Cluster Computing
https://doi.org/10.1007/s10586-024-04562-9

CloudAIBus: a testbed for AI based cloud computing environments

Sasidharan Velu[1] · Sukhpal Singh Gill[1] · Subramaniam Subramanian Murugesan[1] · Huaming Wu[2] · Xingwang Li[3]

Received: 20 February 2024 / Revised: 3 May 2024 / Accepted: 8 May 2024

Abstract
Smart resource allocation is essential for optimising cloud computing efficiency and utilisation, but it is also very challenging as traditional approaches often overprovision CPU resources, leading to financial inefficiencies. Recently developed Artificial Intelligence (AI) techniques have the potential to solve this problem efficiently; for example, deep learning models can accurately forecast how resources will be used, allowing for more efficient distribution of those resources. Despite these encouraging breakthroughs, researchers have not thoroughly investigated these AI models' dynamic scaling potential. To address this gap, we developed a new testbed for an AI-driven cloud computing environment called **CloudAIBus** for effective resource allocation. CloudAIBus employs a deep learning model named DeepAR to provide a robust solution for forecasting CPU usage in order to make cost-effective resource allocation decisions. Furthermore, we implement the DeepAR model using Amazon SageMaker, a robust platform that provides the infrastructure for scalable and efficient training. We evaluated the performance of the DeepAR-based resource management approach (CloudAIBus) using Google Colab, and results show that the proposed approach offers better performance than baselines (LSTM and ARIMA-based resource management) in terms of Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE) and Mean Squared Error (MSE). The proposed approach cut the percentage of unused CPUs from 98.65 to 32.35% compared to the GWA-T-12 dataset. This showed that it was effective at reducing over-provisioning by making accurate predictions.

Keywords Cloud computing · Artificial intelligence · Deep learning · Resource allocation

1 Introduction

The evolution of cloud computing has increasingly harnessed Artificial Intelligence (AI) to enhance resource management, addressing the dynamic and often unpredictable demands of modern cloud environments [1]. Research by Rao (2023) underscores the efficacy of AI-driven strategies in optimizing cloud resources, highlighting significant enhancements in both performance and cost-efficiency [2].

Background and recent advancements: In the realm of decision-making, Wang et al. [3] have demonstrated the utility of data envelopment analysis for assessing efficiency within cloud computing marketplaces, providing a robust framework for evaluating cloud service providers. Additionally, the development of HUNTER [4] and HunterPlus [5], AI-based holistic resource management systems, marks a significant advance in sustainable cloud computing by optimizing energy efficiency. These efforts have improved

✉ Sukhpal Singh Gill
s.s.gill@qmul.ac.uk

Sasidharan Velu
s.velu@se22.qmul.ac.uk

Subramaniam Subramanian Murugesan
s.subramanianmurugesan@se22.qmul.ac.uk

Huaming Wu
whming@tju.edu.cn

Xingwang Li
lixingwang@hpu.edu.cn

[1] School of Electronic Engineering and Computer Science, Queen Mary University of London, London, UK

[2] Center for Applied Mathematics, Tianjin University, Tianjin, China

[3] School of Physics and Electronic Information Engineering, Henan Polytechnic University, Jiaozuo, China

Published online: 06 June 2024

energy efficiency, but they have not investigated the impact of these advancements on economics and environmental aspects [6].

Economic and environmental insights: The economic ramifications of resource allocation strategies are profound [3–5], with strategic missteps potentially leading to substantial inefficiencies [6]. Niyato et al. [7] provide an economic analysis highlighting how resource market dynamics, including monopoly and oligopoly settings, can influence cloud computing environments, affecting the economic decisions of service providers.

Existing challenges: As cloud services become increasingly complex, the need for sophisticated resource management solutions becomes paramount [8]. Jeyaraj et al. [9] discuss the integration of cloud computing with Internet of Things (IoT) technologies, emphasizing the critical role of efficient resource management in supporting IoT applications.

Proposed solution: Against this backdrop, we introduce a new testbed, called CloudAIBus, which leverages the DeepAR model for predictive resource allocation. This innovative approach aims to address the highlighted economic and environmental challenges by providing a scalable and adaptable solution that significantly enhances the efficiency and sustainability of cloud operations. CloudAIBus promises to transform cloud resource management by integrating advanced AI models that not only enhance predictive accuracy but also operational efficiency. This initiative aligns with global efforts to achieve sustainable IT practices, significantly reducing the environmental footprint of cloud services and optimizing economic outcomes. In sum, CloudAIBus and its underlying technologies offer substantial implications for the cloud computing industry, spanning financial, operational, and environmental aspects. The introduction of this system provides a timely response to the critical needs of modern cloud service environments, promising significant advancements in resource allocation and management.

1.1 Motivation and our contributions

As cloud computing environments become increasingly dynamic, traditional static models struggle to cope with fluctuating demands effectively, leading to resource wastage and increased operational costs [10]. While existing models have significantly improved resource allocation in cloud computing [11], there remains an underexplored potential in dynamically scaling AI models to adapt to varying workload demands and environmental sustainability considerations [12]. Dynamic resource allocation is crucial for optimizing cloud resource scaling [13], as evidenced by research leveraging Amdahl's Law and queueing theory to model service time and performance scaling [14]. Proactive scaling techniques using prediction models, such as those implemented in Kubernetes for auto-scaling worker nodes [15], have shown potential to improve utilization and reduce power consumption, supporting dynamic and AI-driven workloads efficiently and sustainably [16, 17]. Moreover, time series forecasting models have been employed to predict cloud workloads and adjust resources accordingly, maintaining optimal operation and minimizing environmental impact [18]. Another research work is utilizing Long Short-Term Memory (LSTM) for predictive scaling has also demonstrated enhanced resource allocation efficiency and energy consumption reduction by dynamically adjusting compute resources based on predicted workloads [19]. Although these works have improved energy efficiency, they fail to consider how these advancements impact economics and environmental factors [20].

This paper addresses the above-mentioned gaps by introducing a novel testbed for an AI-driven cloud computing environment, termed **CloudAIBus**, which utilizes the DeepAR model for predictive resource allocation. The motivation behind CloudAIBus stems from the necessity to enhance cost efficiency, service quality, and environmental sustainability in cloud resource management. CloudAIBus has utilized the potential of deep learning or generative AI models for predicting resource allocation in cloud computing while taking into account the importance of the statistical property of stationarity in time series data. The main contributions of this research are:

- Develop a new testbed for AI-based resource management in cloud computing environments, called *CloudAIBus*, for cost-effective and sustainable cloud computing.
- Utilize the DeepAR model in CloudAIBus for forecasting CPU usage, which would help to make effective resource allocation decisions as this is a critical factor in cloud resource management.
- Implement the DeepAR model on the Amazon SageMaker, a robust platform that provides the infrastructure for scalable and efficient training.
- Evaluate the performance of the CloudAIBus approach using Google Colab by utilizing a dataset of 1750 VM traces from Bitbrains.
- Compare the performance of the CloudAIBus approach with existing approaches (LSTM and AutoRegressive Integrated Moving Average (ARIMA)-based resource management) in terms of Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE) and Mean Squared Error (MSE).
- Measure the performance of the proposed approach against the GWA-T-12 dataset in terms of unused CPUs.

When tested on both stationary and non-stationary data, the DeepAR model-based proposed approach, called CloudAIBus, has demonstrated versatility and robustness, providing accurate and reliable predictions for different types of time series data. Prior to employing these models, we conducted stationarity tests to understand the nature of the time series data and identify the challenges and opportunities it presents for each model. This step is crucial to ensuring the accuracy and robustness of our predictions, given the importance of stationarity in many forecasting models [6]. This comparison is centered around key performance metrics, such as use, MAE, MSE, and MAPE, providing a comprehensive evaluation of each model's effectiveness in the domain of resource allocation prediction while considering the issue of stationarity [21]. Our contributions are twofold: First, we integrate the DeepAR model into cloud resource management, demonstrating its superiority over traditional LSTM and ARIMA models through comprehensive performance evaluations. Second, we implement this model on Amazon SageMaker, ensuring scalable and efficient training that aligns with real-world cloud computing demands. Practically, CloudAIBus significantly reduces CPU over-provisioning-from 98.65% to 32.35% on the GWA-T-12 dataset-demonstrating its effectiveness in enhancing resource utilization and reducing unnecessary expenditures. Such improvements pave the way for more sustainable cloud computing practices, ultimately contributing to reduced environmental impacts.

Lightweight testbed: This study aims to contribute to the growing discourse around AI in cloud computing by assessing the performance of these models for resource allocation prediction, with a particular focus on their handling of non-stationary data. The goal is to provide valuable insights for cloud service providers and develop a lightweight testbed for AI-driven cloud computing services that are more efficient, sustainable, and cost-effective.

1.2 Article organization

The rest of the paper is structured as follows: Section 2 discusses the related work. Section 3 presents a real-world case study of adaptive resource management for a global streaming service. Section 4 introduces the proposed CloudAIBus system and provide detailed explanation. Section 5 presents a performance evaluation and experimental results. Section 6 discusses prediction accuracy, efficiency in resource allocation using the DeepAR model, and implications for scalability and environmental sustainability. Section 7 concludes the paper and highlights possible extensions of CloudAIBus testbed.

2 Related work

In this research, extensive investigations have been conducted on the application of machine learning and statistical models for optimal resource allocation [22]. This literature primarily focuses on three critical aspects: (i) the stationarity of data, (ii) the predictive algorithms utilized, and (iii) the strategy for resource allocation, categorized as proactive or reactive [23]. Stationarity plays a crucial role in enhancing the accuracy of many forecasting models, as consistent statistical properties over time are essential for reliable predictions [24]. The distinction between proactive and reactive allocation methods determines the system's responsiveness to dynamic workloads [25]. This section offers a detailed review of contemporary studies that delve into these aspects within the cloud resource allocation framework, with a summarized overview presented in Table 1.

Several researchers have employed ARIMA and Autoregressive Moving Average (ARMA) models, focusing on data stationarity. Zhang et al. [26] proposed an intelligent workload factoring scheme for Virtual Machine (VM)-based hybrid cloud computing, employing the ARIMA technique. However, the shift towards container-based systems and the often non-stationary nature of real-world cloud workloads raise questions about the model's applicability. Fang et al. [27] developed a resource prediction and provisioning scheme (RPPS) for container-based systems, utilizing the ARMA technique. Their approach assumes stationarity in the dataset, which may not always hold in real-world cloud workloads. The RPPS also includes a mechanism for dynamically adjusting resources but may struggle with sudden load spikes. Ciptaningtyas et al. [28] presented a proactive dynamic provisioning architecture leveraging the ARIMA model, achieving an impressive 91% in accuracy. However, the assumption of stationarity in their dataset raises questions about the model's applicability to real-time workloads. Calheiros et al. [29] developed a system using the ARIMA model for predicting the number of requests, handling up to 57,750 requests with an error ratio of 7.83%. However, the paper does not explicitly discuss the stationarity of the dataset used. Kirchoff et al. [30] presented the advantages and disadvantages of three workload prediction techniques when applied in the context of cloud computing. Their preliminary results compared ARIMA, Multilayer Perceptron (MLP), and Gated Recurrent Units (GRU) under different cloud configurations to help administrators choose the more appropriate and efficient predictive model for their specific problem.

Other works have explored different techniques without explicitly discussing the stationarity of the dataset. Tang

Table 1 Comparison of proposed CloudAIBus with existing works

Work	Type	Features	AI Model	Strategy	Data stationarity	Tool	Metrics	Dataset
Zhang et al. [26]	VM	Request rate	ARIMA	Reactive	Stationary	Amazon EC2	Resource efficiency	Yahoo! Video workload traces
Fang et al. [27]	Container	CPU	ARMA	Reactive	Stationary	Xen & KVM	Prediction accuracy, Throughput	Synthetic workloads
Ciptaningtyas et al. [28]	VM	Request rate	ARIMA	Reactive	Stationary	Docker containers	Running time, CPU, Memory Usage	FIFA World Cup 98 website
Calheiros et al. [29]	Container	Request rate	ARIMA	Reactive	Stationary	CloudSim	Average service time, Prediction accuracy	Wikimedia Foundation
Prachitmutita et al. [35]	VM	Request rate	ANN, RNN	Proactive	Stationary	Docker containers	Root mean squared error (RMSE)	FIFA World Cup 98 website
Imdoukh et al. [37]	Container	Request rate	LSTM	Reactive	Stationary	Docker containers	Prediction accuracy	FIFA world cup 98 website
Tang et al. [31]	Container	CPU	Bi-LSTM	Proactive	Stationary	Kubernetes cluster	Prediction accuracy, RMSE	Traced from data center with 500 containers
Yan et al. [32]	Container	CPU, Memory	Bi-LSTM	Hybrid	Stationary	Kubernetes cluster	SLA conflicts and CPU Utilization	Alibaba Cluster Trace 2018
Toka et al. [36]	Container	Request rate	AR, HTM, LSTM	Reactive	Stationary	Kubernetes cluster	Number of lost requests	NASA-HTTP & FIFA World Cup 98
CloudAIBus TestBed	VM	CPU	DeepAR	Proactive	Both	Google Colab	Prediction accuracy, MAE, MSE, MAPE, CPU Usage, Execution time, Energy cost	Bitbrains

et al. [31] proposed a machine learning-based auto-scaling solution leveraging LSTM networks, but the paper does not explicitly discuss the stationarity of the dataset used, which might limit its direct applicability to research. Yan et al. [32] presented a novel approach to adaptive horizontal scaling using a Bidirectional LSTM (BiLSTM) model, but the dependency of the system's effectiveness on the prediction method could be a potential limitation. Anupama et al. [33] proposed a hybrid prediction model combining statistical and machine learning techniques, including SARIMA (Seasonal Auto-Regressive Integrated Moving Average) for seasonal workloads and LSTM or ARIMA for non-seasonal workloads. Their experimental results confirmed that the accuracy of the prediction of the LSTM model outperformed ARIMA for irregular workload patterns. Ashawa et al. [34] implemented an application of the LSTM algorithm, showing an enhanced accuracy rate by approximately 10-15% as compared with other models. Prachitmutita et al. [35] developed a cost-effective auto-scaling framework for microservices on Infrastructure as a Service (IaaS), leveraging Artificial Neural Network (ANN) and Recurrent Neural Network (RNN), with future work on improving resource planning. Toka et al. [36] made a significant contribution to the field of Kubernetes auto-scaling with their exploration of AI-based prediction models, providing a comprehensive evaluation of several auto-scaling methods.

2.1 Critical analysis

The above-mentioned studies have significantly contributed to the understanding and development of machine learning models for resource allocation in cloud computing. The detailed analysis of these works highlights the need for further research to adapt existing models to modern cloud systems, energy efficiency, fairness, and optimization of resource allocation. Exploring more precise and sophisticated prediction models, shifting towards proactive methods, and considering non-stationary data are promising directions for future work [38]. Existing studies have not fully explored the potential of these AI models in dynamic scalability scenarios, as shown in Table 1. This is particularly important in situations where there is an increasing number of users, which can significantly impact

resource allocation decisions [39]. Further, none of the existing studies have tested their approach on the Birbrains dataset or the Google Colab-based cloud environment. Furthermore, they did not consider both computing and AI parameters, as well as both types of data stationarity, when evaluating performance.

To address this gap, we developed a new testbed for an AI-driven cloud computing environment called **CloudAIBus**, which employs a deep learning model named DeepAR to provide a robust solution for forecasting CPU usage in order to make cost-effective resource allocation decisions. CloudAIBus is a VM-based proactive resource management framework, and we have deployed it on Google Colab, which excels in dynamic scalability scenarios. Specifically, we utilized the traces from the Birbrains dataset to evaluate Quality of Service (QoS)/computing metrics such as execution time, energy cost, and CPU usage, as well as AI parameters like prediction accuracy, MAE, MSE, and MAPE, while taking into account the stationarity of both types of data. We also compare CloudAIBus's performance to known benchmarks [29] [31], which shows that it has the potential to provide dynamic scalability in situations where the number of users grows, which could have a great impact on decisions about how to allocate resources.

3 Real-world case study: adaptive resource management for a global streaming service

A leading global streaming service experiences fluctuating demands due to varying viewer preferences, new content releases, and peak viewing times, such as during major sports events or new series premieres [40]. Managing compute resources dynamically to handle these fluctuations without interruption is crucial to maintaining a high-quality user experience and operational efficiency [41].

3.1 Implementation of adaptive resource management

The streaming company implements several of the discussed adaptive resource management strategies to optimize its cloud infrastructure, ensuring seamless service to millions of concurrent users worldwide.

- Using empirical prediction models: The company utilizes Neural Network and Linear Regression models, as suggested by [42], to predict server load based on historical viewership data and scheduled events. This prediction helps in scaling resources up or down automatically, ensuring adequate capacity during high-demand periods and conserving resources during off-peak times.
- Ensemble-Based Load Forecasting: Leveraging techniques from [43], the service employs an ensemble of models to forecast demand more accurately. This ensemble approach aggregates predictions from multiple models to handle unexpected load spikes during sudden viral content trends or unexpected global events.
- ARIMA for short-term traffic prediction: Short-term traffic predictions using the ARIMA model, as detailed by [44], are particularly useful for minute-to-minute resource adjustments, helping to handle sudden influxes of users, which are common during live events.
- Agent-based dynamic provisioning: The Maximo Application Suite (MAS)-Cloud system proposed by [45] inspires the deployment of intelligent agents that manage resources across different geographic regions, adapting to local demand surges without human intervention.
- QoS optimization: Applying fuzzy model predictive control (FMPC), as explored by [46], allows the streaming service to maintain stringent QoS standards, dynamically allocating bandwidth and compute power to preserve stream quality, even under variable network conditions.

The application of these adaptive resource management strategies has enabled the streaming service to:

1) Reduce operational costs by minimizing idle compute resources.
2) Enhance viewer satisfaction by reducing buffering and load times, even during peak traffic.
3) Improve scalability, seamlessly handling user growth and global expansion without sacrificing service quality.

This case study exemplifies how advanced predictive and adaptive techniques in resource management can be applied in high-demand, real-world scenarios. The streaming service's ability to dynamically adjust its resources not only optimizes costs but also ensures a consistent, high-quality user experience, demonstrating the practical benefits of the theoretical models discussed. The integration of sophisticated predictive models and real-time adaptive management systems represents a significant advancement in cloud computing. As technologies evolve, these systems will become even more crucial in managing the complexities of modern cloud environments, promising further improvements in efficiency and performance across various industries [40, 41]. Therefore, the proposed CloudAIBus testbed has the potential to deal with adaptive resource management for a global streaming service in a proactive manner.

4 CloudAIBus: proposed testbed

In this section, we discuss the proposed CloudAIBus testbed, including the system model, design & implementation, data pre-processing and DeepAR model-based resource allocation.

4.1 System model

The proposed system architecture consists of three main components: the Workload Analyzer, Workload Predictor, and Resource Manager. Together, they form a cohesive framework for managing resource demands within a cloud computing environment as shown in Fig. 1. The architecture integrates real-time data monitoring, probabilistic forecasting algorithms, and automated resource adjustment mechanisms. The cloud scheduler plays a vital role in this integration, initiating processes at regular intervals to facilitate data flow across the tiers. The goal is to optimize resource allocation in response to fluctuating workloads, addressing challenges such as over-provisioning and under-provisioning. The system's ability to accurately predict resource needs and make timely adjustments enhances resource utilization, system performance, and cost-effectiveness.

1) Workload analyser: It is the foundational layer of the system architecture, tasked with continuous monitoring and analysis of CPU utilization metrics. Activated by the cloud scheduler at regular intervals (every X minutes), it collects real-time data from the load balancer, maintaining a rolling 30-minute window of information. By identifying patterns and trends in the workload profile, the *Workload Analyzer* forms the basis for the predictive modeling phase. Its ability to discern complex workload behaviors makes it an essential component in the resource allocation model, setting the stage for accurate forecasting and efficient resource management.

2) Workload predictor (AI Module): It is the second tier of the system architecture, and leverages the data collected by the *Workload Analyzer* to forecast future workload demands. This work mainly focuses on this module to optimize the predictions. It employs the DeepAR model, a state-of-the-art probabilistic forecasting algorithm [47], capable of recognizing complex time-dependent patterns and providing uncertainty estimates for future predictions. Utilizing the most recent 30 min of CPU utilization data, the Predictor formulates accurate workload projections for the subsequent 30-minute period. The application of DeepAR equips the Workload Predictor with the ability to anticipate future resource needs, informed by both historical trends and current conditions. This enables a strategic shift in resource allocation from a reactive model, where resources are allocated based on immediate needs, to a proactive model that pre-allocates resources in alignment with predicted demands.

3) Resource manager: It is the final tier in the architecture, responsible for aligning CPU provisioning with forecasted workload demands. Utilizing the Workload Predictor's forecast data, it calculates the required modifications to CPU allocation based on the 90th percentile of the predicted workload, applying specific weighting factors derived from historical performance. The Resource Manager then communicates the revised CPU requirements to the Autoscaler/Load Balancer, enabling dynamic adjustments to match anticipated demand. This process ensures optimal resource utilization, avoiding both over-provisioning and under-provisioning. The Resource Manager's precise translation of predictive data into actionable resource decisions is central to the system's proactive resource management.

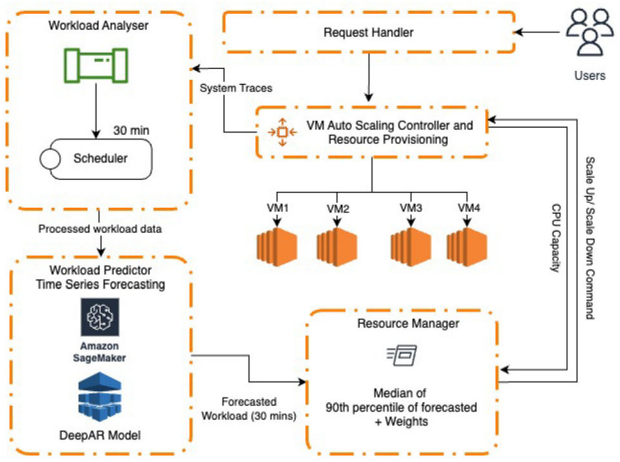

Fig. 1 Architecture of proposed CloudAIBus testbed

4.2 CloudAIBus design

This section provides UML diagrams (the sequence and class diagrams) to show the interaction components of the proposed CloudAIBus system, according to Fig. 1. Figure 2 shows the class diagram of the CloudAIBus Testbed, illustrating the object-oriented structure and relationships of the system components designed to optimize cloud resource management. This diagram provides a detailed view of the classes, their attributes, methods, and the interactions between them, which are crucial for the effective execution of cloud resource management tasks. The following are the main classes of CloudAIBus:

- Users class: Represents the system users who interact with the CloudAIBus environment. Attributes include `user_id: int`, and methods such as

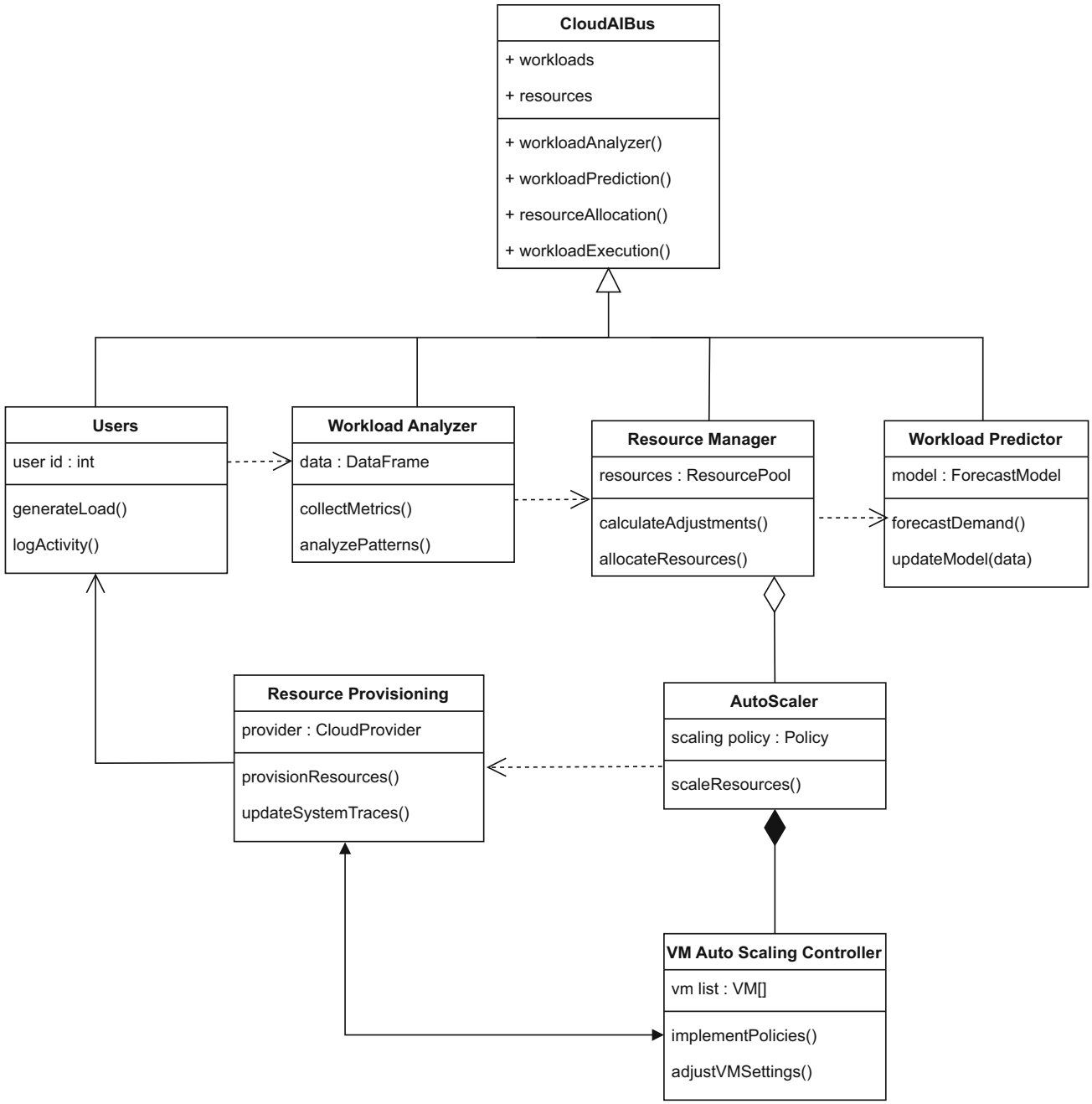

Fig. 2 Class diagram of CloudAIBus testbed

`generateLoad()` and `logActivity()`, which simulate user activities and interactions with the cloud system.

- Workload analyzer class: Critical for monitoring and analyzing the workload on cloud resources. It manages data through `data: DataFrame` and includes methods like `collectMetrics()` to gather system usage data and `analyzePatterns()` to identify trends and anomalies in resource usage.
- Workload predictor class: Utilizes predictive models to forecast future resource demands. It is equipped with `model: ForecastModel` and uses `forecastDemand()` to predict upcoming loads and `updateModel(data)` to continuously refine the predictive accuracy based on new data.
- Resource manager class: Responsible for the dynamic allocation of resources based on predictive insights. It manages `resources: ResourcePool` and employs methods such as `calculateAdjustments()` for

determining necessary resource changes and `allocateResources()` to execute these changes.

- Autoscaler class: Automatically adjusts the scaling of resources to meet the predicted demands, managed through `scaling policy: Policy`. Its primary method, `scaleResources()`, ensures that the resource levels are optimized for current and predicted workloads.
- VM auto scaling controller class: Directly manipulates VM configurations to align with scaling decisions, using `vm list: VM[]` to manage a list of active VMs and methods like `implementPolicies()` and `adjustVMSettings()` to update VM operations according to the autoscaling rules.
- Resource provisioning class: Handles the logistical aspects of resource management, including the provisioning and de-provisioning of cloud resources. It is associated with `provider: CloudProvider` and uses methods like `provisionResources()` and `updateSystemTraces()` to maintain an audit trail of resource allocation changes.

The class diagram encapsulates the modular architecture of the CloudAIBus testbed, highlighting how each component is designed to interact seamlessly to support scalable, efficient, and responsive cloud resource management. This structure promotes maintainability and scalability, ensuring that the system can adapt to changes in workload and technology over time.

Figure 3 shows the sequence diagram of the CloudAIBus testbed, detailing the operational workflow and the interactions between the various system components designed to efficiently manage cloud resources. This diagram captures the dynamic processes involved in monitoring, predicting, and managing the resource demands in cloud computing environments. The sequence begins with the **User** who initiates the process by generating workload and logging activities. This action triggers the **Workload Analyzer**, which is responsible for collecting real-time CPU utilization metrics at predetermined intervals. The collected data is then sent to the **Workload Predictor**.

In the **Workload Predictor**, an AI Module, the data undergoes preparation, including rolling window computation and feature extraction, crucial for accurate forecasting. The predictor updates or retrains its DeepAR model with this historical data to ensure the model reflects the latest patterns and trends.

Following the data preparation and model training, the predictor forecasts future CPU demands for the next 30-minute window. These predictions are crucial for proactive resource management. The forecast data is forwarded to the **Resource Manager**, which calculates the necessary resource adjustments based on the predicted demand and predefined weights.

The **Resource Manager** sends these calculated adjustments to the **Autoscaler**, which dynamically scales the system's resources up or down to match the forecasted demand. This scaling process is vital for optimizing resource utilization and cost-efficiency within the cloud environment.

Additionally, the sequence integrates the **VM Auto Scaling Controller** and **Resource Provisioning** components. The VM Auto Scaling Controller implements policies and adjusts VM settings to fine-tune the resource allocation further. Concurrently, Resource Provisioning handles the provisioning of VMs and other resources,

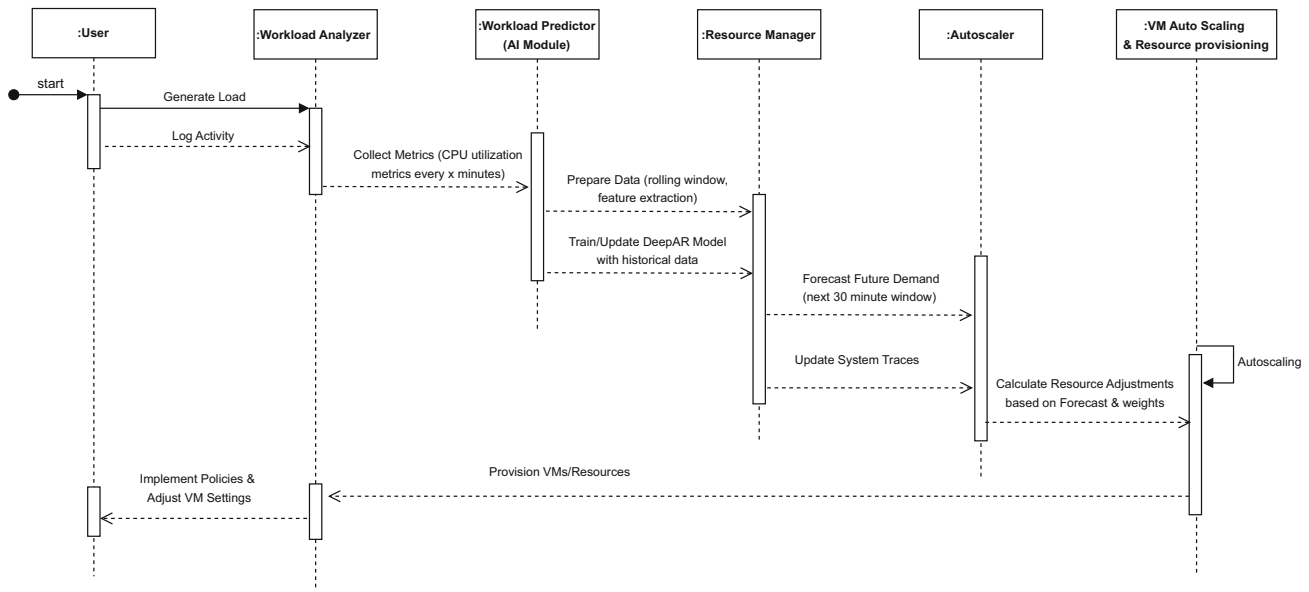

Fig. 3 Sequence diagram of CloudAIBus testbed

ensuring that the system's physical capabilities match the scaling decisions made by the Autoscaler.

Finally, the Resource Provisioning component updates the system traces, maintaining a log of all changes and operations, which is critical for audit and review purposes. This sequence diagram effectively outlines the automated and interactive processes of the CloudAIBus testbed, demonstrating the system's capability to adapt to changing workloads through intelligent data analysis, predictive modeling, and resource management strategies.

4.3 DeepAR based predictive resource recommendation

Algorithm 1 outlines the proposed architecture's workflow. It begins with the initialization of the three core components and the continuous collection of CPU utilization metrics C, summarized within a rolling window of size P as D_{patterns}. The DeepAR model then extrapolates future workload demands (D_{forecast}), used with current provisioning R_{current} and predefined weights W to ascertain adjustments $R_{\text{adjustments}}$. The median of the 90th percentile of these adjustments yields updated CPU requirements R_{median}. The Autoscaler is updated with R_{median}, aligning resources with anticipated demand. The process iterates, allowing dynamic adaptation to variable workloads.

Algorithm 1 DeepAR based predictive resource recommendation

Require: CPU utilization metrics C, prediction window $P = 30$ minutes, weights W
Ensure: Optimal resource allocation
Initialize Workload Analyzer, Workload Predictor, Resource Manager
while System is running **do**
  $C \leftarrow$ collectMetrics()
  $D_{\text{rolling}} \leftarrow$ rollingWindow(C, P)
  $D_{\text{patterns}} \leftarrow$ identifyPatterns(D_{rolling})
  $D_{\text{forecast}} \leftarrow$ deepAR(D_{patterns})
  $R_{\text{current}} \leftarrow$ getCurrentProvisioning()
  $R_{\text{adjustments}} \leftarrow$ calculateAdjustments(D_{forecast}, W)
  $R_{\text{median}} \leftarrow$ computeMedian($R_{\text{adjustments}}$, 0.9)
  updateAutoscaler(R_{median})
end while
function rollingWindow(C, P) **return** $\sum_{i=t-P}^{t} C_i$
end function
function deepAR(D_{patterns}) **return** $f(D_{\text{patterns}})$
end function
function identifyPatterns(D_{rolling}) **return** $g(D_{\text{rolling}})$
end function
function calculateAdjustments(D_{forecast}, W) **return** $W + D_{\text{forecast}}$
end function

4.4 Algorithm overhead analysis

The computational overhead, specifically the time complexity, of the proposed DeepAR based Predictive Resource Recommendation algorithm (Algorithm 1) is primarily influenced by the operations within its iterative loop, particularly the deep learning forecasting and resource adjustment calculations. This subsection details the analysis of the algorithm's overhead.

1) Metric collection and pattern identification: Collecting CPU utilization metrics (`collectMetrics()`) and identifying patterns (`identifyPatterns()`) depend on the volume of data and complexity of pattern detection techniques. The `rollingWindow()` function, operating in linear time relative to the window size P, has a complexity of $O(P)$.
2) Deep learning forecasting: The `deepAR()` function involves running a DeepAR model. The complexity is dependent on the model architecture, particularly the number of layers n, the number of neurons per layer m, and the input sequence length l, typically resulting in a time complexity of $O(n \cdot m \cdot l)$.
3) Resource adjustment calculations: Functions calculateAdjustments() and computeMedian() adjust resource provisioning based on forecasts, typically operating in $O(R)$, where R is the number of resources. Updating the autoscaler (`updateAutoscaler()`) involves system-level operations with complexity depending on cloud infrastructure specifics.
4) Loop and system interaction: The main `While` loop ensures continuous operation as long as the system is running, indicating a constant monitoring and updating overhead.
5) Overall complexity: The overall complexity is dominated by the `deepAR()` function, which is computationally intensive but optimized for performance on scalable cloud infrastructures like Amazon SageMaker.

The analysis indicates that while the algorithm is computationally intensive due to its deep learning operations, it is designed for efficient execution on scalable cloud platforms. This balance ensures the algorithm's practicality and efficiency in real-world environments, optimizing performance without excessive resource consumption.

4.5 Bitbrains' dataset based data pre-processing

The initial step in our research involved detailed analysis and pre-processing of a substantial dataset. We focused on the VM trace data from Bitbrains, containing records from

1500 VMs. This data serves as the foundation of our forecasting models, which include LSTM, ARIMA, and DeepAR. The Bitbrains VM trace data provides a comprehensive overview of the system's CPU utilization and capacity provisioning. The dataset includes information about the CPU capacity that was provisioned and the CPU that was used, over a time period ranging from '2013-07-01' to '2013-10-01' as shown in Fig. 4

1) Data analysis and interpretation: Our preliminary data analysis was based on three key statistical measures - median CPU usage, median CPU capacity provisioned, and the 95th percentile of CPU usage. The median CPU usage was found to be approximately 4,554,176 MHz, which is a central value separating the higher and lower halves of CPU usage data. On the other hand, the median CPU capacity provisioned during the same period was notably larger at approximately 45,484,316 MHz. This suggests that the system's CPU capacity was generally overprovisioned relative to its actual use.

To examine the utilization further, we calculated the 95th percentile of CPU usage, which stood at approximately 11,136,863 MHz This percentile indicates that 95% of the observed CPU usage values were less than or equal to this value, while in 5% of the instances, CPU usage was higher. The data analysis provides a detailed picture of the system's CPU utilization pattern. A cursory look at the median values of CPU usage and provisioned capacity shown in Fig. 4 suggests potential overprovisioning. The provisioned capacity was far more than the average usage, leading to the possibility of resource wastage.

However, a deeper look at the 95th percentile value offers a more nuanced understanding. By provisioning capacity based on the 95th percentile value, the system can efficiently manage peak CPU usage for 95% of the time. It's a potentially more cost-effective approach to balance between resource utilization and capacity provisioning.

2) Test for stationarity: Upon identifying potential overprovisioning in the CPU, we proceeded to further examine the stationarity of our time series data - a crucial assumption for many time series forecasting models that allows for accurate prediction of future values based on historical patterns. To achieve this, we conducted the Augmented Dickey-Fuller (ADF) test, which is a common technique for verifying the stationarity of time series data. In the ADF test, the null hypothesis assumes that the series possesses a unit root and is non-stationary. The results obtained from the test are given in Table 2.

The *p*-value, a feature of the null hypothesis, indicates the probability of obtaining a test statistic that is as severe or even more extreme than the reported one. Given that the test statistic is less than all the critical values at the 1%, 5%, and 10% levels, and that the calculated *p*-value is significantly smaller than 0.05, the basis exists for the rejection of the null hypothesis. This, in turn, contributes substantial empirical support to counter the premise of non-stationarity within the series. Thus, a decisive inference can be drawn, affirming the stationarity of the observed CPU usage data. To complement these findings and to provide a more intuitive understanding, we generated a plot showcasing the original time series data along with its rolling mean and standard deviation.

In Fig. 5, the blue line corresponds to the original time series, while the red line represents the rolling mean, and the black line denotes the rolling standard deviation. Stationarity demands that these lines remain approximately constant over time. While the original time series (blue line) may exhibit some fluctuations, the red and black lines appear stable, suggesting that the rolling mean and standard

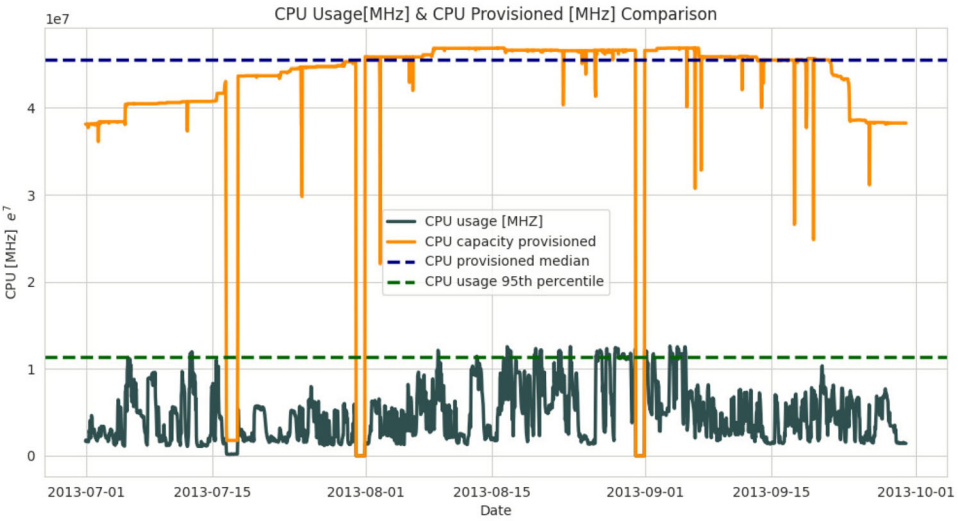

Fig. 4 CPU usage vs CPU provisioned

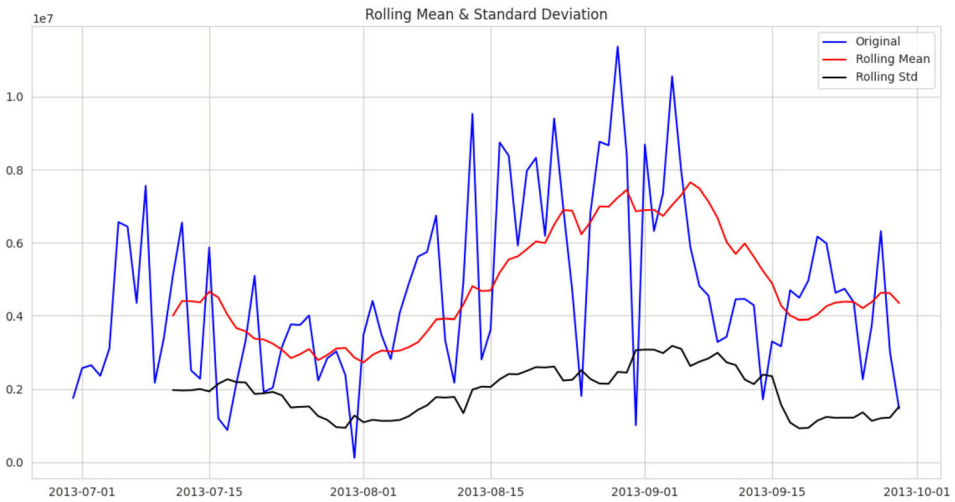

Fig. 5 ADF test: rolling mean and standard deviation

Table 2 Augumented Dickey-Fuller test

Measure	Value
Test statistic	− 6.63
p-value	5.65e− 09
Lags used	12
Critical value (1%)	− 3.43
Critical value (5%)	− 2.86
Critical value (10%)	− 2.57

deviation of the CPU usage do not change significantly over time. Therefore, both visually and statistically, we can affirm that the time series data of CPU usage is stationary. This conclusion paves the way for accurate forecasting using time series analysis methods, reinforcing the reliability and validity of the subsequent steps of our research.

3) Seasonality and trend analysis: To gain a deeper understanding of the CPU usage data, we performed a seasonal decomposition analysis, which revealed distinct patterns in the seasonality and trend components of our time series. Figure 6 shows the seasonality and trend analysis.

- Seasonality: The seasonal component displayed regular, repeating patterns with fluctuations ranging between the interval of [-1, 1]. This indicates the presence of significant seasonality in our data. The observed patterns suggest that CPU usage experiences predictable increases or decreases during specific times of the day, week, or year. The stability of the seasonality over time implies that the timing and magnitude of these fluctuations remain consistent. This characteristic is valuable for forecasting, as we can leverage the seasonality component to make accurate predictions about future CPU usage. It is important to note that while the magnitude of the seasonal fluctuations appears relatively small, their impact on our forecasts can be significant, particularly if the overall CPU usage values are also small. Therefore, incorporating this seasonality into our chosen forecasting model is crucial.
- Trend: The trend component of the data exhibited a random pattern without a clear direction. This randomness in the trend suggests two possibilities. Firstly, it could indicate the absence of an underlying trend in the CPU usage data, apart from the seasonal fluctuations. In other words, CPU usage does not consistently increase or decrease over time. Secondly, it could suggest the presence of a non-linear or complex trend that the seasonal decomposition method fails to capture adequately. Non-linear trends might include quadratic or exponential patterns, while complex trends could involve multiple interacting factors.

The apparent randomness in the trend component further reinforces the observation that our data is relatively stationary, without a strong linear trend. This aligns with the results of the Dickey-Fuller test, which also indicated stationarity in our CPU usage data. However, it is important to consider that many time series forecasting models, such as ARIMA, are designed to perform optimally with data that exhibits a linear trend. In the presence of a non-linear or complex trend, alternative models or data transformations may be necessary to effectively capture and incorporate this trend into our forecasting process.

4.6 DeepAR model-based resource allocation

The DeepAR model, as depicted in Fig. 7, is built on an autoregressive recurrent network architecture. The model

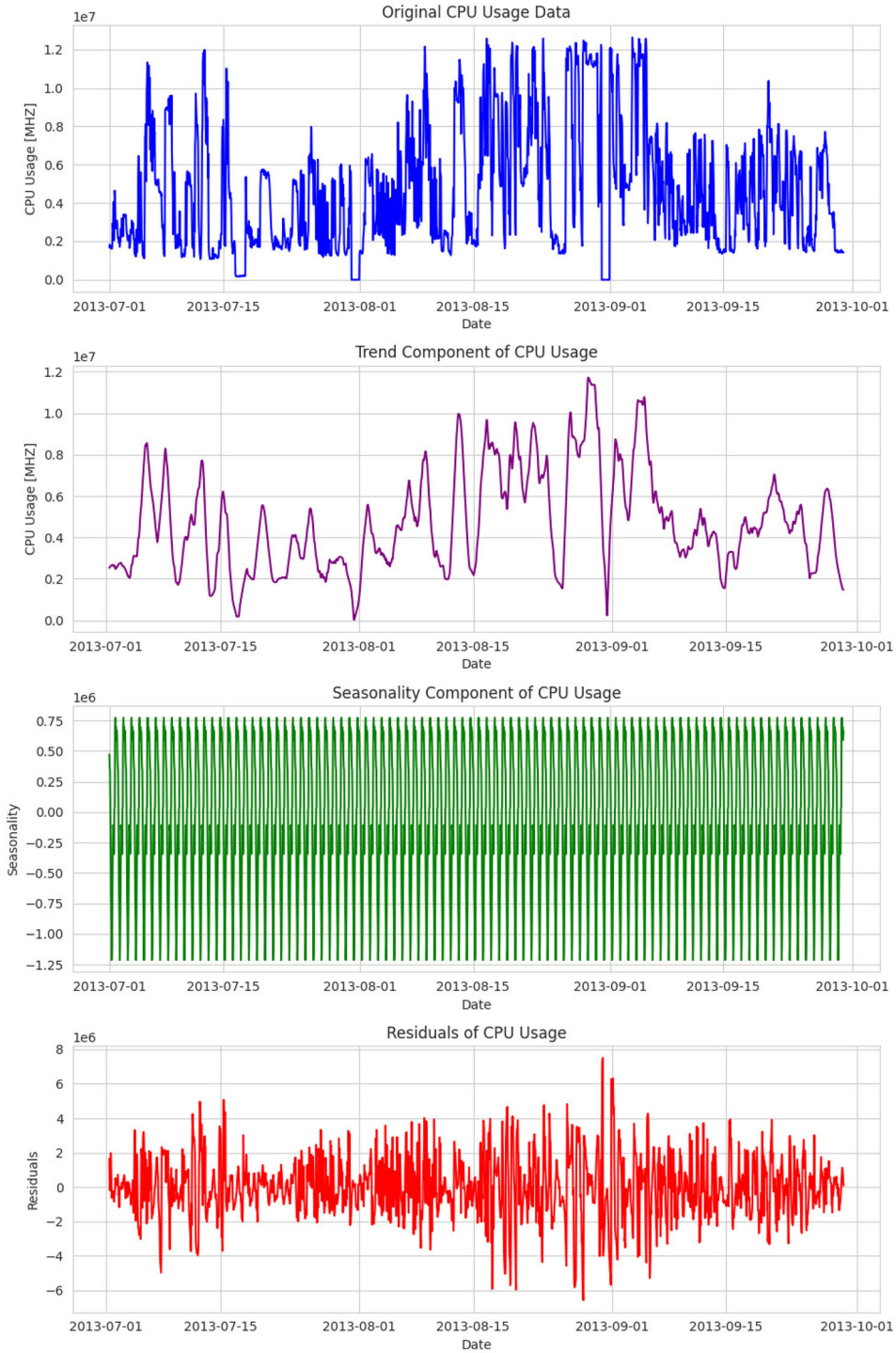

Fig. 6 Seasonality and trend analysis

distribution $Q_\Theta(z_{i,t_0:T}|z_{i,1:t_0-1}, x_{i,1:T})$ is assumed to be a product of conditionals, where each conditional distribution is parameterized by a function of the previous target values and covariates [47].

1) Parameter estimation: In the context of predictive resource allocation in cloud computing, the DeepAR model is trained on a dataset of time series $\{z_{i,1:T}\}_{i=1,\ldots,N}$ representing historical resource usage data and associated covariates $x_{i,1:T}$, which could include factors such as time of day, day of the week, and workload type. The parameters Θ of the model are learned by maximizing the log-likelihood, which is a measure of how well the model's predictions align with the actual observed data [47].

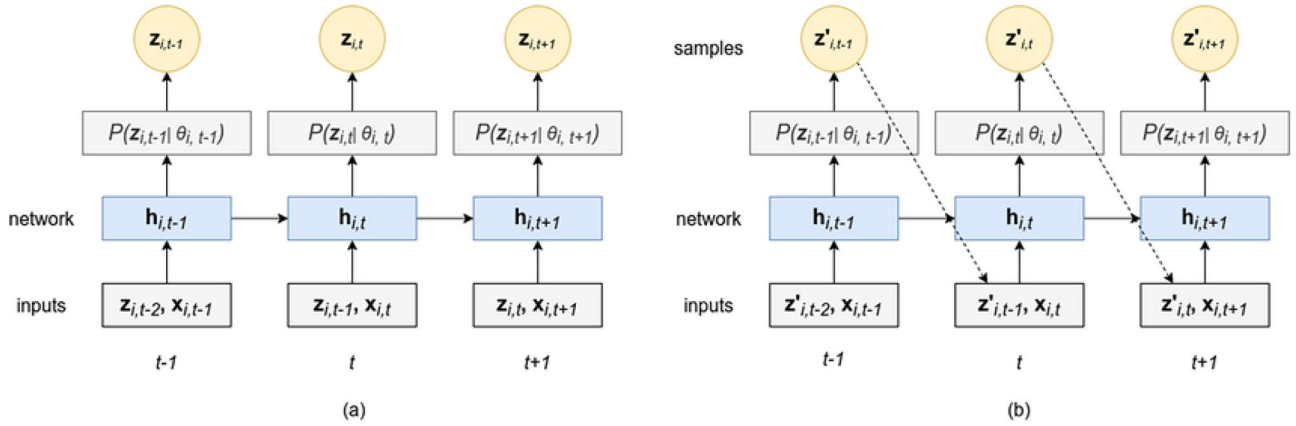

Fig. 7 DeepAR model architecture

$$L = \sum_{i=1}^{N} \sum_{t=t0}^{T} \log \ell(zi, t|\theta(hi, t)), \tag{1}$$

where $h_{i,t}$ is a deterministic function of the input. All quantities required to compute this equation are observed, so no inference is required, and the log-likelihood can be optimized directly via stochastic gradient descent by computing gradients with respect to Θ.

The training process generates multiple instances from each time series by selecting windows with different starting points, ensuring that the entire prediction range is always covered by the available ground truth data. This approach allows the model to learn the behavior of "new" time series, taking into account all other available features, which is crucial for predicting resource requirements for new workloads or services in the cloud [47].

2) Scale handling: In a cloud environment, resource usage can exhibit a wide range of scales due to the diverse nature of workloads and services. This variability in scale follows a power-law distribution, which can pose significant challenges for predictive models.

The DeepAR model addresses this issue by incorporating an item-dependent scale factor, v_i. Specifically, the autoregressive inputs $z_{i,t}$ (or $\tilde{z}_{i,t}$) are divided by this scale factor. This normalization step ensures that the inputs to the model are within a similar range, regardless of the original scale of the resource usage data.

Simultaneously, the scale-dependent likelihood parameters are multiplied by the same scale factor v_i. This adjustment ensures that the model's predictions are appropriately scaled to match the original scale of the resource usage data.

$$\mu = v_i \log(1 + \exp(o_\mu)) \text{ and } \alpha = \log(1 + \exp(o_\alpha))/\sqrt{v_i}, \tag{2}$$

where μ and α represent the parameters of the likelihood function, and o_μ and o_α are the outputs of the network for these parameters. By adjusting these parameters based on the scale factor, the DeepAR model can effectively handle data that spans a wide range of scales.

This scale-handling mechanism is crucial for predictive resource allocation in cloud computing. It allows the model to accurately predict resource requirements for a wide variety of workloads and services, regardless of their scale. This contributes to efficient resource allocation and improved performance in cloud environments [47].

5 Performance evaluation

This section discusses the dataset preparation, model training, experimental setup, evaluation metrics, and results.

5.1 Dataset preparation

In our study, we leverage a rich dataset derived from the Grid Workload Archive (GWA) to predict CPU utilization [48]. The data collection methodology ensures the inclusion of diverse patterns, trends, and fluctuations in CPU usage, providing a robust foundation for our model. The raw data consists of multiple time series, each representing a unique VM. For each VM, there are recordings of CPU usage over specific intervals along with a timestamp. This structured data collection over a continuous time frame enables us to perform time series forecasting. Our initial data cleaning efforts involved parsing timestamps to a suitable format and handling missing data. Timestamps were converted from milliseconds to a more standard format, facilitating easier manipulation and interpretation of the data. Any gaps in the series were filled using forward-

fill imputation to maintain data continuity and avoid loss of information critical to model performance.

Furthermore, we conducted feature engineering by extracting additional information from the dataset. This involved the creation of new columns indicating the day of the week, whether the day falls on a weekend, as well as the specific month and day associated with each observation. This augmentation aimed to enable our model to potentially identify any underlying seasonality or periodic patterns in CPU usage. In addition to these temporal features, we computed and incorporated the past values of CPU usage, as well as the differences in CPU usage, network received throughput, and network transmitted throughput into the dataset. This comprehensive feature set was designed to empower the model to capture dependencies or autocorrelations within the series.

5.2 Model training and validation

For training the DeepAR model, we set up an instance on Amazon Sagemaker, a robust platform that provides the infrastructure for scalable and efficient training. We used the pre-built container for the DeepAR algorithm available in the specified region. Hyperparameters for the model, including the frequency of the time series, prediction and context lengths, number of layers and cells in the RNN, the likelihood function, mini-batch size, learning rate, dropout rate, and early stopping patience, were set according to our problem requirements and computational resources. The training data, formatted as a JSON Lines file, was uploaded to an S3 bucket. The model was then trained on this data, with SageMaker handling the provisioning of resources.

1) Data preprocessing: It is a critical step in ensuring the reliability of our forecasting models. For our study, we employed a comprehensive preprocessing strategy that included:

- Handling missing data: We used forward-fill imputation to address any gaps in the data, which helps maintain the continuity of time series without introducing bias.
- Feature engineering: We extracted temporal features such as the day of the week and the presence of holidays, which are known to affect CPU usage patterns. We also utilized derived statistical features such as rolling means and standard deviations, which help in capturing longer-term trends and cyclic behaviors within the time series.

The importance of these preprocessing steps is supported by several studies, such as the work by Bassi, Gomekar, and Murthy [49], which emphasizes the role of statistical features in enhancing model performance for economic time series. Yeh et al. [50] further highlight the significance of preprocessing in building robust foundation models for time series by utilizing unlabeled data across domains.

2) Model configuration and training: The training of the DeepAR model was executed on Amazon SageMaker, leveraging its scalable infrastructure. The model was configured with specific hyperparameters tailored to the nuances of our dataset:

- RNN configuration: We opted for LSTM cells due to their efficacy in handling long sequence dependencies, which is crucial for forecasting tasks in dynamic environments like cloud computing.
- Hyperparameters: Settings such as the number of layers, dropout rates, and learning rates were tuned based on initial testing phases, ensuring optimal learning without overfitting.

This approach aligns with the findings of Ahmed et al. [51], who discuss the impact of different preprocessing methods and hyperparameter settings on the forecasting accuracy of various machine learning models.

3) Validation strategy: We adopted a rigorous validation strategy to ensure the robustness of our forecasts:

- Cross-validation: Employed a rolling window cross-validation technique to assess the model's performance over different time periods, enhancing the generalizability of our results.
- Performance metrics: Utilized MAE, MAPE, and MSE to evaluate model accuracy and reliability. These metrics are crucial for understanding different aspects of forecasting errors, as detailed by Venkatraman, Hebert, and Bagnell [52], who underscore their importance in reducing multi-step prediction errors through learned correction strategies.

To validate the efficacy of the DeepAR model, we compared its performance with traditional models like ARIMA and advanced neural network-based models such as LSTM. The choice of DeepAR was vindicated by its superior handling of non-stationary data and complex temporal dependencies, which is particularly challenging in cloud computing environments.

- Comparative analysis: Our results indicate that DeepAR provides more accurate and consistent predictions than the baselines [29] [31], particularly in scenarios with abrupt changes and non-linear patterns in the data. This supports the model's suitability for applications requiring high reliability and precision in forecasts.

The meticulous training and validation processes adopted in this study, supplemented by strategic data preprocessing and model tuning, have significantly enhanced the forecasting accuracy of the DeepAR model. This comprehensive approach, backed by contemporary research and best

practices in machine learning, ensures that our model can effectively address the complex challenges in predicting CPU utilization in cloud computing environments.

5.3 Evaluation metrics

To evaluate the performance of our DeepAR model-based resource allocation, we employ three widely used metrics: MSE, MAE, and MAPE. These metrics provide a comprehensive understanding of the model's accuracy and error distribution.

a) Mean absolute error (MAE): It quantifies the average magnitude of the errors in a set of forecasts, without considering their direction. It is calculated as:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y_i}|, \tag{3}$$

where y_i are the actual values and $\hat{y_i}$ are the predicted values. MAE is particularly effective for scenarios where all errors are equally important. Its simplicity and clarity in interpretation make it indispensable, especially in studies requiring a straightforward representation of forecasting accuracy.

b) Mean absolute percentage error (MAPE): It is the average of the absolute percentage errors and is expressed as:

$$\mathrm{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y_i}}{y_i}\right|, \tag{4}$$

This metric provides a normalized measure of error, allowing for meaningful comparisons across data with varying scales and magnitudes. MAPE is particularly valuable for its ability to provide a scale-independent measure of error, which is crucial for studies comparing performance across different datasets or market segments.

c) Mean squared error (MSE): It measures the average of the squares of the errors:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2, \tag{5}$$

MSE is highly sensitive to large errors, which makes it suitable for applications where the avoidance of large prediction errors is critical. This sensitivity to outliers can be crucial in financial and risk management applications where large errors can have disproportionately large consequences.

5.3.1 Contextual justification for metric selection

The choice of MAE, MAPE, and MSE in this study is strategically aligned with the need to comprehensively evaluate forecast accuracy and reliability across various scenarios and scales. These metrics were selected due to their demonstrated utility in numerous similar studies, as evidenced by their application in different forecasting contexts:

- MAE and MAPE in seasonal time series forecasting: As reported by Tseng, Yu, and Tzeng [53], these metrics were critical in assessing the accuracy of hybrid models in seasonal time series, highlighting their utility in handling data with inherent seasonal fluctuations. Their study not only validates the effectiveness of these metrics in assessing forecast accuracy but also underscores their capacity to differentiate between model performances in complex seasonal patterns.
- MSE in financial forecasting: Isiaka et al. [54] employed MSE to evaluate Autoregressive Moving Average (ARMA) models in forecasting exchange rates, underlining MSE's relevance in financial applications where the impact of large errors can be substantial. Their use of MSE reflects its importance in financial settings, where precision in prediction and the minimization of large errors are critical for effective risk management.
- Comprehensive model evaluation: In a study combining SARIMA with neural networks, Tseng, Yu, and Tzeng [55] utilized all three metrics to demonstrate the superior performance of hybrid models over traditional ones, validating their effectiveness in comparing complex model architectures. This study particularly highlights the versatility of these metrics in evaluating and comparing the performance of different types of forecasting models, from classical statistical models to modern machine learning approaches.

These metrics will serve as the foundation for evaluating the performance of the proposed CloudAIBus approach, i.e. DeepAR model-based resource allocation and facilitating a comparative analysis with the baseline resource allocation models using ARIMA [56] and LSTM [57]. The model that achieves the lowest values in terms of MSE, MAE, and MAPE will be designated as the preeminent and optimal-performing model. The inclusion of MAE, MAPE, and MSE in our evaluation framework ensures a balanced assessment of model performance, providing insights into average errors (MAE), relative errors (MAPE), and the severity of errors (MSE). This multi-faceted approach allows for robust comparisons across different models and contexts within our study, enhancing the reliability and validity of our conclusions.

5.4 Experimental setup: Google Colab

We deployed proposed (DeepAR) and existing (ARIMA and LSTM) models using Google Colab to evaluate the

performance of resource allocation in cloud computing environments. All models were trained on the same dataset of 1,500 VM traces from Bitbrains to ensure a fair performance comparison.

1) Platform description and rationale for selection: For the evaluation of our predictive models, including DeepAR, ARIMA, and LSTM, we chose Google Colab as our primary computational platform. Google Colab is a cloud-based environment that allows for the execution of Python code through Jupyter notebooks, which are hosted and run on Google servers.

a) Advantages : The following are the main advantages of Google Colab:

- Accessibility and Scalability: Google Colab provides a highly accessible platform with the advantage of using Google's robust hardware, including Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs), at no cost. This accessibility is particularly beneficial for data scientists and researchers requiring substantial computational resources without the need for significant infrastructure investment.
- Integration with Google Drive: Google Colab seamlessly integrates with Google Drive, making it easier to manage and share large datasets and notebooks, which facilitates collaboration among researchers.
- Pre-installed libraries: The environment comes pre-configured with most of the popular data science and machine learning libraries, reducing setup time and allowing for rapid prototyping and testing.

b) Reason for selection The choice of Google Colab was primarily motivated by its zero-cost access to high-performance computing resources, enabling us to train complex models more efficiently. Additionally, the ease of setup and use allowed our team to focus more on model development and less on computational issues. Studies by Carneiro et al. [58] and Gujjar and Kumar [59] provide detailed analyses of Google Colab's performance, noting its effectiveness in accelerating deep learning applications and its role in democratizing access to advanced computational tools.

2) Limitations: While Google Colab offers numerous advantages, several limitations were considered in the context of our study:

- Session timeouts: Google Colab sessions are limited to 12 h, after which all progress is lost if not saved externally. This aspect posed a challenge for training models that require extended computation times.
- Data Privacy and Security: Since the data is stored and processed in the cloud, there are inherent risks associated with data privacy and security. Sensitive data may require additional measures to ensure compliance with data protection regulations.
- Dependency on Internet Connectivity: Being an online platform, Google Colab requires a stable Internet connection. Interruptions in connectivity can lead to loss of work or interruptions in model training.

3) Impact on the Study: Despite these limitations, the use of Google Colab was deemed suitable for our study due to the balance between computational power and cost-efficiency. The platform enabled our team to efficiently train and evaluate complex models, providing a robust environment for comprehensive performance comparisons. However, considerations regarding session management and data security were meticulously managed to ensure the integrity and continuity of our research.

Google Colab served as an effective platform for the evaluation of our predictive models due to its computational capabilities and ease of use. While aware of its limitations, careful planning and management allowed us to leverage this environment to achieve insightful results without compromising the quality or security of our research.

5.5 In-depth analysis of the DeepAR model

This section delves deeper into the particular deep learning models utilised in the study, examining the DeepAR model's architecture and characteristics while drawing comparisons to other models such as LSTM and ARIMA.

1) Foundational Aspects of DeepAR: Introduced by Flunkert et al. [60], DeepAR is a probabilistic forecasting method that employs an autoregressive recurrent network architecture. It is designed to handle the inherent uncertainties in time series data by predicting the entire distribution of future values, rather than just single points. This approach allows for the quantification of prediction uncertainties, making DeepAR particularly useful in fields such as retail for inventory management and finance for risk assessment [61]. The following are the main features:

- Recurrent Neural Network (RNN): Utilizes LSTM or Gated Recurrence Unit (GRU) cells to capture time dependencies, enabling memory of past events for long sequences.
- Probabilistic Outputs: Instead of single point estimates, it outputs a probability distribution for each future time point, enhancing decision-making under uncertainty.
- Scalability: Capable of being trained across multiple related time series, improving its accuracy and generalization through shared learning.

2) Enhancements and Comparative Studies: The following are the enhanced DeepAR models:

- Liao and Liang [62] enhanced DeepAR for temperature forecasting by adding convolutional layers and LSTM within an encoder-decoder architecture, demonstrating improved speed and accuracy.
- Jeon and Seong [63] adapted DeepAR for intermittent time series by modifying the training process to enhance robustness, which is critical for irregular data patterns found in retail sales.
- Jungbluth and Lederer [64] developed the DeepCAR method that extends DeepAR with change point detection capabilities, crucial for managing abrupt shifts in time series data.
- Shi et al. [65] introduced a CNN-LSTM Attention DeepAR model that incorporates multi-scale and local dependencies, showing versatility and superior performance in complex forecasting environments.

The key differences between the DeepAR model and LSTM & ARIMA are as mentioned below:

- DeepAR vs. LSTM: Both utilize RNNs, but DeepAR's probabilistic forecasting nature provides detailed uncertainty estimates which are absent in standard LSTM models. This makes DeepAR more suitable for applications requiring detailed risk assessment.
- DeepAR vs. ARIMA: Unlike ARIMA, which requires data stationarity and primarily provides point estimates, DeepAR handles non-stationary data and outputs probability distributions, offering broader applicability and enhanced forecasting reliability.

3) Practical applications: The practical applications of DeepAR span across various sectors due to its robust and flexible architecture:

- In **retail**, it enhances inventory management by accurately predicting demand fluctuations.
- In **finance**, it aids in risk management by forecasting potential financial outcomes with quantified uncertainties.
- In **supply chain management**, it predicts potential disruptions, enabling proactive adjustments.

The DeepAR model represents a significant advancement in the field of time series forecasting. Its ability to produce probabilistic forecasts allows for better management of uncertainties, making it a valuable tool across various applications. The continuous enhancements and adaptations of the model further attest to its versatility and effectiveness in addressing complex forecasting challenges.

a) Architectural framework of DeepAR: DeepAR employs a sophisticated architecture primarily based on RNNs with options for LSTM or GRU cells, which are well-suited for modeling sequential data. The model is structured to process input data through multiple hidden layers, each capable of capturing different temporal dependencies within the data [66]. Inputs to the model typically involve sliding windows of past observations, which are transformed into features that feed into the RNN layers. The output layer of DeepAR is designed to predict parameters of a probability distribution (such as Gaussian, Negative Binomial, or others depending on the application), which provides the basis for its probabilistic forecasts. The training of DeepAR involves optimizing the parameters of the network to maximize the likelihood of the observed data, using gradient descent methods. This setup allows DeepAR to effectively model and forecast complex patterns in time series data, adapting its predictions to the inherent uncertainties and variabilities of real-world scenarios.

4) Real-world applications: The following are key real-world applications of DeepAR:

a) Demand forecasting in bike-sharing services Lim et al. [67] utilized DeepAR for station-wise demand forecasting in bike-sharing services. By employing an RNN-LSTM architecture, DeepAR was able to forecast parameters of distributions like normal, truncated normal, and negative binomial. The study demonstrated DeepAR's superior performance in capturing complex demand patterns and correlations between stations, significantly enhancing inventory management and rebalancing strategies.

b) Production forecasting in oilfields: Han and Xue [68] highlighted DeepAR's effectiveness in forecasting production time series for oilfields. Their study leveraged DeepAR's forward architecture based on an autoregressive recurrent neural network to predict the normal distribution of outputs. The model's ability to handle frequent changes and classify predictions in a dataset of over 2000 wells underscores its utility in complex, industrial contexts.

5.6 Performance comparison with baselines

The performance of the proposed (DeepAR) and existing (ARIMA and LSTM) models is evaluated in cloud computing environments.

1) ARIMA model-based resource allocation:

The ARIMA model, a popular method for time series forecasting, was first introduced by Box and Jenkins [56]. Despite its widespread use in various fields, the ARIMA model has limitations when applied to the context of predictive resource allocation in cloud computing [69]. Specifically, ARIMA models assume that the time series is stationary, i.e., its properties do not change over time. This assumption often does not hold in cloud computing environments, where resource usage can exhibit non-stationary behaviors such as trends or seasonality. The performance of the ARIMA model, as shown in Fig. 8, indicates that

while it can capture some of the temporal dependencies in the data, it struggles with more complex patterns and sudden changes in the series.

2) LSTM model-based resource allocation:

The LSTM model, a type of RNN, is capable of learning long-term dependencies in time series data [57]. It has been widely used in various tasks, including speech recognition, natural language processing, and time series forecasting. However, training an LSTM model can be computationally expensive and requires careful tuning of the hyperparameters [70]. Unlike ARIMA, LSTM does not require the data to be stationary, making it a more flexible choice for predictive resource allocation in cloud computing. However, as shown in Fig. 9, the LSTM model may still struggle with data that exhibits complex temporal dependencies. One of the main issues is the dynamic nature of cloud environments. Resource usage can fluctuate significantly over time due to varying user demands, making it difficult for LSTM models to predict future resource needs [57] accurately. [37]

In contrast to both ARIMA and LSTM, our proposed DeepAR model is designed to handle both stationary and non-stationary data, as well as capture complex temporal dependencies. It does this by leveraging both historical and covariate data, and by using a scalable and efficient training process on Amazon Sagemaker. This makes it a more flexible and robust solution for forecasting resource allocation in cloud computing environments, particularly in the context of non-stationary data.

3) LSTM and ARIMA-based resource allocation models:

We have selected the best baselines from the literature, which used ARIMA and LSTM for resource allocation in cloud computing, to test the performance of our proposed approach using the DeepAR model. Calheiros et al. [29] developed a system using the ARIMA model for predicting the number of requests, handling up to 57,750 requests with an error ratio of 7.83%. However, the paper does not explicitly discuss the stationarity of the dataset used. We chose another Tang et al. [31] paper that suggested a machine learning-based auto-scaling solution using LSTM networks. However, the paper doesn't talk about the stationary nature of the dataset used, which could make it less useful for research. These works explored different techniques without explicitly discussing the stationarity of the dataset.

Table 3 presents a comparative analysis of the ARIMA, LSTM, and DeepAR models based on three key performance metrics, namely, MAE, MSE and MAPE. Figures 10, 11 and 12 show the performance comparison of the proposed CloudAIBus approach (DeepAR) with baselines (ARIMA and LSTM) in terms of MAE, MSE, and MAPE, respectively. The ARIMA model shows a relatively high MAE of 10.86 and a substantial MSE of 511.39. This suggests that the model's predictions often deviate from the actual values, both on average (as indicated by the MAE) and in terms of larger errors (as indicated by the MSE). The MAPE of 29.61 further confirms that these errors represent a significant proportion of the actual values, which can be problematic in scenarios where precise predictions are crucial. The LSTM model, while demonstrating a lower MAE of 5.082, exhibits a considerably higher MSE of 1415.5. This indicates the presence of larger errors in the LSTM's predictions, which can be particularly detrimental in resource allocation tasks where over-provisioning or under-provisioning of resources can lead to significant costs or service disruptions. Furthermore, the high MAPE of 69.5 suggests that these errors are not just outliers, but represent a consistent over- or under-estimation of the actual values by the LSTM model. Our proposed DeepAR model, in contrast, strikes a balance between these metrics. While its MAE of 19.25 is higher than that of the LSTM model, indicating a higher average deviation

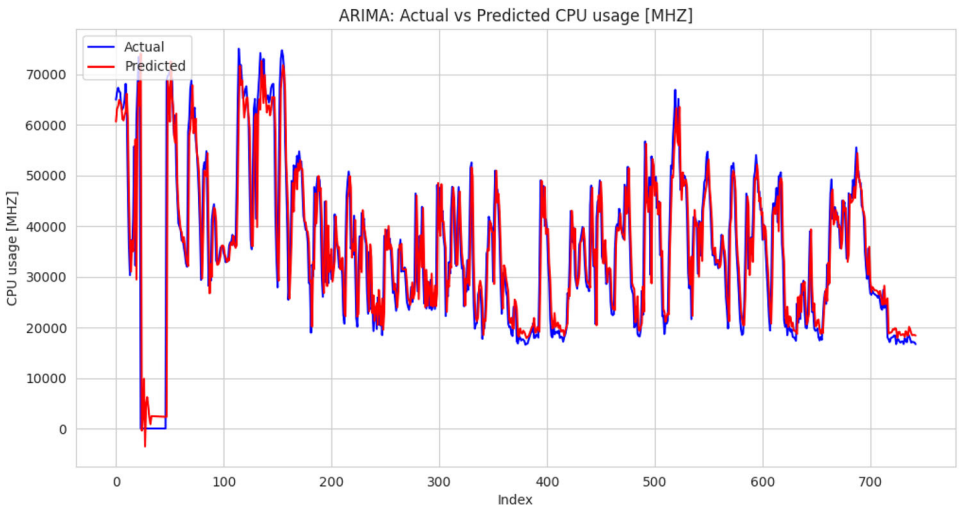

Fig. 8 ARIMA model (Actual vs. Predicted)

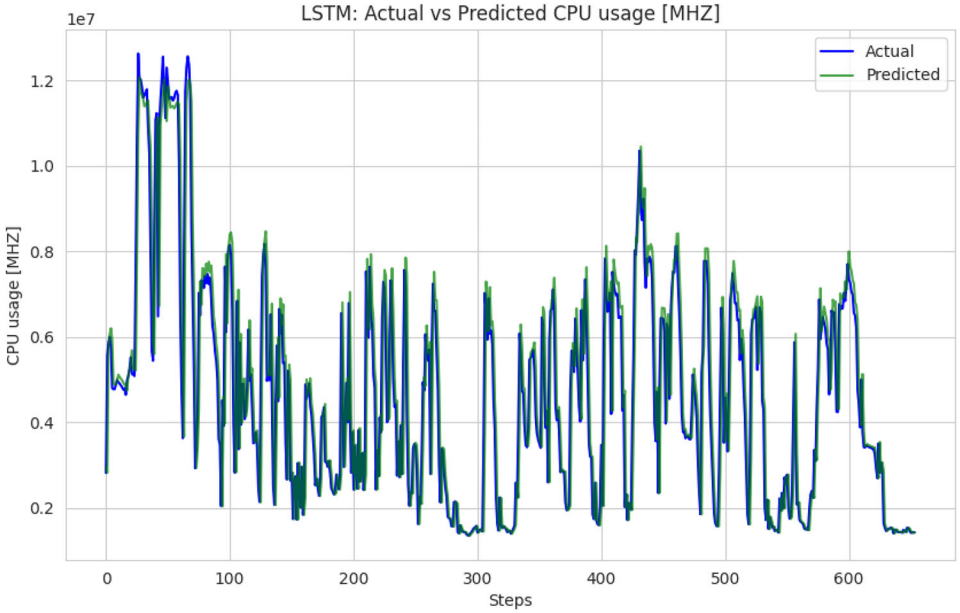

Fig. 9 LSTM model (actual vs. predicted)

Table 3 Performance comparison of proposed CloudAIBus approach (DeepAR) with baselines (ARIMA and LSTM)

Model	MAE	MSE	MAPE
Calheiros et al. [29] (ARIMA)	10.86	511.39	29.61
Tang et al. [31] (LSTM)	5.082	1415.5	69.5
Proposed CloudAIBus (DeepAR)	19.25	438.0	16.75

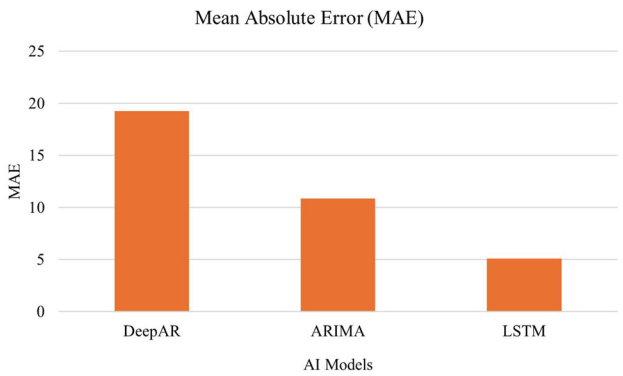

Fig. 10 Performance comparison of proposed CloudAIBus Approach (DeepAR) with baselines (ARIMA and LSTM) in terms of MAE

from the actual values, its MSE of 438.0 is significantly lower than both the ARIMA and LSTM models. This suggests that the DeepAR model is less prone to large prediction errors. Most notably, the DeepAR model achieves a MAPE of 16.75, which is substantially lower than the other models. This indicates that the DeepAR model's predictions are more consistent and reliable, with errors representing a smaller proportion of the actual

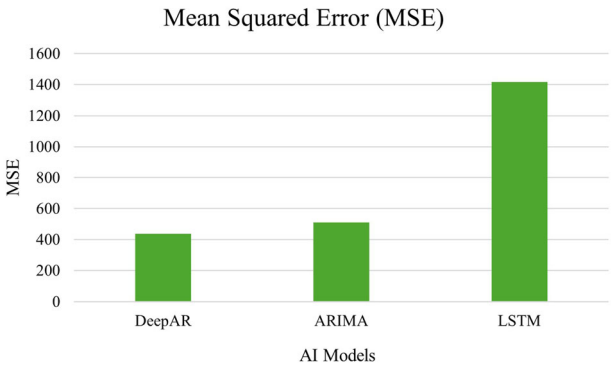

Fig. 11 Performance comparison of proposed CloudAIBus Approach (DeepAR) with baselines (ARIMA and LSTM) in terms of MSE

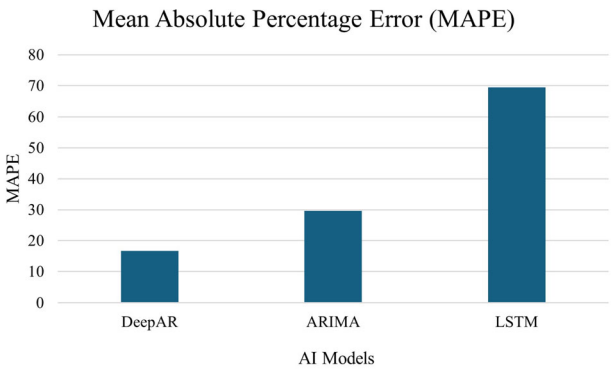

Fig. 12 Performance comparison of proposed CloudAIBus Approach (DeepAR) with baselines (ARIMA and LSTM) in terms of MAPE

values. These results highlight the effectiveness of the DeepAR model in handling the complex, non-stationary time series data typically encountered in cloud computing

environments. By leveraging both historical and covariate data, the DeepAR model is able to capture intricate temporal dependencies and provide more accurate and consistent predictions for resource allocation tasks.

5.7 Energy cost and execution time

Table 4 shows the training and prediction energy costs and execution times for different models: ARIMA, LSTM, and DeepAR. Table 4 highlights the differences in energy costs and execution times among ARIMA, LSTM, and DeepAR models. ARIMA has the lowest training and prediction energy costs and times, followed by LSTM. DeepAR, while having the highest costs and times, offers improved predictive performance that can outweigh these higher resource requirements, especially in dynamic cloud environments.

6 Prediction accuracy and efficiency for resource allocation using DeepAR model

In this section, we critically evaluate the performance of the DeepAR model in predicting CPU usage and allocation using the GWA-T-12 dataset. Our experimental setup is based on the Amazon SageMaker platform, which provides a fully managed service for deploying machine learning models. The model was given historical data with a context length of 30 min and it gives a forecast of the utilisation for the next 30 min. The evaluation focuses on three key aspects: prediction accuracy, efficiency in resource allocation and implications for scalability and environmental sustainability. By testing the DeepAR model on both stationary and non-stationary data, we aim to evaluate its performance under a variety of conditions and demonstrate its versatility and robustness in handling different types of time series data. The results of these tests, as shown in the Fig. 13, suggest that the DeepAR model can provide accurate and reliable predictions for both stationary and non-stationary data, making it a promising tool for predictive resource allocation in cloud computing.

6.1 Prediction accuracy analysis

The DeepAR model's prediction accuracy was critically evaluated by using MSE, MSE and MAPE across four different tests (Test#1, Test#2, Test#3, Test#4) as shown in Table 5. The analysis provides a comprehensive understanding of the model's performance in predicting CPU usage within the GWA-T-12 dataset. The model exhibited a relatively consistent performance, as evidenced by the average MAPE of 8.5%. This consistency reflects the model's robustness in handling variations within the dataset. The variation in MSE across tests further highlights the model's adaptability. Specifically, the MSE ranged from 37.80 in Test#2 to 522.83 in Test#4, with intermediate values in Test#1 and Test#3, as shown in Table 5. The corresponding MAE values ranged from 5.78 to 14.49, demonstrating the model's ability to capture complex patterns and nuances.

The relatively low and consistent MAPE values suggest a stable performance in terms of percentage errors. This stability supports the model's potential for accurately allocating resources and reducing over-provisioning in cloud computing services. The combined analysis of MSE, MAE, and MAPE offers a balanced perspective on the model's accuracy. The variations in these metrics provide insights into the magnitude and deviation of the errors, while the MAPE offers a normalized comparison across different scales.

The specific values of MSE, MAE, and MAPE across tests substantiate the model's performance, with an emphasis on its adaptability and efficiency. The results align with the research objective of optimizing resource allocation, demonstrating the model's potential in real-world applications.

Table 4 Training and prediction energy costs (in kWh) and times (in ms) for different models

Model	Training energy cost (kWh)	Training Time (ms)
ARIMA [29]	3.60	3
LSTM [31]	43.20	36
DeepAR (CloudAIBus)	252.00	210
Model	**Prediction energy cost (kWh)**	**Prediction time (ms)**
ARIMA [29]	0.0036	0.003
LSTM [31]	0.60	0.5
DeepAR (CloudAIBus)	1.068	0.89

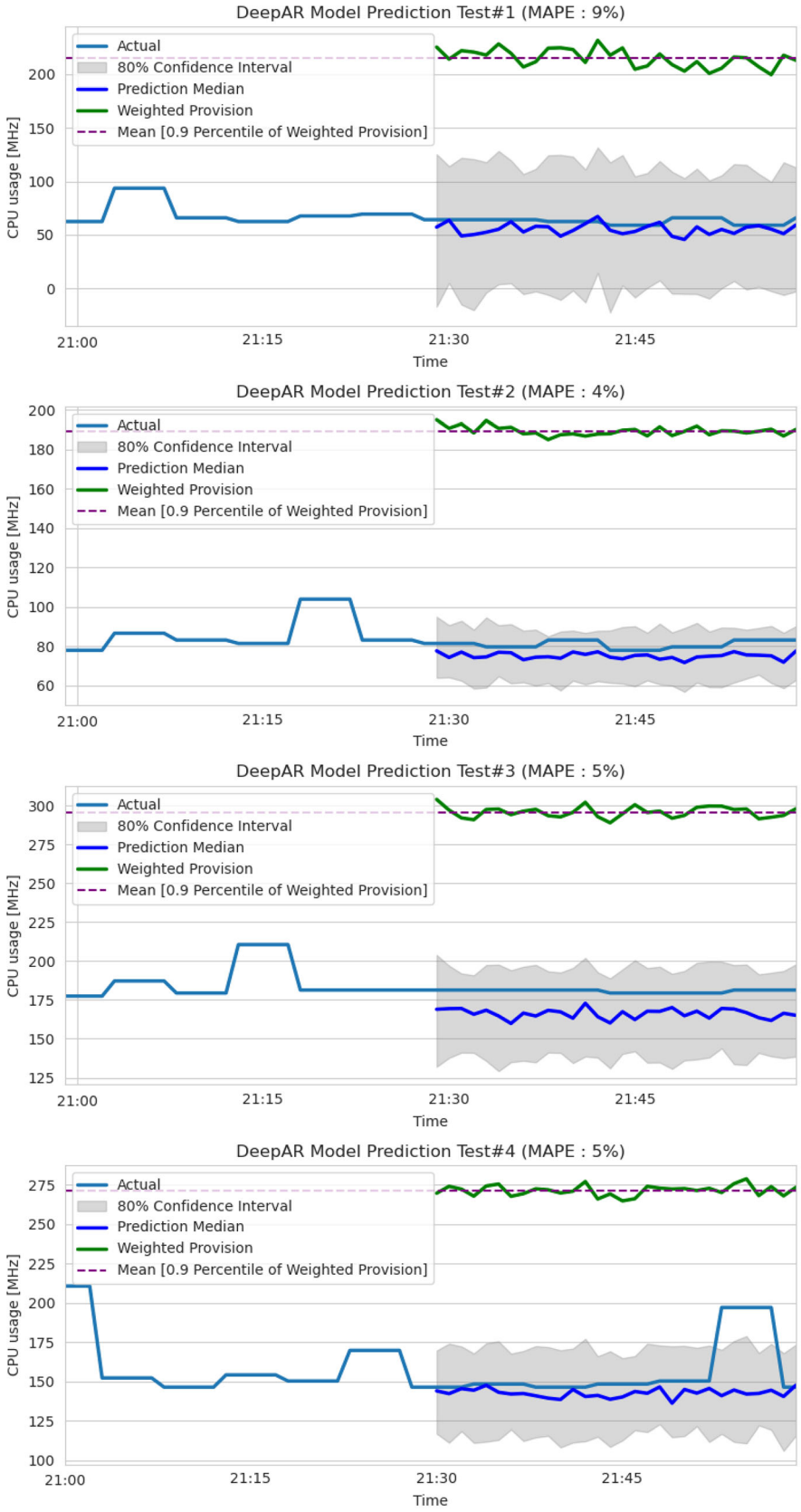

Fig. 13 DeepAR model prediction vs. resource provisioned

Table 5 Summary of prediction accuracy metrics for DeepAR model

Test	MSE	MAE	MAPE %
Test 1	87.82	7.80	12.30
Test 2	37.80	5.78	7.10
Test 3	219.62	14.49	8.01
Test 4	522.83	13.59	7.66
Average	216.25	9.75	8.50

6.2 Efficiency in resource allocation

The efficiency of the DeepAR model was assessed through a series of tests (Test#1, Test#2, Test#3, Test#4), comparing its performance with the actual CPU provisioning and usage in the GWA-T-12 dataset. The evaluation revealed significant insights:

1) Reduction in over-provisioning: The DeepAR model consistently demonstrated a reduction in over-provisioning across all tests. While the GWA-T-12 dataset exhibited a high percentage of unused CPU, ranging from 97.45% to 98.65%, the DeepAR model managed to reduce this percentage to as low as 32.35%. This reduction indicates a more balanced allocation, minimizing the gap between provisioned and used CPUs.

2) Optimization of resources: The DeepAR model's ability to closely match CPU usage with allocation minimizes waste. For instance, in Test#3, the DeepAR model allocated only 271.38 MHz total CPU compared to GWA-T-12's 11703.99 MHz, reducing the unused CPU from 11520.41 MHz to 87.79 MHz. This optimization aligns with the research objective of accurately allocating resources to reduce over-provisioning.

3) Consistency in performance: The DeepAR model's consistent reduction in unused CPU across diverse scenarios within the GWA-T-12 dataset suggests robustness and adaptability. This consistency is evident in the reduction of unused CPU percentage across all tests, showcasing the model's capability to minimize over-provisioning in various contexts.

6.3 Scalability and environmental impact

:

The DeepAR model's efficiency in resource allocation is evident in its capacity to dynamically allocate resources according to demand. In the conducted tests, the model reduced unused CPU from 2533.64 MHz in Test#1, 5116.42 MHz in Test#2, 11520.41 MHz in Test#3, and 11546.19 MHz in Test#4, demonstrating adaptability across diverse scenarios. This adaptability is a critical factor for scalability in cloud computing, as it ensures that resources can be effectively scaled up or down as needed. The consistent performance in minimizing over-provisioning across all tests substantiates the DeepAR model's potential for scalable cloud services.

The reduction in over-provisioning by the DeepAR model translates to lower energy requirements, contributing to environmental sustainability. For example, the decrease in unused CPU from 2533.64MHz to 205.03MHz in Test#1 represents a potential energy saving of approximately 91.9% for that specific scenario. By avoiding unnecessary CPU allocation, the model contributes to sustainable computing practices. The average reduction of unused CPU across all tests by 89.5% indicates a substantial decrease in energy consumption. Environmental sustainability in cloud computing necessitates energy-efficient practices, and the DeepAR model's significant reduction in over-provisioning, as evidenced by the test results in Table 5, aligns with global efforts towards reducing energy consumption and the associated environmental footprint.

In the Fig. 14 we compare the traditional and proposed approach in CPU resource provisioning. The 'CPU Unused' segment, representing the provisioned but unutilized CPU resources, is noticeably smaller in our approach compared to the traditional method. This difference underscores the over-provisioning that often occurs in traditional resource allocation strategies, where resources are provisioned based on peak demand or worst-case scenarios. Our proposed method, leveraging the predictive power of the DeepAR model, provides a more accurate estimate of resource needs, thereby reducing the extent of over-provisioning. This reduction in unused CPU resources translates into more efficient use of provisioned resources, which can lead to significant cost savings for cloud service providers, as they can avoid paying for unused resources.

Moreover, the minimized over-provisioning also contributes to energy efficiency. Given that unused CPUs still consume power, accurately predicting resource needs and reducing unused CPU time can significantly decrease the energy footprint of the cloud infrastructure. This is particularly crucial in the current era, where energy efficiency and sustainability are of paramount importance. Furthermore, by ensuring that resources are provisioned in line with predicted demand, we can avert potential service disruptions due to under-provisioning, thereby improving user experience and maintaining adherence to service level agreements (SLAs).

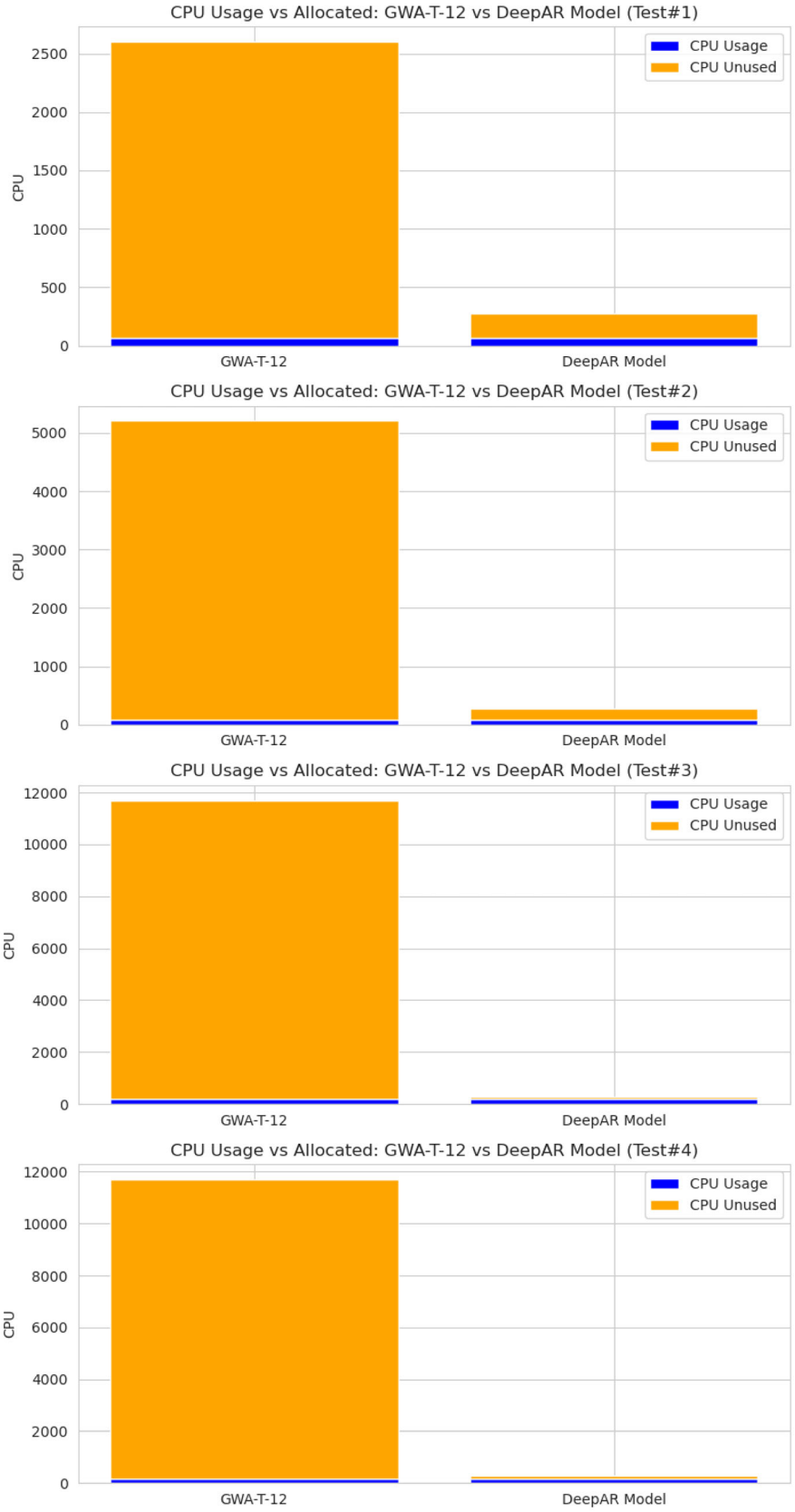

Fig. 14 CPU provisioning: GWA-T-12 vs. DeepAR model

7 Conclusions and future work

This paper proposes a new testbed for AI-driven cloud computing environments called CloudAIBus for effective resource allocation in uisng the DeepAR model. CloudAIBus utilises a deep learning model to forecast CPU usage for making cost-effective resource allocation decisions. The implementation of the DeepAR model for CPU provisioning has been rigorously evaluated, revealing significant insights into its accuracy, adaptability, and efficiency. The efficiency of the DeepAR model was assessed through a series of tests, comparing its performance with the actual CPU provisioning and usage in the GWA-T-12 dataset. The evaluation revealed a consistent reduction in over-provisioning across all tests. While the GWA-T-12 dataset exhibited a high percentage of unused CPU, ranging from 97.45 to 98.65%, the DeepAR model managed to reduce this percentage to as low as 32.35%. This reduction indicates a more balanced allocation, minimizing the gap between provisioned and used CPU, and aligns with the research objective of optimizing resource allocation, demonstrating the model's potential in real-world applications. The implementation of DeepAR enables the workload predictor to provide crucial insights into future resource requirements based on historical patterns and current demand. This predictive capability signifies a shift in resource allocation strategy, transitioning from a reactive approach, where resources are adjusted based on current demand, to a proactive approach that adjusts resources based on predicted future demand. This predictive resource allocation significantly enhances the system's ability to respond and adapt to fluctuating workloads, leading to more efficient resource utilization.

7.1 Possible extensions of CloudAIBus testbed

In the future, the proposed CloudAIBus Testbed can be extended in the following ways:

1) Automatic resource allocation: Future research should explore the integration of these models with existing cloud management systems, ensuring seamless and automated resource allocation and scaling [6]. This would involve addressing challenges related to real-time data processing, model updating, and system integration. The development of a specialized framework or integration with existing cloud services could further streamline this process. This direction provides a comprehensive and technically advanced approach to resource allocation in cloud computing, addressing not just the technical aspects but also the financial and environmental implications.

2) Generative AI: The exploration of latest Generative AI models, the development of more sophisticated prediction models, and the assessment of the overall system performance and cost-effectiveness represent additional promising directions for future research and development [1].

3) Sustainable cloud computing: Given the potential for the proposed method to reduce energy consumption by minimizing unused CPU time, future work could involve a more detailed exploration of this aspect, including quantifying the potential energy savings and investigating strategies to further enhance energy efficiency [6].

4) Serverless edge computing: Future research should focus on extending the application of the DeepAR model to serverless architectures within multi-tenant edge-cloud environments. This would involve developing a dynamic resource allocation algorithm that can handle the event-driven, stateless nature of serverless edge computing [71]. The algorithm should be capable of predicting resource needs based on the diverse and fluctuating demands of multiple tenants, taking into account factors such as function invocation frequency, execution time, and concurrency requirements.

5) Carbon neutral computing: In addition, the research should aim to optimize both cost and environmental impact. This would require the development of a cost-optimization model that considers various factors such as the pricing model of the cloud provider, the cost associated with different resource types, and the penalties for under-provisioning or over-provisioning resources [72]. The model should also factor in the energy consumption of the computing resources, with the aim of minimizing the carbon footprint of the cloud infrastructure. This direction would provide a comprehensive and technically advanced approach to resource allocation in cloud computing, addressing not just the technical aspects but also the financial and environmental implications.

6) Real-time data processing: Furthermore, the research should explore the integration of these models with the existing cloud management systems, ensuring seamless and automated resource allocation and scaling based on the predictions of the DeepAR model. This would involve addressing challenges related to real-time data processing, model updating, and system integration [73].

7) Statistical modelling: While the results from the Augmented Dickey-Fuller test support the stationarity of our data, we recognize that the evaluation lacks the necessary rigor for a comprehensive scientific comparison of stochastic algorithms. In future work, we will employ statistical hypothesis testing methods, such as paired t-tests or ANOVA, to ensure that the differences observed in the figures are statistically significant and not due to random chance. This will enhance the robustness of our comparisons, providing a more rigorous evaluation of our model's performance.

8) Heterogeneous cloud environment: Further research will also expand the analysis by testing the model under varying scenarios and conditions, such as different data sizes, hardware setups, and prediction horizons, to better understand the model's efficiency and scalability [74]. These improvements will help ensure the reliability of our conclusions and the applicability of our approach in real-world scenarios.

9) Security: CloudAIBus framework can be extended against security threats in future studies. To do this, behavioral and anomaly detection can be done using latest AI models [23]. In this way, possible threats to cloud environments can be detected in advance, and prevention efforts can be carried out.

10) Privacy: Secure and robust cryptography techniques can be used to ensure the privacy of all data used in the CloudAIBus framework. Additionally, privacy standards can be increased by using anonymization techniques in data transmission processes [39, 70].

11) Net zero emissions: Green information technologies can be used to increase energy efficiency and keep carbon emissions to a minimum. To do this, it is a bright idea to use AI models with low energy requirements and high performance in future research [66].

12) Edge AI: The CloudAIBus framework can be extended to edge systems used for optimization problems such as latency and bandwidth efficiency and deployed at the edge of the network [24]. In addition, higher performance predictions can be obtained for the CloudAIBus framework by developing Edge AI models to be used for complex tasks [71].

13) Industry 4.0: The CloudAIBus framework can be expanded and used for solutions such as predictive maintenance in Industry 4.0 applications integrated with cloud systems [61]. In this way, system failures can be predicted with the Deep learning (DL) models available in CloudAIBus, production processes can be optimized and efficiency can be maximized.

14) Quantum cloud computing: It is certain that quantum computing, which is currently in its infancy, will shape many new fields with its high speed and capabilities such as easily solving complex problems [75]. One of these areas is cloud computing [76]. CloudAIBus framework can be developed and used for new paradigms such as quantum cloud computing (resource management etc.)

15) IoT: Devices with low processing capacity, such as IoT, can be integrated with systems such as edge and cloud and used in applications that require high processing power [69]. To do this, IoT data is usually sent to cloud servers over the internet and processed in systems on the cloud. As the amount of processed data reaches huge levels, problems such as optimizing network traffic and energy efficiency arise [21]. This is where CloudAIBus can come into play and notify cloud systems of times of demand fluctuation (monitoring network traffic). In this way, cloud systems can automatically scale resources to respond to increases in demand.

16) 6 G and beyond: The future of CloudAIBus lies in its ability to adapt to the rapidly evolving landscape of connectivity driven by 6 G and beyond. With ultra-low latency, higher data rates, and increased connectivity, 6 G networks will demand more advanced resource allocation strategies. CloudAIBus will leverage 6 G's capabilities to efficiently manage cloud resources, utilizing predictive AI models to ensure real-time scalability and distribution [6]. The framework will integrate seamlessly with 6 G's advanced AI and security measures, enabling enhanced predictive accuracy and privacy. Additionally, with 6 G's emphasis on energy efficiency, CloudAIBus will optimize its resource allocation models to minimize environmental impact, thus aligning with the global effort toward carbon neutrality.

17) Large-scale ML: The future of CloudAIBus will involve integrating large-scale machine learning capabilities in alignment with the findings of Hazra et al. [73]. Leveraging collaborative offloading techniques, CloudAIBus will efficiently manage the offloading of computationally intensive ML tasks from edge devices to fog and cloud layers. This approach will enable the framework to handle complex models effectively while ensuring optimal transmission scheduling to minimize latency and energy consumption. By incorporating dynamic resource allocation and adaptive load balancing strategies, CloudAIBus will ensure that computational resources are utilized efficiently to support large-scale ML applications. In the context of industrial IoT, where real-time data analytics and predictive maintenance are essential, CloudAIBus will provide robust support for large-scale ML by harnessing the combined power of fog and cloud computing.

18) Federated learning: The future of CloudAIBus will incorporate federated learning to meet the privacy and efficiency challenges in modern computing [6]. With federated learning, CloudAIBus will enable decentralized machine learning across distributed edge devices, ensuring data privacy by keeping data local while sharing only model updates. This aligns perfectly with the framework's aim to leverage edge computing for scalable and secure resource management. By utilizing the computational power of distributed systems, federated learning within CloudAIBus will enhance efficiency in model training without the need for centralized data storage, crucial in privacy-sensitive and bandwidth-constrained environments. However, challenges like handling heterogeneous data, optimizing communication, and maintaining model accuracy will require advanced coordination, which the framework aims to address. Through this integration,

CloudAIBus will support applications in IoT, healthcare, and smart cities, enabling efficient and privacy-preserving model training in these domains.

19) Quantum AI: As we look towards the future, the CloudAIBus framework will need to adapt to the evolving landscape of quantum computing and AI integration, as outlined by Pinto et al. [74]. Privacy-aware IoT systems will face new challenges and opportunities with the rise of quantum computing. Quantum AI, with its superior computational capabilities, will enable the processing of vast amounts of data for improved decision-making while also posing a potential threat to data privacy [75]. To address this, CloudAIBus will integrate quantum-resistant cryptographic techniques to safeguard personal data against quantum-based attacks. By incorporating AI and quantum computing for privacy-preserving methods, CloudAIBus aims to develop innovative strategies that not only leverage the power of quantum AI but also ensure robust data privacy in IoT environments. This will pave the way for privacy-focused applications that are resilient to future computational advancements.

20) Explainable AI (XAI): In its future evolution, CloudAIBus will integrate XAI into its distributed AI frameworks, particularly in edge networks, as emphasized by Hazra et al. [72]. With the growing adoption of zero-touch provisioning, ensuring transparency in distributed AI is crucial for clarity and confidence. CloudAIBus will embrace models that allow users and stakeholders to clearly understand the reasoning behind AI-driven decisions. Utilizing XAI, CloudAIBus aims to build user trust and acceptance by offering traceable, dependable, and understandable decision-making in distributed AI systems. This will empower the framework to self-manage autonomously while maintaining trustworthiness in edge network environments.

Acknowledgements Huaming Wu is supported by the National Natural Science Foundation of China (Grant No. 62071327) and Tianjin Science and Technology Planning Project (Grant No. 22ZYYYJC00020).

Author Contributions Sasidharan Velu (Conceptualization: Lead; Data curation: Lead; Formal analysis: Lead; Funding acquisition: Lead; Investigation: Lead; Methodology: Lead; Software: Lead; Validation: Lead; Writing - original draft: Lead) Sukhpal Singh Gill (Conceptualization: Lead; Data curation: Lead; Formal analysis: Lead; Funding acquisition: Lead; Investigation: Lead; Methodology: Lead; Software: Lead; Validation: Lead; Writing - original draft: Lead; Supervision: Supporting) Subramaniam Subramanian Murugesan (Conceptualization: Lead; Data curation: Lead; Formal analysis: Lead; Investigation: Lead; Methodology: Lead; Software: Lead; Validation: Lead; Writing - original draft: Lead) Huaming Wu (Conceptualization: Lead; Formal analysis: Lead; Funding acquisition: Lead; Investigation: Lead; Methodology: Lead; Writing - original draft: Lead; Supervision: Supporting) Xingwang Li (Conceptualization: Lead; Formal analysis: Lead; Writing - original draft: Lead; Writing - review & editing: Supporting)

Availability of data and materials We released CloudAIBus as an open source software and the implementation code with experiment scripts and results can be found at the GitHub repository: https://github.com/Subramaniam-dot/CloudAIBus

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no Conflict of interest.

Ethics approval Not available.

Consent to participate Not available.

Consent for publication Not available.

Code availability The code is available at https://github.com/Subramaniam-dot/CloudAIBus

References

1. Gill, S.S., Xu, M., Ottaviani, C., et al.: Ai for next generation computing: emerging trends and future directions. Internet Things **19**, 100514 (2022)
2. Rao, P.D.A.S.: "Orchestrating efficiency: Ai-driven cloud resource optimization for enhanced performance and cost reduction," *International Journal of Research Publication and Reviews*, (2023)
3. Wang, C.-N., Nguyen, M.-N., Nguyen, T.-D., Hsu, H., Nguyen, T.: Effective decision making: Data envelopment analysis for efficiency evaluation in the cloud computing marketplaces. Axioms **10**, 309 (2021)
4. Tuli, S., Gill, S.S., Xu, M., Garraghan, P., Bahsoon, R., Dustdar, S., Sakellariou, R., Rana, O., Buyya, R., Casale, G., et al.: Hunter: Ai based holistic resource management for sustainable cloud computing. J. Syst. Softw. **184**, 111124 (2022)
5. Iftikhar, S., Ahmad, M.M.M., Tuli, S., et al.: Hunterplus: Ai based energy-efficient task scheduling for cloud-fog computing environments. Internet Things **21**, 100667 (2023)
6. Gill, S.S., Wu, H., Patros, P., et al.: Modern computing: vision and challenges. Telemat. Inform. Rep. **13**, 100116 (2024)
7. Niyato, D., Chaisiri, S., Lee, B.-S.: "Economic analysis of resource market in cloud computing environment," 2009 IEEE Asia-Pacific Services Computing Conference (APSCC), pp. 156–162, (2009)
8. Alizadeh Javaheri, S.D., Ghaemi, R., Monshizadeh Naeen, H.: An autonomous architecture based on reinforcement deep neural network for resource allocation in cloud computing. Computing **106**(2), 371–403 (2024)
9. Jeyaraj, R., Balasubramaniam, A., Guizani, N., Paul, A.: Resource management in cloud and cloud-influenced technologies for internet of things applications. ACM Comput. Surv. **55**, 1–37 (2022)
10. Gurusamy, S., Selvaraj, R.: "Resource allocation with efficient task scheduling in cloud computing using hierarchical auto-associative polynomial convolutional neural network," Exp Syst. Appl., p. 123554, (2024)
11. Etemadi, M., Ghobaei-Arani, M., Shahidinejad, A.: A cost-efficient auto-scaling mechanism for iot applications in fog computing environment: a deep learning-based approach. Clust. Comput. **24**(4), 3277–3292 (2021)
12. Heidari, A., Navimipour, N.J., Jamali, M.A.J., Akbarpour, S.: A hybrid approach for latency and battery lifetime optimization in

iot devices through offloading and cnn learning. Sustain. Comput.: Inform. Syst. **39**, 100899 (2023)
13. Amiri, Z., Heidari, A., Navimipour, N.J., Unal, M., Mousavi, A.: Adventures in data analysis: a systematic review of deep learning techniques for pattern recognition in cyber-physical-social systems. Multim. Tools Appl. **83**(8), 909–973 (2024)
14. Gandhi, A., Dube, P., Karve, A., Kochut, A., Zhang, L.: Model-driven optimal resource scaling in cloud. Softw. Syst. Model. **17**, 509–526 (2018)
15. Heidari, A., Navimipour, N.J., Jamali, M.A.J., Akbarpour, S.: A green, secure, and deep intelligent method for dynamic iot-edge-cloud offloading scenarios. Sustain. Comput.: Inform. Syst. **38**, 100859 (2023)
16. Sallam, A., Li, K.: "Virtual machine proactive scaling in cloud systems," *2012 IEEE International Conference on Cluster Computing Workshops*, pp. 97–105, (2012)
17. Thurgood, B., Lennon, R.G.: "Cloud computing with kubernetes cluster elastic scaling," *Proceedings of the 3rd International Conference on Future Networks and Distributed Systems*, (2019)
18. Lu, Y., Liu, L., Panneerselvam, J., Zhai, X., Sun, X., Antonopoulos, N.: Latency-based analytic approach to forecast cloud workload trend for sustainable datacenters. IEEE Trans. Sustain. Comput. **5**, 308–318 (2020)
19. Jayalakshmi, S.: Predictive scaling for elastic compute resources on public cloud utilizing deep learning based long short-term memory. Int. J. Adv. Comput. Sci. Appl. **12**, 73–81 (2021)
20. Torabi, E., Ghobaei-Arani, M., Shahidinejad, A.: Data replica placement approaches in fog computing: a review. Clust. Comput. **25**(5), 3561–3589 (2022)
21. Golec, M., Gill, S.S., Parlikad, A.K., Uhlig, S.: Healthfaas: Ai based smart healthcare system for heart patients using serverless computing. IEEE Internet Things J. (2023). https://doi.org/10.1109/JIOT.2023.3277500
22. Heidari, A., Jamali, M.A.J., Navimipour, N.J., Akbarpour, S.: A qos-aware technique for computation offloading in iot-edge platforms using a convolutional neural network and markov decision process. IT Professional **25**(1), 24–39 (2023)
23. Golec, M., Ozturac, R., Pooranian, Z., Gill, S.S., Buyya, R.: Ifaasbus: a security-and privacy-based lightweight framework for serverless computing using iot and machine learning. IEEE Trans. Industr. Inf. **18**(5), 3522–3529 (2021)
24. Nandhakumar, A.R., et al.: Edgeaisim: a toolkit for simulation and modelling of ai models in edge computing environments. Measure.: Sens. **31**, 100939 (2024)
25. Heidari, A., Navimipour, N.J.: A new sla-aware method for discovering the cloud services using an improved nature-inspired optimization algorithm. PeerJ Comput. Sci. **7**, e539 (2021)
26. Zhang, H., Jiang, G., Yoshihira, K., Chen, H., Saxena, A.: "Intelligent workload factoring for a hybrid cloud computing model,'' in. Congr. Serv. - I **2009**, 701–708 (2009)
27. Fang, W., Lu, Z., Wu, J., Cao, Z.: "Rpps: a novel resource prediction and provisioning scheme in cloud data center,'' in. IEEE Ninth Int. Conf. Serv. Comput. **2012**, 609–616 (2012)
28. Ciptaningtyas, H. T., Santoso, B. J., Razi, M.F.: "Resource elasticity controller for docker-based web applications," in *2017 11th International Conference on Information & Communication Technology and System (ICTS)*, pp. 193–196 (2017)
29. Calheiros, R.N., Masoumi, E., Ranjan, R., Buyya, R.: Workload prediction using arima model and its impact on cloud applications' qos. IEEE Trans. Cloud Comput. **3**(4), 449–458 (2015)
30. Kirchoff, D.F., Xavier, M., Mastella, J., F De Rose, C.A.: "A preliminary study of machine learning workload prediction techniques for cloud applications," in *2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pp. 222–227 (2019)
31. Tang, X., Liu, Q., Dong, Y., Han, J., Zhang, Z.: "Fisher: An efficient container load prediction model with deep neural network in clouds," in *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, pp. 199–206 (2018)
32. Yan, M., Liang, X., Lu, Z., Wu, J., Zhang, W.: Hansel: Adaptive horizontal scaling of microservices using bi-lstm. Appl. Soft Comput. **105**, 107216 (2021)
33. Anupama, K., Shivakumar, B., Nagaraja, R.: Resource utilization prediction in cloud computing using hybrid model. Int. J. Adv. Comput. Sci. Appl. **12**, 2021 (2021)
34. Ashawa, M., Douglas, O., Osamor, J., Jackie, R.: Improving cloud efficiency through optimized resource allocation technique for load balancing using lstm machine learning algorithm. J Cloud Comput. **11**(1), 1–19 (2022)
35. Prachitmutita, I., Aittinonmongkol, W., Pojjanasuksakul, N., Supattatham, M., Padungweang, P.: "Auto-scaling microservices on iaas under sla with cost-effective framework," in *Proceedings of the 2018 Tenth International Conference on Advanced Computational Intelligence (ICACI)*, Xiamen, China, 29–31 pp. 583–588 (March 2018)
36. Toka, L., Dobreff, G., Fodor, B., Sonkoly, B.: "Adaptive ai-based auto-scaling for kubernetes,'' in,: 20th IEEE/ACM International Symposium on Cluster. Cloud Internet Comput. (CCGRID) **2020**, 599–608 (2020)
37. Imdoukh, M., Ahmad, I., Alfailakawi, M.: Machine learning-based auto-scaling for containerized applications. Neural Comput. Appl. **32**, 9745–9760 (2019)
38. Donta, P.K., Murturi, I., Casamayor Pujol, V., Sedlak, B., Dustdar, S.: Exploring the potential of distributed computing continuum systems. Computers **12**(10), 198 (2023)
39. Golec, M., Gill, S.S., Golec, M., Xu, M., Ghosh, S.K., Kanhere, S.S., Rana, O., Uhlig, S.: Blockfaas: Blockchain-enabled serverless computing framework for ai-driven iot healthcare applications. J. Grid Comput. **21**(4), 63 (2023)
40. Liang, Q., Hanafy, W.A., Ali-Eldin, A., Shenoy, P.: Model-driven cluster resource management for ai workloads in edge clouds. ACM Transactions on Autonomous and Adaptive Systems **18**(1), 1–26 (2023)
41. Yaqoob, A., Bi, T., Muntean, G.-M.: A survey on adaptive 360 video streaming: Solutions, challenges and opportunities. IEEE Commun. Surv. Tutor. **22**(4), 2801–2838 (2020)
42. Islam, S., Keung, J., Lee, K., Liu, A.: Empirical prediction models for adaptive resource provisioning in the cloud. Future Gener. Comput. Syst. **28**, 155–162 (2012)
43. Mehmood, T., Latif, S., Malik, S.: "Prediction of cloud computing resource utilization," *2018 15th International Conference on Smart Cities: Improving Quality of Life Using ICT & IoT (HONET-ICT)*, pp. 38–42, (2018)
44. Gadhavi, L.J., Bhavsar, M.D.: Adaptive cloud resource management through workload prediction. Energy Syst. **13**, 601–623 (2019)
45. Ralha, C., Mendes, A.H.D., Laranjeira, L.A.F., Araujo, A.P.F., Melo, A.: Multiagent system for dynamic resource provisioning in cloud computing platforms. Future Gener. Comput. Syst. **94**, 80–96 (2019)
46. Wang, L., Xu, J., Duran-Limon, H., Zhao, M.: "Qos-driven cloud resource management through fuzzy model predictive control," *2015 IEEE International Conference on Autonomic Computing*, pp. 81–90, (2015)
47. Salinas, D., Flunkert, V., Gasthaus, J.: "Deepar: Probabilistic forecasting with autoregressive recurrent networks," arXiv preprint arXiv:1704.04110, (2017)

48. Iosup, A., Li, H., Jan, M., Anoep, S., Dumitrescu, C., Wolters, L., Epema, D.H.: The grid workloads archive. Futur. Gener. Comput. Syst. **24**(7), 672–686 (2008)
49. Bassi, S., Gomekar, A., Murthy, A.: A learning algorithm for time series based on statistical features. Int. J. Adv. Eng. Sci. Appl. Math. **11**, 230–235 (2019)
50. Yeh, C.-C. M., Dai, X., Chen, H., Zheng, Y., Fan, Y., Der, A., Lai, V., Zhuang, Z., Wang, J., Wang, L., Zhang, W.: "Toward a foundation model for time series data," *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023
51. Ahmed, N., Atiya, A., Gayar, N.E., El-Shishiny, H.: An empirical comparison of machine learning models for time series forecasting. Economet. Rev. **29**, 594–621 (2010)
52. Venkatraman, A., Hebert, M., Bagnell, J.: Improving multi-step prediction of learned time series models. Proceed AAAI Conf Artif Intell
53. Tseng, F.-M., Yu, H.-C., Tzeng, G.: Applied hybrid grey model to forecast seasonal time series. Technol. Forecast. Soc. Chang. **67**, 291–302 (2001)
54. Isiaka, A., Isiaka, A., Isiaka, A.: Forecasting with arma models. Int. J. Res. Bus. Soc. Sci. **10**, 205–234 (2021)
55. Tseng, F.-M., Yu, H.-C., Tzeng, G.: Combining neural network model with seasonal time series arima model. Technol. Forecast. Soc. Chang. **69**, 71–87 (2002)
56. Alsharif, M., Younes, M.K., Kim, J.: Time series arima model for prediction of daily and monthly average global solar radiation: The case study of seoul, south korea. Symmetry **11**(2), 240 (2019)
57. Du, B., Wu, C., Huang, Z.: Learning resource allocation and pricing for cloud profit maximization. Proceed. AAAI Conf. Artif. Intell. **33**(01), 7570–7577 (2019)
58. Carneiro, T., et al.: Performance analysis of google colaboratory as a tool for accelerating deep learning applications. IEEE Access **6**(61), 677 (2018)
59. Gujjar, J.P., Kumar, V.: Google colaboratory: tool for deep learning and machine learning applications. Int. J. Comput. Simul. **6**, 23–26 (2021)
60. Flunkert, V., Salinas, D., Gasthaus, J.: "Deepar: Probabilistic forecasting with autoregressive recurrent networks," ArXiv, vol. abs/1704.04110, (2017)
61. Golec, M., Gill, S. S., Wu, H., Can, T. C., Golec, M., Cetinkaya, O., Cuadrado, F., Parlikad, A. K., Uhlig, S.: "Master: Machine learning-based cold start latency prediction framework in serverless edge computing environments for industry 4.0," *IEEE Journal of Selected Areas in Sensors*, pp. 1–13, (2024)
62. Liao, Y., Liang, C.: "A temperature time series forecasting model based on deepar," *2021 7th International Conference on Computer and Communications (ICCC)*, pp. 1588–1593, (2021)
63. Jeon, Y., Seong, S.: Robust recurrent network model for intermittent time-series forecasting. Int. J. Forecast. **38**(4), 1415–25 (2021)
64. Jungbluth, A., Lederer, J.: "The deepcar method: Forecasting time-series data that have change points," *ArXiv*, vol. abs/2302.11241, (2023)
65. Shi, S., Qiu, X., Ru, Y., Tan, X.: "A deepar-based neural network for time series forecasting," *2023 5th International Conference on Data-driven Optimization of Complex Systems (DOCS)*, pp. 1–7, (2023)
66. Golec, M., Gill, S. S., Cuadrado, F., Parlikad, A. K., Xu, M., Wu, H., Uhlig, S.: "Atom: Ai-powered sustainable resource management for serverless edge computing environments," *IEEE Transactions on Sustainable Computing*, (2023)
67. Lim, H., Chung, K., Lee, S.: Probabilistic forecasting for demand of a bike-sharing service using a deep-learning approach. Sustainability **14**(23), 15889 (2022)
68. Han, J., Xue, L.: "Multiple production time series forecasting using deepar and probabilistic forecasting," *Day 3 Wed, October 18, 2023*, (2023)
69. Murugesan, S.S., et al.: Neural networks based smart e-health application for the prediction of tuberculosis using serverless computing. IEEE J. Biomed. Health Inform. (2024). https://doi.org/10.1109/JBHI.2024.3367736
70. Golec, M., Golec, M., Xu, M., Wu, H., Gill, S. S., Uhlig, S.: "Priceless: Privacy enhanced ai-driven scalable framework for iot applications in serverless edge computing environments," *Internet Technology Letters*, p. e510, (2024)
71. Singh, R., Gill, S.S.: Edge ai: a survey. Internet Things Cyber-Phys. Syst. **3**, 71–92 (2023)
72. Hazra, A., Morichetta, A., Murturi, I., Lovén, L., Dehury, C.K., Pujol, V.C., Donta, P.K., Dustdar, S.: Distributed ai in zero-touch provisioning for edge networks: challenges and research directions. Computer **57**(3), 69–78 (2024)
73. Hazra, A., Donta, P.K., Amgoth, T., Dustdar, S.: Cooperative transmission scheduling and computation offloading with collaboration of fog and cloud for industrial iot applications. IEEE Internet Things J. **10**(5), 3944–3953 (2022)
74. Pinto, G.P., Donta, P.K., Dustdar, S., Prazeres, C.: A systematic review on privacy-aware iot personal data stores. Sensors **24**(7), 2197 (2024)
75. Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., Lloyd, S.: Quantum machine learning. Nature **549**(7671), 195–202 (2017)
76. Golec, M., Hatay, E.S., Golec, M., Uyar, M., Golec, M., Gill, S.S.: Quantum cloud computing: Trends and challenges. J. Econ. Technol. (2024). https://doi.org/10.1016/j.ject.2024.05.001

Sasidharan Velu completed an M.Sc. in Big Data Science from Queen Mary, University of London, UK. He holds a Bachelor's degree in Electronics and Communication Engineering (ECE). Sasidharan has professional software development experience, and his research interests include Cloud Computing and the Internet of Things (IoT).

Sukhpal Singh Gill is Assistant Professor of Cloud Computing with the School of Electronic Engineering and Computer Science, Queen Mary University of London, U.K since 2019. He is serving as an Editor-in-Chief for IGI Global IJAEC, Associate Editor in IEEE IoT, Wiley SPE, Elsevier IoT, Wiley ETT and IET Networks Journal and Area Editor for Springer Cluster Computing Journal. He has published in prominent international journals and conferences such as IEEE/ACM Transactions, IEEE IoT Journal, IEEE/ACM UCC and IEEE CCGRID. He was the winner of the Queen Mary University Education Excellence Award. His research interests include Cloud Computing, Edge Computing, IoT and Energy Efficiency. For further information, please visit: http://www.ssgill.me.

Subramaniam Subramanian Murugesan completed an M.Sc. in Big Data Science from Queen Mary, University of London, UK. He holds a Bachelor's degree in Bioinformatics. Subramaniam has professional software development experience. Recently, he has published his MSc disseration in IEEE Journal of Biomedical and Health Informatics (JBHI). His research interests include IoT, Machine Learning, Smart Healthcare, and Cloud Computing.

Huaming Wu (Senior Member, IEEE) received the B.E. and M.S. degrees from Harbin Institute of Technology, China in 2009 and 2011, respectively, both in electrical engineering. He received the Ph.D. degree with the highest honor in computer science at Freie Universit"at Berlin, Germany in 2015. He is currently a professor at the Center for Applied Mathematics, Tianjin University, China. His research interests include mobile cloud computing, edge computing, Internet of Things, deep learning, complex networks, and DNA storage.

Xingwang Li (Senior Member, IEEE) received the M.Sc. degree from the University of Electronic Science and Technology of China in 2010 and the Ph.D. degree from the Beijing University of Posts and Telecommunications in 2015. From 2010 to 2012, he was an Engineer with Comba Telecom Ltd., Guangzhou, China. He is currently an Associate Professor with the School of Physics and Electronic Information Engineering, Henan Polytechnic University, Jiaozuo, China. His research interests include span wireless communication, intelligent transport system, artificial intelligence, and Internet of Things. He is an Editorial Board of the IEEE Transactions on Intelligent Transportation Systems, IEEE Transactions on Vehicular Technology, IEEE Systems Journal, and IEEE Sensors Journal.