE values and higher accuracy rates compared to models from other studies. This strong performance is attributed to the specific dataset, preprocessing, and tuning strategies

employed in this research, with the final one-step GRU model emerging as the most effective model for predicting CPU utilization in this context.

B. Model Deployment

The model deployment section outlines the steps taken to deploy the trained machine learning model, starting with saving the model and scalers, followed by the deployment process utilizing GitHub, Streamlit, and AWS EC2. The process begins with saving the trained TensorFlow model and the associated scalers. The pickle library is used to save the model as an HDF5 file. The scalers, which were used to preprocess the data during training, are also saved to ensure that any new data in the deployment phase is processed consistently, thereby maintaining the reliability of the predictions.

The next phase of deployment involves the actual implementation of the model using Streamlit, a tool that enables the creation of a user-friendly web application. The deployment code is designed to load the saved model and scalers, process the input data, and make predictions. Users can upload a CSV file containing historical CPU usage data, which the application will preprocess, detect anomalies, and perform feature engineering on before making predictions using the pre-trained model. The application then displays the first five rows of the input data, the predicted values, a plot of the prediction, and provides advice to the administrator based on the predicted CPU usage.

The final step connects the application to the AWS cloud by creating an EC2 instance. The EC2 instance is configured with Ubuntu, and a security group is set up to allow SSH, HTTP, and HTTPS access. Additionally, port 8501, which is used by the Streamlit application, is opened. The deployment process continues with connecting to the instance via Putty, where several commands are executed to update and upgrade system packages, install necessary tools, clone the GitHub repository, and install Python along with the required dependencies. The application is then launched using Streamlit by running the application script.

Once the application is running, it can be accessed through a web browser using the public IP address of the EC2 instance with port 8501. The user interface allows administrators to upload CSV files, process the data, and receive predictions along with visualizations and actionable advice, ensuring effective monitoring and management of CPU utilization.

7. CONCLUSION

The present research successfully achieved its primary objectives of comparing machine learning models for CPU utilization prediction, surpassing the accuracy of existing studies, and incorporating multiple time series components to enhance predictive performance. The comparative analysis of GRU, LSTM, and Random Forest models yielded valuable insights into model selection and optimization.

The developed system, capable of predicting CPU utilization based on historical data, offers practical benefits for cloud administrators. By anticipating resource demands, organizations can optimize resource allocation, minimize over-provisioning and under-provisioning, and enhance overall operational efficiency. Accurate prediction contributes to cost savings, improved service level agreements, and reduced environmental impact.

While the research yielded promising results, several limitations must be acknowledged. The sole focus on CPU utilization limits the model's generalizability to complex cloud environments where multiple resources interact. Incorporating additional resource metrics, such as memory and disk usage, is necessary to capture the intricate relationships between different resource types.

Furthermore, the study primarily concentrated on short-term prediction, leaving room for improvement in long-term forecasting. The development of models capable of accurately predicting resource utilization over extended periods requires further investigation into complex temporal patterns and their impact on prediction accuracy.

The availability of a limited dataset spanning only one month posed challenges in capturing long-term trends and seasonal variations. A more extensive dataset encompassing multiple years would provide a richer foundation for model development and evaluation.

To address the limitations and advance the field of cloud resource utilization prediction, several recommendations are proposed. Future research should prioritize the development of multivariate models that consider the interdependencies between various cloud resources. This would enhance the model's ability to capture the holistic behaviour of cloud systems.

Additionally, exploring advanced time series modeling techniques and feature engineering approaches is crucial for improving long-term prediction accuracy. Investigating the impact of different time scales, such as hourly, daily, and weekly patterns, on model performance can provide valuable insights. Expanding the dataset to cover a longer duration is essential for capturing the full spectrum of temporal variations and trends in cloud resource utilization. This would enable the development of more robust and reliable prediction

models. By addressing these recommendations, future research can contribute to the development of sophisticated and accurate cloud resource utilization prediction systems, enabling organizations to optimize their cloud operations and achieve greater efficiency and sustainability.

ACKNOWLEDGMENT

This work was supported by the Deanship of Scientific Research, Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, Saudi Arabia (Grant No. KFU241776)

REFERENCES

[1]. Anupama, K. C., Shivakumar, B. R., & Nagaraja, R. (2021). Resource Utilization Prediction in Cloud Computing using Hybrid Model. International Journal of Advanced Computer Science and Applications, 12(4), 373–381. https://doi.org/10.14569/IJACSA.2021.0120447

[2]. Anushuya, G., Gopikaa, K., Gokul, S., & Keerthika, P. (2018). Resource Management in Cloud Computing using SVM with GA and PSO. www.ijert.org

[3]. Ashraf Zargar, S. (2021). Introduction to Sequence Learning Models: RNN, LSTM, GRU. https://doi.org/10.13140/RG.2.2.36370.99522

[4]. Awad, M., & Khanna, R. (2015a). Efficient learning machines: Theories, concepts, and applications for engineers and system designers. In Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers. https://doi.org/10.1007/978-1-4302-5990-9

[5]. Awad, M., & Khanna, R. (2015b). Support Vector Machines for Classification. In Efficient Learning Machines. https://doi.org/10.1007/978-1-4302-5990-9_3

[6]. Barker, J. (2020). Machine learning in M4: What makes a good unstructured model? In *International Journal of Forecasting* (Vol. 36, Issue 1). https://doi.org/10.1016/j.ijforecast.2019.06.001

[7]. Barroso, L. A., & Hölzle, U. (2009). The datacenter as a computer: An introduction to the design of warehouse-scale machines. Synthesis Lectures on Computer Architecture, 6. https://doi.org/10.2200/S00193ED1V01Y200905CAC006

[8]. Borkowski, M., Schulte, S., & Hochreiner, C. (2016). Predicting cloud resource utilization. Proceedings - 9th IEEE/ACM International Conference on Utility and Cloud Computing, UCC 2016. https://doi.org/10.1145/2996890.2996907

[9]. Chandy, A. (2019). SMART RESOURCE USAGE PREDICTION USING CLOUD COMPUTING FOR MASSIVE DATA PROCESSING SYSTEMS. Journal of Information Technology and Digital World, 01(02), 108–118. https://doi.org/10.36548/jitdw.2019.2.006

[10]. Chen, J., & Wang, Y. (2018). A resource demand prediction method based on EEMD in cloud computing. Procedia Computer Science, 131. https://doi.org/10.1016/j.procs.2018.04.193

[11]. Cutler, A., Cutler, D. R., & Stevens, J. R. (2012). Random forests. In Ensemble Machine Learning: Methods and Applications. https://doi.org/10.1007/9781441993267_5

[12]. Emmert-Streib, F., & Dehmer, M. (2022). Taxonomy of machine learning paradigms: A data-centric perspective. In Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery (Vol. 12, Issue 5). https://doi.org/10.1002/widm.1470

[13]. Javatpoint. (2024). Agile Software Development Life Cycle (SDLC). https://www.javatpoint.com/agile-sdlc

[14]. Khan, T., Tian, W., Ilager, S., & Buyya, R. (2022). Workload forecasting and energy state estimation in cloud data centres: ML-centric approach. Future Generation Computer Systems, 128. https://doi.org/10.1016/j.future.2021.10.019

[15]. Khan, T., Tian, W., Zhou, G., Ilager, S., Gong, M., & Buyya, R. (2022). Machine learning (ML)-centric resource management in cloud computing: A review and future directions. In Journal of Network and Computer Applications (Vol. 204). https://doi.org/10.1016/j.jnca.2022.103405

[16]. Kumar, J., Singh, A. K., & Buyya, R. (2021). Self-directed learning based workload forecasting model for cloud resource management. Information Sciences, 543. https://doi.org/10.1016/j.ins.2020.07.012

[17]. LeadingIndiaAI. (2020). *Prediction-of-Dynamic-Cloud-Resources-Provisioning-for-Workflow*. GitHub. Retrieved August 20, 2024, from https://github.com/LeadingIndiaAI/Prediction-of-Dynamic-Cloud-Resources-Provisioning-for-Workflow/blob/master/AzureReadings_at_a_timestamp.csv

[18]. Lipton, Z. C. (2015). A Critical Review of Recurrent Neural Networks for Sequence Learning. https://www.researchgate.net/publication/277603865

[19]. Lutkevich, B. (2024). Waterfall Model. https://www.techtarget.com/searchsoftwarequality/definition/waterfall-model

[20]. Nasteski, V. (2017). An overview of the supervised machine learning methods. HORIZONS.B, 4. https://doi.org/10.20544/horizons.b.04.1.17.p05

[21]. Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. 30th International Conference on Machine Learning, ICML 2013, PART 3.

[22]. Rana, A., & Tanwar, G. (n.d.). Exemplifying Practical issues of Resource Management in Cloud Computing. www.ijert.org

[23]. Shyam, R., & Singh, R. (2021). A Taxonomy of Machine Learning Techniques. Advancements in Robotics, 8(3).

[24]. Spiliotis, E. (2023). Time Series Forecasting with Statistical, Machine Learning, and Deep Learning Methods: Past, Present, and Future (pp. 49–75). https://doi.org/10.1007/978-3-031-35879-1_3

[25]. Spiliotis, E. (2023). *Time Series Forecasting with Statistical, Machine Learning, and Deep Learning Methods: Past, Present, and Future* (pp. 49–75). https://doi.org/10.1007/978-3-031-35879-1_3

[26]. Srivastava, A., & Kumar, N. (2020). Resource management techniques in cloud computing: A state of art. ICIC Express Letters, 14(9). https://doi.org/10.24507/icicel.14.09.909

[27]. Utmal, M. (2021). Taxonomy on Machine Learning Algorithms. International Journal of Recent Development in Engineering and Technology Website: Www.Ijrdet.Com, 10(8).

[28]. Van Houdt, G., Mosquera, C., & Nápoles, G. (2020). A review on the long short-term memory model. Artificial Intelligence Review, 53(8). https://doi.org/10.1007/s10462-020-09838-1

[29]. Velicer, W. F., & Molenaar, P. C. (2012). Time Series Analysis for Psychological Research. In Handbook of Psychology, Second Edition. https://doi.org/10.1002/9781118133880.hop202022