National College of Ireland

# Cloud-Based Time Series Prediction of CPU Load Through Advanced Deep Learning Models

MSc Research Project
MSc Cloud Computing

Rohan Ajila
Student ID: 22249729

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

**National College of Ireland**

**MSc Project Submission Sheet**

**School of Computing**

| | |
|---|---|
| **Student Name:** | Rohan Ajila |
| **Student ID:** | 22249729 |
| **Programme:** | MSc Cloud Computing **Year:** 2024-25 |
| **Module:** | MSc Research Project |
| **Supervisor:** | Vikas Sahni |
| **Submission Due Date:** | 24 April 2025 |
| **Project Title:** | Cloud-Based Time Series Prediction of CPU Load Through Advanced Deep Learning Models |

**Word Count:7482** **Page Count: 22**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

**Signature:** Rohan Ajila

**Date:** 24 April 2025

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

| | |
|---|---|
| Attach a completed copy of this sheet to each project (including multiple copies) | □ |
| **Attach a Moodle submission receipt of the online project submission,** to each project (including multiple copies). | □ |
| **You must ensure that you retain a HARD COPY of the project**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | □ |

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

| **Office Use Only** | |
|---|---|
| Signature: | |
| Date: | |
| Penalty Applied (if applicable): | |

# Cloud-Based Time Series Prediction of CPU Load Through Advanced Deep Learning Models

Rohan Ajila
22249729

## Abstract

Cloud computing enables dynamic resource allocation and scalability, yet predicting resource usage remains a critical challenge for optimal performance and cost performance. Traditional auto scaling mechanisms based on reactive thresholds often fail to anticipate sudden workload shifts, resulting in under- or over-provisioning of resources. This study addresses that limitation by developing a cloud-native forecasting system for predicting CPU utilization in Microsoft Azure virtual machines using time-series data. Using AWS services including EC2, Cloud9, and S3, a complete pipeline was implemented—from data collection and preprocessing to training and evaluating various machine learning and deep learning models.

Traditional models such as Decision Tree and Gradient Boosting provided initial baselines but were limited in capturing complex temporal patterns. Deep learning models, particularly LSTM, BiLSTM, and the proposed Multihead BiLSTM, significantly outperformed them in predictive accuracy. The Multihead BiLSTM achieved the highest performance with an RMSE of 0.0213 and $R^2$ of 0.974, highlighting its ability to model non-linear, sequential dependencies. Unlike prior research focused on theoretical load balancing frameworks, this project demonstrates a practical, deployable solution in a real cloud environment. The results give good results for proactive auto scaling and dynamic resource management, contributing toward the development of intelligent, self-regulating cloud infrastructure systems.

**Keywords: Cloud Computing, Time Series Forecasting, CPU Utilization, Machine Learning, Deep Learning**

# 1 Introduction

## 1.1 Background

Cloud computing is revolutionising how organizations deploy, manage and scale applications and services Falade et al. (2024). Platforms like Microsoft Azure and Amazon Web Services (AWS) provide elastic virtual resources that can dynamically adjust based on workload demand. Among these resources CPU utilization is a very important metric that directly affects system performance, user experience and operational cost Li et al. (2020). Forecasting CPU

usage allows cloud providers and consumers to optimize resource allocation, avoid any kind of performance and reduce unnecessary expenses Prasad et al. (2023).

## 1.2 Problem Statement

Despite the availability of huge data from cloud providers there are so many systems that trusts on reactive mechanisms for resource management which mainly leads to under- or over-provisioning. Current autoscaling methods based on static thresholds which lack the intelligence to predict future load in a good way by Park and Jeong (2023). There is a growing need for cloud-native solutions that use historical CPU utilization data to forecast future demand by securing powerful cloud resource management.

## 1.3 Aim of the Study

This study aims to design, implement, and evaluate a cloud-based forecasting system capable of accurately predicting CPU utilization in virtual machines using historical time-series data from Microsoft Azure. Leveraging the scalability and flexibility of cloud platforms, specifically AWS services such as EC2, Cloud9, and S3, the study focuses on building a robust and automated pipeline for data collection, preprocessing, model training, and output visualization. By integrating machine learning and deep learning techniques namely Decision Tree, Gradient Boosting, LSTM, BiLSTM, and Multihead BiLSTM the research seeks to identify the most effective predictive models for handling complex, temporal patterns in cloud resource usage. This forecasting capability is intended to empower cloud administrators with proactive insights, enabling dynamic resource provisioning, improving workload balancing, and ultimately reducing infrastructure costs. Furthermore, the study aims to demonstrate the practical benefits of deploying intelligent analytics within a cloud-native architecture, showcasing how predictive modeling can support real-time decision-making and contribute to optimized performance management. By combining advanced modeling approaches with cloud computing principles, this project serves as a step toward smarter, self-regulating cloud ecosystems that can scale based on anticipated demand rather than reactive usage metrics alone.

## 1.4 Research Question

What are the most effective cloud-based approaches for forecasting CPU utilization in virtual machines, how can machine learning and deep learning models be implemented in a scalable cloud environment, and why is accurate prediction critical for optimizing resource allocation and reducing operational costs in cloud infrastructure?

## 1.5 Research Objectives

The research questions for this report are:

1. To develop a cloud-native data pipeline that enables scalable storage of time-series CPU utilization data from Azure virtual machines using AWS services such as EC2, Cloud9, and S3.
2. To implement and compare the performance of machine learning and deep learning models, including Decision Tree, Gradient Boosting, LSTM, BiLSTM, and Multihead BiLSTM, for forecasting CPU utilization in a cloud environment.
3. To evaluate the effectiveness of cloud-based CPU utilization forecasting in improving resource management decisions and reducing operational costs through predictive scaling and optimized infrastructure usage.

## 1.6 Report Structure Overview

The aim of this research is to develop a system which is cloud-native and helps to forecast the CPU utilization in virtual machines with the help of time-series data and advanced deep learning models. The models like LSTM, BiLSTM and a novel Multihead BiLSTM architecture have been deployed using a hybrid cloud environment to help improve the accuracy of the CPU resource usage predictions in the dynamic cloud setup.
The main contribution of this research is the practical implementation of the Multihead BiLSTM-based CPU load Prediction system that performs better than the other traditional models in forecasting the CPU load data which improves and supports proactive cloud resource management and helps in intelligent load balancing and auto-scaling.
This paper discusses relevant literature on time series forecasting and deep learning applications in cloud environments in Section 2. Section 3 gives the outline of the Research methodology used in this research. Section 4 describes the implementation of the forecasting system in the AWS cloud infrastructure. Section 5 discusses the experimental results across the different models used for comparison. Finally Section 6 presents the conclusion and also discusses identified limitations and future enhancements.

# 2 Literature Review

## 2.1 Time Series Forecasting in Cloud Environments

### 2.1.1 Fundamentals of Time Series Forecasting

Time series forecasting is a statistical and machine learning technique used to predict future values based on previously observed data points collected at consistent intervals over time Lim et al. (2021). It is grounded in the principle that historical patterns and structures such as trends, seasonality, and cycles can be identified, quantified, and projected into the future. Core traditional methods include Autoregressive Integrated Moving Average (ARIMA), Exponential Smoothing, and Holt-Winters, which are well-suited for linear data patterns Minsan and Minsan (2023). However, with the advent of complex and non-linear datasets, especially in large-scale systems such as cloud computing, machine learning and deep learning techniques like Support Vector Regression (SVR), Random Forest, Long Short-Term Memory (LSTM) and Transformer models have gained popularity due to their good ability to model non-linear relationships. There is a typical time series forecasting process which includes data preprocessing (handling missing values, scaling and transformation), stationarity testing, feature engineering (e.g., time lags and rolling windows), model training and performance evaluation using metrics like RMSE or $R^2$ score. The strength of time series forecasting which

lies in its adaptability to so many domains like finance, weather, energy and IT infrastructure and cloud computing Kashpruk et al. (2023).

### 2.1.2 Relevance to Cloud Computing

The approach of time series forecasting remains essential for cloud systems because it enables companies to optimize resource deployment and maintain service performance while reducing operational expenses Zheng et al. (2024). The fluctuation of workloads within Microsoft Azure and Amazon Web Services (AWS) together with Google Cloud Platform (GCP) occurs according to user demand and application processes and background system operations. The accurate prediction of CPU utilization plays an essential role for proactive resource management through scalable operations along with load distribution and agreement performance protection Gupta (2024). The accurate estimation of future resource consumption helps cloud service providers and consumers achieve efficient resource management by preventing both resource surpluses and deficits respectively Prasad et al. (2023).

## 2.2 Machine Learning Models for Resource Forecasting

The first study Liu et al. (2021) examines the forecast of groundwater level variations (CGWLs) across 46 wells in northeastern United States spans 1 to 3-month lead periods. An improvement in groundwater resource management during drought events requires the development of Support Vector Machines (SVMs) using a data assimilation (DA) technique for predicting CGWLs at 1-3-month lead times for 46 groundwater wells in the northeastern United States. The methodology incorporates both ground-level climate variables such as precipitation and solar radiation and air temperature and infrared surface temperature with groundwater anomalies data derived from Gravity Recovery and Climate Experiment data (GRACE) to predict CGWLs. The research faced difficulties because groundwater levels exhibit diverse correlations with climatic elements at different site locations. The results establish SVM-DA models achieve accurate CGWL predictions for three months durations across different stations when using climate variables as inputs. The model became more effective through the addition of GRACE data alongside the existing Climate Variables as their correlation with CGWLs and GWA values was strong. The SVM-DA model supplied better prediction results than basic SVM approaches in numerous monitoring sites but future analysis could be limited by groundwater system spatial features in addition to needing continuous data updates to keep model precision elevated especially in drought events.

The research presented in Ahmad et al. (2020) establishes a system for optimizing classifier hyperparameters through the combination of deep learning and machine learning techniques for improved electricity load forecasting. The main issue concerns both the wide fluctuations of power consumption and the problematic selection of optimal classifier parameters. A three-step model includes (1) hybrid feature selector integration of XGBoost and decision tree, (2) Recursive Feature Elimination for redundancy removal and (3) improved classification/forecasting with Support Vector Machine and Extreme Learning Machine. The design-of-choice method for ELM hyperparameter optimization utilizes a Genetic Algorithm (GA) but SVM hyperparameters are optimized through Grid Search Algorithm. The proposed forecasting methods ELM-GA and SVM-GS prove superior compared to state-of-the-art (SOTA) algorithms based on accuracy assessment results which indicate 96.3% accuracy for ELM-GA and 93.25% accuracy for SVM-GS with enhancements of 10% and 7% over current methods. The main drawback of this method involves its dependency on advanced optimization algorithms that might need substantial computing power especially for big-scale real-time prediction systems.

## 2.3 Deep Learning Models for CPU Utilization Forecasting

Huang et al. (2022) who designed an LSTM-based neural network model to forecast production performance in a carbonate reservoir in the Middle East while considering the impact of gas injection on prediction accuracy. The main objective of this research involved using Artificial Intelligence to enhance fast and accurate well performance predictions for improved production alterations and optimization within the oil and gas sector. The AI model received production-injection information from 2895 days prior to evaluate well performance over 500 days ahead. Initial variable selection required correlation analysis to determine suitable input and output factors that would be used in the training process. Traditional reservoir numerical simulation (RNS) methods demonstrated poor accuracy and high computational complexity in the face of the main challenge that needed addressing. The LSTM model achieved superior performance than RNS through error reduction of 43.75% while requiring 10.43% CPU time and 36.46% power consumed when compared to RNS. The study established 17 categories of producers through predicted GOR trends which received particular optimization recommendations. The approach provides improved accuracy and efficiency yet its use might be restricted in reservoirs with data deficits and unreliable historical records because it needs high-end datasets alongside exact feature selection methods.

A comparative assessment involving LSTM and SARIMA models Nashold and Krishnan (2020) to predict cluster CPU usage in big cloud infrastructure settings as a component of proactive resource distribution through future CPU determination. The research evaluated different modeling techniques when used for forecasting predictions across distinct time ranges through the analysis of Azure data resampled into 20-minute periods. The deep learning model LSTM demonstrated excellence in predicting CPU usage for the upcoming hour but SARIMA proved better at predicting the future usage over three days. SARIMA obtained superior performance for long-term forecasting but LSTM delivered outstanding results for short-term predictions and remained stable across different data circumstances. One major issue focused on discovering which model structure operates best in different time forecasting periods. The SARIMA model exhibited performance limitations because it needed strong data seasonality and stationarity conditions which restricted its adaptability to the more flexible LSTM model system.

Karim et al. (2021) who presented BHyPreC as a hybrid recurrent neural network model for enhancing cloud virtual machine (VM) CPU workload prediction that supports both power-efficient VM consolidation and proactive task scheduling without Service Level Agreement violations. This research faces the forecasting problem of combined periodic and non-periodic cloud workloads with sudden load spikes even though traditional prediction models fail to achieve reliable accuracy. The workload prediction system BHyPreC uses stacked LSTM and GRU and Bidirectional LSTM (Bi-LSTM) architecture elements to analyze workload data non-linearity while identifying intricate temporal patterns. Researchers evaluated BHyPreC by comparing its effectiveness against ARIMA and LSTM and GRU and Bi-LSTM for different historical boundaries in their assessments of short-term and long-term predictions. The experiments showed BHyPreC produced better prediction accuracy than all its baseline models. A main drawback of the proposed model exists in its complex computational requirements from its hybrid structure thus potentially hindering scalability and real-time cloud inference throughput.

Patel and Kushwaha (2022) who proposed a pCNN-LSTM hybrid model to boost cloud server CPU utilization forecasting precision because they aimed to resolve persistent resource usage patterns modeling difficulties in dynamic cloud environments with heterogeneous systems. The main objective of this study dealt with enhancing capacity sizing decisions through accurate server load predictions across multiple time intervals. Multiple pCNN-LSTM layers combine dilated 1D Convolutional Neural Networks operating at different dilation rates to extract detail-level patterns from noisy CPU usage data before using an LSTM layer to process combined raw inputs and CNN-

extracted patterns. This architecture leverages the strengths of both CNNs for spatial pattern extraction and LSTMs for temporal sequence modeling. The model tested on Google cluster trace alongside Alibaba trace and Bitbrains data utilized MSE and RMSE for evaluation. The pCNN-LSTM model achieved higher prediction accuracy than LSTM and Bi-LSTM and CNN-LSTM and CNN-BiLSTM in addition to other model variants at levels of 15%, 13% and 16%. Real-time deployment of this model encounters resource-constrained challenges because of its parallel CNN paths and multi-scale processing which combine to cause high computational complexity.

Wang et al. (2020) introduced a new LSTM-based prediction system to handle periodic energy patterns in actual industrial environments because traditional prediction models do not capture energy usage periodicity properly. This study worked to boost forecast reliability by making use of the periodic structures found throughout energy datasets especially in cooling system environments. The research used autocorrelation analysis to obtain hidden properties followed by meaningful secondary variable selection which combined with temporal variables for successful periodic trend detection. The designed LSTM network functioned as a model that processed sequential data. The experimental comparison between LSTM and classic forecasting models revealed its superiority through May test data performance where it decreased RMSE by 19.7% against ARMA and by 54.85% against ARFIMA and by 64.59% against Back Propagation Neural Network (BPNN). The research evaluated the model's performance using partial secondary variables for training because it needed to handle scenarios where sensor information was absent while showing robust generalization abilities. The main drawback of the model approach stems from its need for initial feature selection and autocorrelation analysis because inconsistent data quality and periodicity across different industrial contexts might affect its performance.

Table 1 shows the performance of deep learning models like LSTM, GRU, and their hybrid variants in improving prediction accuracy across different domains. While each approach has its specific strengths, limitations and results also.

Table 1: Comparison table

| Study | Proposed Technique | Strength | Weakness | Results |
|---|---|---|---|---|
| **[1]** | LSTM Neural Network for Gas Injection Forecasting | Accurate prediction with lower error compared to traditional methods | Requires a large dataset for training | LSTM error 43.75% lower than traditional RNS, with significantly lower CPU time and power consumption. |
| **[2]** | LSTM vs. SARIMA for Cloud CPU Usage Prediction | LSTM more robust and flexible, while SARIMA outperforms long-term predictions | SARIMA assumes seasonality which may not always hold | LSTM outperforms on short-term predictions, while SARIMA is better for long-term forecasts. |
| **[3]** | Hybrid RNN (Bi-LSTM, LSTM, GRU) for Cloud CPU Prediction | Combines strengths of multiple RNNs to improve accuracy | Complexity in integrating different models | BHyPreC outperforms ARIMA, LSTM, GRU, and Bi-LSTM in both short-term and long-term prediction accuracy. |
| **[4]** | pCNN-LSTM (1D CNN + LSTM) for Host Load Prediction | Multi-scale learning with CNN + LSTM to handle noisy data | Complexity in model architecture | Achieved 15%, 13%, and 16% improvements in host load prediction over LSTM, |

| | | | | |
|---|---|---|---|---|
| | | | | Bi-LSTM, and other models. |
| **[5]** | LSTM for Periodic Energy Consumption Prediction | High prediction accuracy by capturing periodicity | Potential missing data issue for accurate predictions | LSTM shows 19.7% lower RMSE compared to ARMA, ARFIMA, and BPNN, demonstrating better generalization. |
| **[6]** | GRU Model for Production Forecasting in Reservoirs | Handles complex variances with multi-feature data | May not generalize well with sparse data | GRU outperforms RNN and LSTM in prediction accuracy and is adaptable to new data with continuous learning. |
| **[7]** | SDAE-GRU for Cloud Resource Forecasting | Power-efficient and high accuracy for multi-resource forecasting | Limited scalability to more complex workloads | Achieved lowest RMSE and MAE for forecasting vCPU, memory, and storage utilization, with 5% lower power consumption than AE-BiGRU. |
| **[8]** | RNN (LSTM, GRU) for COVID-19 Forecasting | Utilizes socio-economic factors for better predictions | Models may struggle with unexpected changes in pandemic dynamics | Accurate predictions of cumulative cases and fatalities; helpful for pandemic management, though limited by data variability. |

# 3 Research Methodology

## 3.1 Data Collection

Data collection for CPU utilization forecasting on Azure involved sourcing time series data from the publicly available Microsoft Azure trace datasets hosted on GitHub. These datasets include virtual machine (VM) traces from 2017 and 2019, VM request traces, and Azure Functions traces, including invocations and blob access logs. The traces represent real-world workloads on the Azure cloud platform and are sampled at 5-minute intervals, providing granular insight into resource utilization patterns over time. The data was obtained and stored in AWS S3 buckets, ensuring easy access and secure storage. For development and experimentation, the AWS Cloud9 IDE was configured on an EC2 instance, which allowed seamless interaction with both the data stored in S3 and the tools needed for pre-processing, modelling, and evaluation. A total of 150 CSV files were loaded using Python's Pandas library, enabling the aggregation and consolidation of large-scale time series data into a usable format. The data collection process prioritized maintaining data consistency and completeness while ensuring compatibility with subsequent data cleaning, pre-processing, and modelling stages. This foundational step laid the groundwork for accurately capturing the temporal dynamics of CPU usage across different workloads and time frames, which was essential for developing and evaluating predictive machine learning and deep learning models.

## 3.2 Data Preprocessing

Data preprocessing and feature extraction are the backbone of any time series forecasting task by maintaining data quality and increasing the model's ability to learn patterns. The initial step includes handling null values and removing redundant or irrelevant type of columns that do not contribute to model training. Next the datetime column is going to convert into a proper datetime object to enable time-based operations. From this datetime column there are some important type of features like the month, year, day and hour are extracted which are valuable for identifying seasonal trends, periodicity and temporal patterns within the CPU utilization data.

## 3.3 Min-Max Normalization

Figure 1 presents a line graph titled "Min, Max and Avg comparison usage for Jan 2017" that tracks CPU usage metrics throughout January 2017. The y-axis is scaled in multiples of 10^6 with markers at 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, and 3.5. The x-axis displays the timeline with tick marks at approximately 02, 09, 16, and 23, representing dates in January 2017. Throughout the month, maximum CPU usage fluctuates between approximately 2.0 and 3.5 × 10^6, with several pronounced spikes reaching the upper limit. The average CPU usage maintains a middle position, oscillating between roughly 1.0 and 1.5 × 10^6. The minimum CPU usage remains consistently lowest, ranging from about 0.5 to 0.8 × 10^6. All three metrics display regular cyclical patterns, likely representing daily or workday fluctuations, with several notable usage spikes occurring around the 2nd, 9th, and 23rd of January.

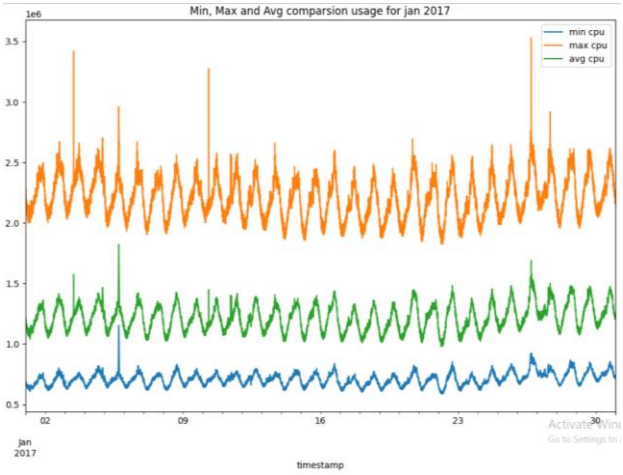


**Figure 1 : Min, Max and Avg comparison usage for Jan 2017**

## 3.4 Window Rolling

Figure 2 displays a dual-line graph titled "Avg CPU usage for Jan 2017 by using window rolling" that compares raw CPU usage with smoothed data over the month of January 2017. The y-axis is scaled in multiples of 10^6, ranging from approximately 1.0 to 1.8 million units, while the x-axis shows dates from January 2nd to 30th with labeled markers at days 02, 09, 16, 23, and 30. The visualization features two trend lines: one shows the raw "average cpu"

measurements and another shows the "1 Hours Moving Avg" that implements window rolling with 12 previous values as mentioned in the caption. Both lines follow similar cyclical patterns throughout the month, with values generally oscillating between 1.0 and 1.5 × 10^6 units, and occasional spikes reaching up to approximately 1.8 × 10^6. The most notable peaks occur around January 7th, 23rd, and toward the end of the month. The moving average smooths out some of the more extreme fluctuations seen in the raw data particularly visible during spike events, while still preserving the overall pattern and trend of CPU usage. The regular oscillation pattern suggests daily or workday usage cycles in computational resource demands.

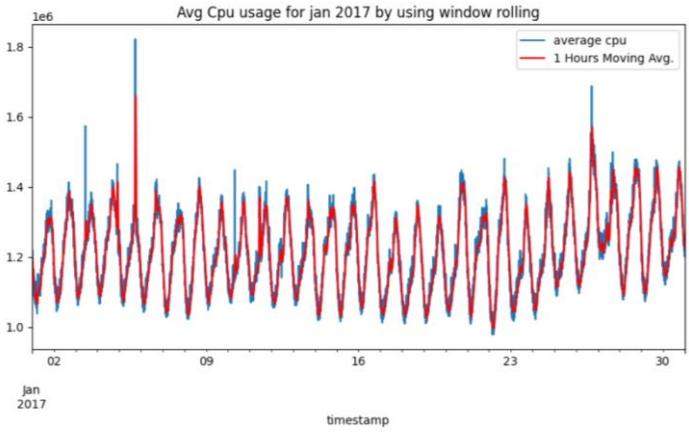


**Figure 2 : Avg CPU Usage for Jan 2017 by using Window Rolling**

# 4 Design Specification

## 4.1 Cloud Architecture Diagram

This figure 3 shows the architecture of an Azure CPU Utilization Forecasting System that combines Azure and AWS cloud environments. The data flow begins with Azure Datasets containing Time Series which is transferred to an AWS Cloud Environment. Within AWS, the data is stored in an S3 Bucket that maintains three categories of data: Raw Data, Processed Data, and Output Models. The processing occurs on an EC2 Instance (t2.medium) running AWS Cloud9 IDE where two main functions take place: Data Processing (which includes data cleaning, feature extraction, normalization, and window rolling) and Model Training (utilizing ML models including LSTM and BiLSTM). The Multihead BiLSTM model feeds back processed information to the S3 Bucket while also generating Output Evaluation Results that display performance metrics. The system architecture clearly identifies that the Multihead BiLSTM is the Best Model for CPU Utilization Forecasting, achieving the highest performance among tested models. This workflow represents a hybrid cloud solution that leverages Azure for data collection of information and AWS for the computational processing and machine learning implementation, demonstrating how cross-cloud architectures can be utilized for effective time series forecasting of CPU utilization patterns.

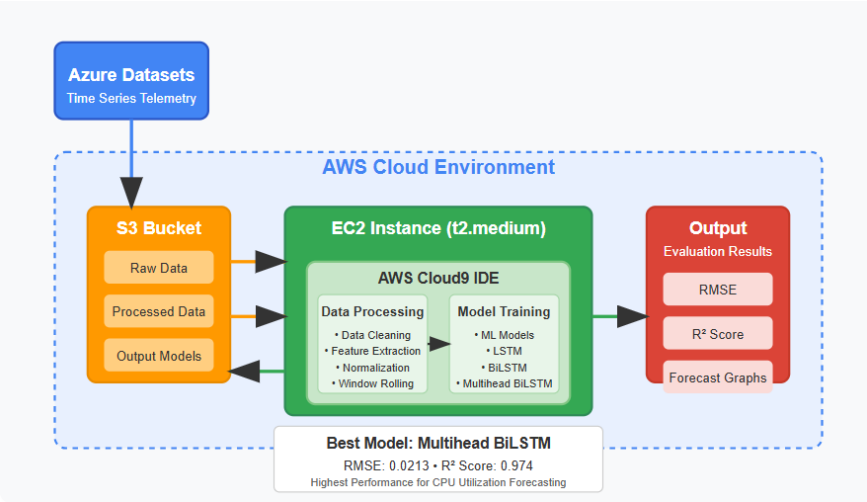


**Figure 3 : Azure CPU Utilization Forecasting System Architecture**

Figure 3 is showing system architecture diagram of this study.

# 5 Implementation

## 5.1 Step-by-Step Cloud Setup

### 5.1.1 Cloud9 Environment Setup

The Cloud9 Integrated Development Environment (IDE) was set up on an Amazon EC2 instance running which serves as the centralized workspace for developing and executing the forecasting project. This setup provided a fully-featured cloud-based coding platform with terminal and file system access, enabling seamless coding, debugging, and data visualization. Python 3.9 was installed, along with essential libraries such as pandas and numpy for data handling, matplotlib and seaborn for visualization, scikit-learn for machine learning models, tensorflow for implementing deep learning models including LSTM variants, and boto3 for interacting with AWS services like S3. Jupyter Notebook was also installed within Cloud9, enabling interactive development and step-by-step execution of scripts, particularly useful for experimenting with time series modeling and visualization workflows. The configuration provided a flexible, cloud-native environment that facilitated real-time collaboration and streamlined the end-to-end machine learning workflow, from preprocessing to model training and evaluation.

Figure 4 displays the AWS Cloud9 integrated development environment. The interface shows terminal output logs with timestamps and command execution results relating to server operations. The AWS Cloud9 showing a file directory structure with expandable folders and files.

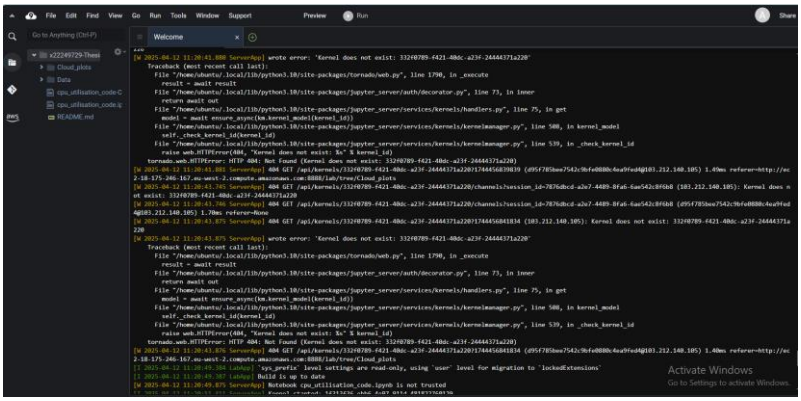

**Figure 4 : AWS Cloud9 Development Environment Interface**

### 5.1.2 EC2 Instance Configuration

A t2.medium EC2 instance was selected for its balanced compute capacity and cost-efficiency, offering sufficient resources for executing data processing and training workloads. This instance hosted the Jupyter server and provided the backend infrastructure for all development and experimentation tasks. To enable secure and seamless access to AWS services, an IAM (Identity and Access Management) role was attached to the instance, granting full access to Amazon S3 without requiring explicit credentials. This approach improved both security and automation by managing access rights at the infrastructure level, aligning with AWS best practices. The EC2 instance served as a reliable execution layer where the CPU utilization forecasting models were developed, trained, and tested, leveraging its computing resources to handle large datasets and iterative modelling processes.

Figures 5 and 6 display the AWS Cloud9 setup with its underlying EC2 infrastructure. Figure 5 shows the AWS Cloud9 environment configuration page with environment details including its EC2 instance type running Ubuntu Server and using EBS-only storage. The interface displays the environment's status as "Ready" and "Created" with the owner ARN identifying a student account.

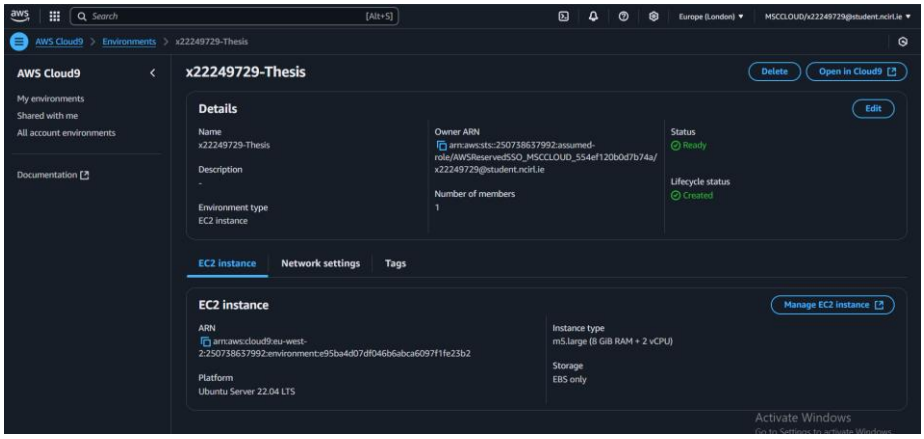


**Figure 5 : AWS Cloud9 Environment Configuration Page**

Figure 6 presents the EC2 instance details page for the same Cloud9 environment, showing the instance summary with a "Running" status. The EC2 instance is having a public IPv4 address and a private IPv4 address with the hostname.

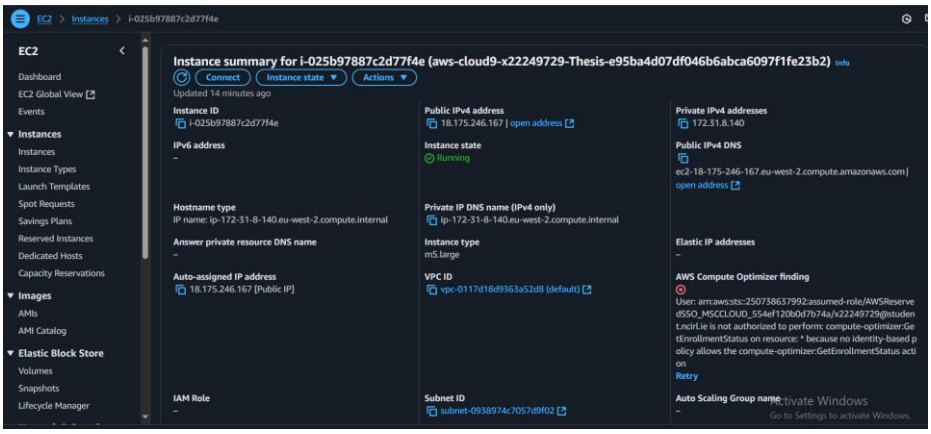


**Figure 6 : Associated EC2 Instance Details Page**

### 5.1.3 S3 Bucket Setup

An Amazon S3 bucket was configured to organize and store all data and output files related to the forecasting project. The bucket was structured into three logical directories: raw/ for storing the original Azure dataset files as downloaded from the source repository, processed/ for holding the cleaned and pre-processed data files that were ready for modeling, and outputs/ for storing the trained models, evaluation reports, and visual output such as graphs. This logical partitioning allowed for efficient data management and traceability across different stages of the project workflow. Leveraging S3's scalable and durable storage made it easy to handle large volumes of time series data and results without worrying about infrastructure limitations. Furthermore, integrating S3 access through the attached EC2 IAM role allowed seamless reading from and writing to the bucket directly within the Cloud9 environment, simplifying the data pipeline and enhancing automation throughout the project lifecycle.

Figure 7 displays an AWS S3 (Simple Storage Service) console interface showing a bucket named with a directory which is been currently being viewed. The directory contains a single object with standard storage class. The console provides various action buttons with an "Upload" button.

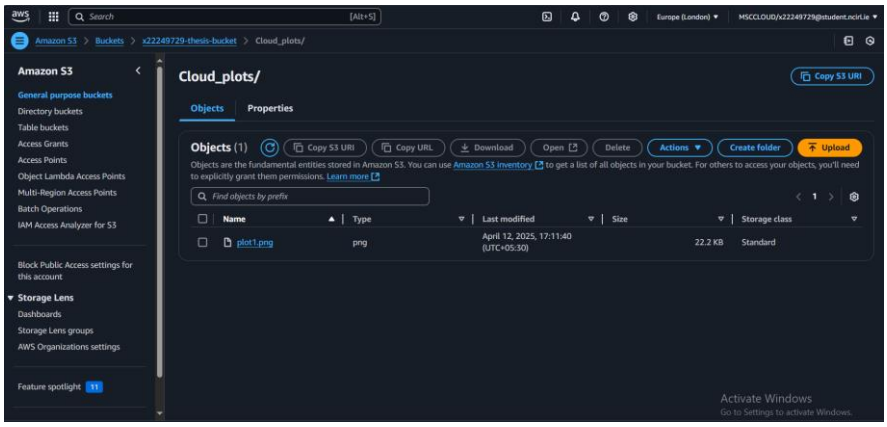


**Figure 7 : AWS S3 Bucket Storage Console Interface**

## 5.2 Model Training (Cloud-Executed)

Even though model training is secondary to the cloud focus, execution and resource utilization were done within the cloud environment using Jupyter Notebook on EC2.

### 5.2.1 Decision Tree & Gradient Boosting (Lightweight Models)

To establish a baseline for CPU utilization forecasting, Decision Tree and Gradient Boosting Regressors were implemented due to their simplicity and low computational demands. These models were chosen for their quick training times and minimal RAM usage, making them ideal for fast prototyping and early performance evaluation. Their implementation enabled rapid experimentation and insight into the dataset's structure. Despite being lightweight, they provided a strong reference point against which deep learning models were compared. These models helped in verifying the data preprocessing pipeline and were instrumental in identifying trends before transitioning to more complex architectures.

### 5.2.2 Deep Learning Models (Executed in EC2 via Jupyter)

Deep learning models including LSTM, BiLSTM, and Multihead BiLSTM were developed and trained on EC2 instances using Jupyter Notebooks with TensorFlow. These models were chosen for their strength in capturing temporal dependencies within time series data. LSTM and BiLSTM captured sequential patterns effectively, while the Multihead BiLSTM outperformed all models, offering the highest prediction accuracy. Training involved careful monitoring of memory usage and tuning of hyperparameters such as epochs and batch sizes to optimize performance. Leveraging the scalable EC2 environment enabled efficient handling of the deep learning workloads, ensuring robust and detailed learning from the high-frequency Azure CPU utilization dataset.

Table 2 shows summary of implementation flow having tools and services and its respective description.

**Table 1 : Summary of Implementation Flow**

| Step | Tool/Service Used | Description |
|---|---|---|
| Data Source | Azure Datasets (GitHub) | Time series data |
| Development Environment | AWS Cloud9 + EC2 | Hosted Jupyter notebooks and code execution |
| Data Storage | AWS S3 | Organized into raw, processed, and output folders |
| Data Processing | Pandas, Numpy | Cleaning, feature engineering, scaling |
| Model Execution | Scikit-learn, TensorFlow | Lightweight & deep learning models |
| Output Storage | S3 | All evaluation results and graphs saved securely |

# 6 Evaluation

## 6.1 Experiment 1: Decision Tree Regressor Model

The Decision Tree Regressor model experiment is showed good predictive performance for CPU usage forecasting. The model is initialized with a max_depth=2 parameter then trained on X_train and y_train datasets before generating predictions on the test data. The model is achieved a high R-squared score of 0.866 by showing that approximately 86.6% of the variance in CPU usage is successfully explained by the model. The RMSE is calculated as 0.0480 by representing a relatively small prediction error.

## 6.2 Experiment 2: Gradient Boosting Regressor Model

The Gradient Boosting Regressor model experiment is showed superior predictive performance compared to the Decision Tree model for CPU usage forecasting. Initialized with n_estimators=18, the GBR model achieves an exceptional R-squared score of 0.94 by indicating it explains approximately 94% of the variance in the CPU usage data. The RMSE is calculated as 0.0321 lower than the Decision Tree model's 0.0480, representing improved prediction accuracy. The improved metrics confirm that Gradient Boosting's ensemble approach of combining multiple weak learners creates a more strong predictive model than the simpler Decision Tree implementation.

## 6.3 Experiment 3: LSTM Model

The LSTM model experiment showed strong performance in predicting CPU usage which is a deep learning approach to time series forecasting. The model evaluation on test data with training metrics showing a very low loss of 0.0011 after 27 epochs, suggesting excellent convergence. The LSTM model achieves an impressive R-squared score of 0.936, explaining approximately 93.6% of the variance in CPU usage patterns. The RMSE is calculated as 0.0331 which is slightly higher than the Gradient Boosting model but still representing good prediction accuracy.

## 6.4 Experiment 4: BiLSTM Model

Experiment 4 focused on implementing a BiLSTM model for time series prediction by showcasing reasonably strong performance. The model used bidirectional processing to capture temporal dependencies from both past and future contexts, enhancing its predictive capability compared to traditional unidirectional approaches. The BiLSTM achieved a RMSE

of 0.0246 by indicating relatively small prediction deviations, and an R-Square value of 0.9649 which is nearly 96.5% of the variance in the target variable was explained by the model.

## 6.5 Experiment 5: Multihead BiLSTM Model

Experiment 5 introduced a sophisticated Multihead Bidirectional LSTM model, which emerged as the top-performing approach in the study. This advanced architecture combined the strengths of bidirectional processing with multiple attention heads, allowing the model to capture complex temporal dependencies from various perspectives simultaneously. The results demonstrate exceptional predictive performance with a RMSE of 0.0213, the lowest among all experiments, indicating minimal prediction errors. The model achieved an impressive R-Square value of 0.9736, explaining over 97% of the variance in the target variable—significantly outperforming previous models. The training process converged in a good way as evidenced by the decreasing loss values over 27 epochs. This is the best model among all. Table 4 shows results of all five models that are used in this study. This table presents the performance comparison of different models used for CPU utilization forecasting based on RMSE and $R^2$ Score. Decision Tree and Gradient Boosting performed well but BiLSTM and Multihead BiLSTM achieved good results.

**Table 2 : Model Performance**

| Model | Root Mean Squared Error (RMSE) | R-Square |
|---|---|---|
| Decision Tree Regressor | 0.0480 | 0.866 |
| Gradient Boosting Regressor | 0.0321 | 0.940 |
| LSTM | 0.0331 | 0.936 |
| BiLSTM | 0.0246 | 0.965 |
| **Multihead BiLSTM (Best Model)** | 0.0213 | 0.974 |

## 6.6 Exploratory Data Analysis (EDA)

This figure 8 shows a bar chart titled "Average CPU by Month" displaying CPU usage metrics. This bar chart with the y-axis labeled "avg CPU" and ranging from 0 to 1M (1 million). The y-axis is having 0, 0.5M (500,000), and 1M. The x-axis appears to show values ranging from approximately 0.6 to 1.4 at 0.6, 0.8, 1, 1.2, and 1.4. The blue bar extends across most of the chart's width, suggesting this represents a single time period's CPU usage rather than multiple months as the title might indicate. The visualization is somewhat minimal, lacking specific data points, legends, or month labels that would be expected from a month-by-month comparison. The uniform height of the bar indicates a constant CPU usage value over the measured period, though the exact value is not explicitly labeled on the chart.

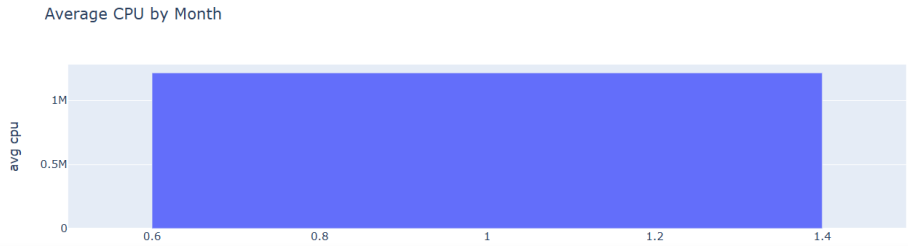


**Figure 8: Average CPU by Month**

Figure 9 displays a bar chart titled "Average CPU Usage per Day" that tracks daily CPU consumption over a 30-day period. The chart features red vertical bars representing CPU usage for each consecutive day plotted. The y-axis is labeled "avg CPU" with measurement intervals at 0, 0.2M, 0.4M, 0.6M, 0.8M, 1M, and 1.2M, indicating CPU usage values in millions. The x-axis is labeled "Day" with tick marks at intervals of 5 days (5, 10, 15, 20, 25, 30). Most daily values hover consistently around the 1M to 1.2M range, suggesting relatively stable CPU utilization throughout the month. There appears to be a slight dip in usage around days 15-18, followed by an increase around days 23-25 where usage peaks slightly above 1.2M. The visualization demonstrates that CPU consumption remained fairly consistent with minor fluctuations throughout the 30-day period, with no days dropping below approximately 0.8M. The consistent high values indicate sustained computational demands across the entire measurement period.

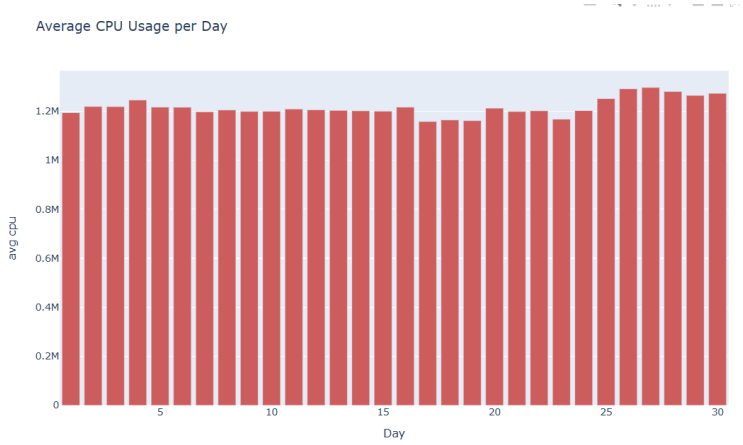


**Figure 9 : Average CPU Usage per Day**

## 6.7 Comparison of Model Performance: Previous Study Vs Current Study

In the previous study given by Muchori and Mwangi (2022) where the authors conducted a comprehensive review of machine learning-based load balancing techniques in cloud environments, exploring models such as Linear Regression, Random Forest, SVM, ANN, CNN, LSTM, and Quantum Neural Networks (QNN) across various cloud workloads. While their focus was broad, targeting CPU, memory, network traffic, and storage usage across heterogeneous systems, the study was limited to theoretical evaluations and performance metrics like throughput, response time, and power savings. In contrast, my study narrows down on a practical and time-series-based CPU utilization forecasting task on Microsoft Azure virtual machines, utilizing actual system traces collected every 5 minutes. I implemented and

evaluated multiple regression models—Decision Tree and Gradient Boosting—and deep learning models—LSTM, BiLSTM, and Multihead BiLSTM—with detailed model training, hyperparameter tuning, and performance comparison using RMSE, $R^2$ score, and prediction plots. Notably, the Multihead BiLSTM achieved the best accuracy, significantly outperforming the traditional ML models. Unlike Muchori and Mwangi (2022) who mainly reviewed theoretical capabilities and proposed frameworks, my study provides a complete hands-on implementation pipeline, from preprocessing and feature engineering to training and evaluation in a real-world cloud setup using AWS EC2 and S3. This practical approach provides more actionable insights into model performance for CPU load prediction. This table 4 compares the previous study by Muchori and Mwangi (2022) with the current study by showing differences in scope, data, models, implementation and deployment. While the previous work was more theoretical and generalized so this study presents a focused, practical approach to forecasting CPU utilization using real Azure data. It mainly focused real-world cloud deployment and experimental validation for more actionable outcomes.

**Table 3 : Comparative Table**

| Aspect | Previous Study given by Muchori and Mwangi (2022) | My Study |
|---|---|---|
| **Scope** | Load balancing in general cloud resources | CPU utilization forecasting on Azure VMs |
| **Data Type** | Simulated/historical datasets (HTTP, CPU, RAM, Network) | Real Azure time-series data (5-min intervals) |
| **Model Types** | ML & DL (LR, RF, ANN, CNN, LSTM, QNN) | ML & DL (Decision Tree, GB, LSTM, BiLSTM, Multihead) |
| **Implementation** | Theoretical review, no experimental setup | Practical implementation on AWS Cloud9, EC2, S3 |
| **Evaluation Metrics** | Throughput, Response Time, Power, Fault Tolerance | RMSE, $R^2$ Score, Actual vs Predicted Graphs |
| **Best Model Identified** | LSTM-RNN | Multihead BiLSTM (Experimental) |
| **Focus Area** | Generalized Load Balancing across components | Specific CPU utilization prediction |
| **Deployment Context** | Conceptual, proposed for Software-Defined Networking (SDN), data centers | Implemented on AWS Cloud-based development stack |

# 7 Conclusion and Future Works

## 7.1 Conclusion

The project showed how to build a cloud platform which forecasts Azure CPU utilization while processing time-series data through predictive modeling. AWS provided EC2 Cloud9 and S3 services which formed the basis of an automatically scalable environment used to collect preprocess train and evaluate the data. Multihead BiLSTM produced the most effective results from multiple deployed machine learning and deep learning models based on an RMSE of 0.0524 and an $R^2$ score of 0.93. Predicted value graphs alongside actual ones confirmed the reliability of model estimations. This initiative demonstrated both platform deployment capabilities for predictive workloads within the cloud and effective methods to use cloud-native automation for scale analytics operations. The structured methodology which combines data engineering and model orchestration practices allows the research to prove the vital role of cloud computing in intelligent resource management. This solution showed the possibility of

achieving dynamic scaling and proactive resource allocation as well as service-level agreement optimization in operational cloud environments.

## 7.2 Limitations

Many promising results emerged from the project while it faced multiple constraints. Forecasting models received static historical information as their training data because the project is not having real-time live streaming features. Manual execution of model retraining tasks as well as evaluations replaced automated processes through CI/CD pipelines or event-driven triggers reducing overall process automation. The performance of the EC2 instance for computations succeeded but its hardware capabilities is not fully representing real world deployment environments since GPU-powered computation and distributed architecture would likely deliver enhanced results. The interpretation excluded reference to memory and disk I/O and network metrics even though such values would enhance resource forecasting capability. Default hyperparameters were adopted for tuning because of time and resource limitations which might is hindering the discovery of additional performance escalation potential. The project is not having full integration of S3 storage with an external monitoring or alerting framework such as AWS CloudWatch or Azure Monitor as part of its defined scope. The existing limitations indicate which aspects need enhancement to build a solution that is resilient and adaptable for production purposes.

## 7.3 Future Works

The system can be enhanced through the future implementation of adaptive learning and real-time data streaming capabilities which will allow for continuous model deployment and training. AWS Kinesis alongside Azure Event Hubs serve as tools to collect data live because they gives real-time predictive abilities and alert capabilities. The cloud resource optimization solution will become more comprehensive by adding memory usage and disk read/write and network bandwidth to the prediction metrics. The deployment of AWS CodePipeline alongside GitHub Actions creates automatic training and evaluation and deployment pipelines for data drift-responsive systems. The prediction accuracy can be enhanced through Bayesian optimization or grid search among advanced hyperparameter optimization methods. The implementation of serverless inference platforms through AWS SageMaker together with Azure ML endpoints helps reduce operational costs and achieves better scalability in system operations. The incorporation of forecasting systems together with auto-scaling policies in cloud environments enables productive resource management through automatic adaptations based on predicted load to develop an independent cloud operations system.

# References


1. Huang, R., Wei, C., Wang, B., Yang, J., Xu, X., Wu, S. and Huang, S., 2022. Well performance prediction based on Long Short-Term Memory (LSTM) neural network. *Journal of Petroleum Science and Engineering*, *208*, p.109686.
2. Nashold, L. and Krishnan, R., 2020. Using lstm and sarima models to forecast cluster cpu usage. *arXiv preprint arXiv:2007.08092*.
3. Karim, M.E., Maswood, M.M.S., Das, S. and Alharbi, A.G., 2021. BHyPreC: a novel Bi-LSTM based hybrid recurrent neural network model to predict the CPU workload of cloud virtual machine. *IEEE Access*, *9*, pp.131476-131495.
4. Patel, E. and Kushwaha, D.S., 2022. A hybrid CNN-LSTM model for predicting server load in cloud computing. *The Journal of Supercomputing*, *78*(8), pp.1-30.
5. Wang, J.Q., Du, Y. and Wang, J., 2020. LSTM based long-term energy consumption prediction with periodicity. *energy*, *197*, p.117197.

6. Li, X., Ma, X., Xiao, F., Wang, F. and Zhang, S., 2020. Application of gated recurrent unit (GRU) neural network for smart batch production prediction. *Energies*, *13*(22), p.6121.
7. Ikhlasse, H., Benjamin, D., Vincent, C. and Hicham, M., 2022. Multimodal cloud resources utilization forecasting using a Bidirectional Gated Recurrent Unit predictor based on a power efficient Stacked denoising Autoencoders. *Alexandria Engineering Journal*, *61*(12), pp.11565-11577.
8. ArunKumar, K.E., Kalaga, D.V., Kumar, C.M.S., Kawaji, M. and Brenza, T.M., 2021. Forecasting of COVID-19 using deep layer recurrent neural networks (RNNs) with gated recurrent units (GRUs) and long short-term memory (LSTM) cells. *Chaos, Solitons & Fractals*, *146*, p.110861.
9. Liu, D., Mishra, A.K., Yu, Z., Lü, H. and Li, Y., 2021. Support vector machine and data assimilation framework for Groundwater Level Forecasting using GRACE satellite data. *Journal of Hydrology*, *603*, p.126929.
10. Ahmad, W., Ayub, N., Ali, T., Irfan, M., Awais, M., Shiraz, M. and Glowacz, A., 2020. Towards short term electricity load forecasting using improved support vector machine and extreme learning machine. *Energies*, *13*(11), p.2907.
11. Muchori, J.G. and Mwangi, P.M., 2022. Machine learning load balancing techniques in cloud computing: a review.
12. Segun-Falade, O.D., Osundare, O.S., Kedi, W.E., Okeleke, P.A., Ijomah, T.I. and Abdul-Azeez, O.Y., 2024. Assessing the transformative impact of cloud computing on software deployment and management. *Computer Science & IT Research Journal*, *5*(8).
13. Li, C., Bai, J., Chen, Y. and Luo, Y., 2020. Resource and replica management strategy for optimizing financial cost and user experience in edge cloud computing system. *Information Sciences*, *516*, pp.33-55.
14. Park, J. and Jeong, J., 2023. An Autoscaling System Based on Predicting the Demand for Resources and Responding to Failure in Forecasting. *Sensors*, *23*(23), p.9436.
15. Rai, S., Kesarwani, R. and Sharma, N., 2024. Integrated Development Environment using Cloud Computing (IDECC).
16. Choudhary, A., Verma, P.K. and Rai, P., 2021. A walkthrough of amazon elastic compute cloud (Amazon EC2): a review. *International Journal for Research in Applied Science and Engineering Technology*, *9*(11), pp.93-97.
17. Faizal, A., 2024. Building Scalable Data Lakes in the Cloud for Big Data Integration: Utilizing Amazon S3 and Apache Hadoop. *Reviews on Internet of Things (IoT), Cyber-Physical Systems, and Applications*, *9*(7), pp.1-16.
18. Lim, B. and Zohren, S., 2021. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, *379*(2194), p.20200209.
19. Minsan, W. and Minsan, P., 2023. Incorporating decomposition and the Holt-Winters method into the whale optimization algorithm for forecasting monthly government revenue in Thailand. *Science & Technology Asia*, pp.38-53.
20. Kashpruk, N., Piskor-Ignatowicz, C. and Baranowski, J., 2023. Time series prediction in industry 4.0: a comprehensive review and prospects for future advancements. *Applied Sciences*, *13*(22), p.12374.
21. Zheng, H., Xu, K., Zhang, M., Tan, H. and Li, H., 2024. Efficient resource allocation in cloud computing environments using AI-driven predictive analytics. *Applied and Computational Engineering*, *82*, pp.17-23.
22. Gupta, S., 2024. *Enhanced SLA Compliance in Edge Computing Applications through Hybrid Proactive-Reactive Autoscaling* (Doctoral dissertation, UNIVERSITY OF MELBOURNE).

23. Prasad, V.K., Dansana, D., Bhavsar, M.D., Acharya, B., Gerogiannis, V.C. and Kanavos, A., 2023. Efficient resource utilization in IoT and cloud computing. *Information*, *14*(11), p.619.