National College of Ireland

Attention-Enhanced Bidirectional Deep Learning for Short-Term CPU Usage Forecasting in Cloud Resource Allocation

MSc Research Project
Cloud Computing

E. Hemanth Kumar
Student ID: x22183744

School of Computing
National College of Ireland

Supervisor: Shaguna Gupta

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	E. Hemanth Kumar
Student ID:	x22183744
Programme:	Cloud Computing
Year:	2024-2025
Module:	MSc Research Project
Supervisor:	Shaguna Gupta
Submission Due Date:	02/01/2026
Project Title:	Attention-Enhanced Bidirectional Deep Learning for Short-Term CPU Usage Forecasting in Cloud Resource Allocation
Word Count:	8272
Page Count:	25

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:	
Date:	2nd January 2026

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):	

Attention-Enhanced Bidirectional Deep Learning for Short-Term CPU Usage Forecasting in Cloud Resource Allocation

E. Hemanth Kumar
x22183744

Abstract

Cloud computing environments have experienced highly dynamic and unpredictable workloads by making powerful and good CPU resource allocation as a challenge for maintaining performance, scalability, and cost efficiency. There are some traditional rule-based and reactive autoscaling mechanisms who fail to anticipate short-term workload fluctuations which results in resource underutilization or service degradation. This study solves the problem of short-term CPU usage forecasting for intelligent cloud resource allocation using deep learning–based predictive models. The study has been proposed an attention-enhanced Bidirectional Long Short-Term Memory model with Cross-Attention Tensor Fusion (BiLSTM-CATF) to improve forecasting accuracy by selectively focusing on the most relevant temporal dependencies in cloud workload time-series data. This study have implemented and deployed on an AWS cloud environment using EC2 for computation, Cloud9 for development and experimentation and SNS for performance notification and monitoring. This study shows that BiLSTM-CATF achieved the highest prediction error (RMSE = 0.0223, R^2 = 0.9709) which outperforms baseline models while capturing complex temporal patterns. This study advances the state of the art by combining cross-attention into bidirectional temporal modeling for cloud workloads.

Keywords: Cloud Computing, CPU Usage Forecasting, Bidirectional LSTM, Attention Mechanism, Resource Allocation

1 Introduction

This chapter presents the history behind cloud computing and describes the difficulties of dynamically allocating resources to meet changing workloads. It identifies the motivation of the research, statement of the problem, research question, and objectives aimed at the prediction of deep learning-based CPU usage. The chapter further observes the study limitations and gives general organization of the report.

1.1 Background

Cloud computing has changed the nature and way computational resources are provided

with scalability, flexibility, and cost effectiveness to both small and big organizations

Akoh Atadoga, U. J. et al. (2024). But since workloads and user demands are dynamically changing, effective allocation of resources has become a vital issue Prasad et al. (2023). A lack of proper allocation may cause the lack of resource utilization, latency, and operations costs. The existing rule-based and static assignment methods are usually unable to respond to the dynamism of the workloads in the cloud environment in real-time. To address these shortcomings, the adoption of machine learning and deep learning models is going up to process historical and real-time data to be used to make smart decisions. Using big data, like BitBrains, Microsoft Azure and Google Cluster, predictive models can predict CPU usage, memory consumption, and network throughput which will optimize the provisioning of resources Demirbaga et al. (2024). By combining smart systems with cloud computing platforms such as Amazon EC2 and S3, the system can be scaled and used intelligently based on data and automated. Therefore, this research will contribute to improving the performance and cost optimization based on predictive modeling and smart resource allocation policy.

The figure 1 illustrates the distribution of resources in the cloud computing with the help of cloud brokering system. The resource demands that are made by the users are sent via their computer terminal to a centralized cloud broker which is fitted with an optimization algorithm. This smart broker evaluates the needs of the virtual machine resources through various cloud providers in a way that maximizes the performance, cost and availability Hamzah Khan et al. (2024). The two-way arrows indicate that there is constant communication between the cloud providers and the broker and that resource provisioning is dynamic.

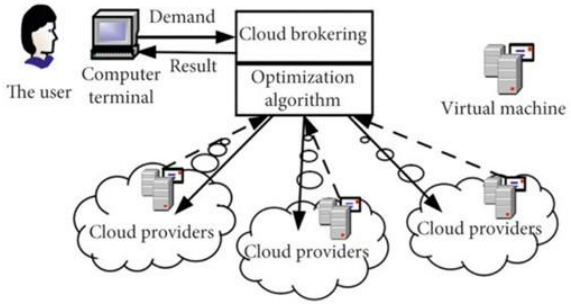

Figure 1: Resource Allocation in Cloud Computing by Shi and Lin (2022)

1.2 Motivation

The intensive growth of cloud computing has subjected resource allocation an important concern in making sure that the performance, scalability and cost-effectiveness is guaranteed. Conventional fixed provisioning will tend to cause either over or underutilization of resources, which creates inefficiencies and wastage of resources. Predictive and smart allocation strategies have been introduced with the emergence of sophisticated workloads. Deep learning and machine learning models can be used to predict demand by analyzing the historical trends in cloud workloads. The research is driven by the concept of increasing the performance of the cloud with the help of data-based optimization to facilitate the dynamic, automated, and intelligent distribution of resources among the distributed clouding systems.

1.3 Problem Statement

Although cloud infrastructure has made tremendous progress, the allocation of resources remains unbalanced in their provisioning, workload behavior is hard to predict, and very few resources are automatically allocated. Traditional rule-based or threshold-based approaches are not responsive to real-time changes to deteriorate performance and lead to unwanted expenses. The problem which addressed in the study is the efficient prediction and allocation of cloud resources using deep learning models. This paper will utilize the capabilities of time-series data, e.g., CPU and memory usage, to construct intelligent models to forecast the demand of the workload, thus enabling efficient and dynamic resource scheduling and allocation in dynamic cloud computing systems.

1.4 Research Question

"How does incorporating cross-attention into a Bidirectional LSTM (BiLSTM) architecture affect the accuracy and computational performance of short-term CPU usage forecasting for cloud resource allocation?"

1.5 Problem Solution

To address the limitations of reactive and rule-based cloud autoscaling mechanisms this study has proposed a predictive deep learning–based solution for short-term CPU usage forecasting. The proposed approach has used a BiLSTM network with a Cross-Attention Tensor Fusion (CATF) mechanism to capture complex temporal dependencies in cloud workload data. By selectively focusing on the most relevant time steps the model has improved forecasting accuracy and performance.

1.6 Research Objectives

This study does have two important research objectives which includes:

1. To evaluate how using a cross-attention mechanism within a bidirectional LSTM architecture improves the accuracy of short-term CPU usage forecasting in dynamic cloud environments.

2. To analyze the impact of attention-enhanced deep learning models on computational performance and scalability for intelligent cloud resource allocation.

1.7 Limitations of the Study

This study has several limitations that can affect the final results and the applicability of the results. The number of Microsoft Azure dataset was not used at its full extent, so it is possible that it did not accurately reflect large-scale cloud workloads. Experiments were performed in a controlled AWS setting, i.e. real-life aspects of the network (instability, workload bursts, hardware crashes, etc.) could not be fully emulated. The deep learning models are also time-consuming and need high-performance computer hardware, which restricts the possibility of extensive search of hyperparameters. It is also difficult to interpret the models, which are usually black boxes in deep learning structures. Finally, the study mainly focuses on CPU and memory indicators, without considering other

parameters like energy efficiency, cost forecasting, or cross-cloud orchestration, which may be pursued in further studies to have a more holistic optimisation.

1.8 Structure of the Study

This report has been divided into seven main chapters. Chapter 1 provides an overview of the research which includes the study background, motivation, problem definition, research question, objective and report organization. Chapter 2 shows a good review of existing literature on cloud resource management, autoscaling techniques, machine learning and deep learning–based forecasting and identified research gaps. Chapter 3 explains the adopted methodology by focusing on the dataset, preprocessing steps and all. Chapter 4 shows the system design and architectural framework with model justification. Chapter 5 discusses the implementation of proposed models, forecasting strategy and cloud deployment. Chapter 6 analyzes experimental results and performance evaluation. Chapter 7 summarizes conclusions and future research scope.

2 Related Work

The chapter is a review on the development of cloud resource management beyond the concept of the static provisioning and into an intelligent and predictive autoscaling system. It speaks of machine learning, reinforcement learning and deep learning methods, especially LSTM and BiLSTM methods in temporal workload prediction in cloud and other related fields. The conclusion drawn at the end of the chapter is the identification of research gaps in connection with the existing research and the state of the art as a reference of the research which requires consideration on attention-enhanced bidirectional models and practical validation on the cloud.

2.1 Evolution of Resource Management in Cloud Environments

Cloud computing initially relied on manual provisioning and static allocation to manage computational resources Ali and Zeebaree (2024). Early models assumed predictable workloads and fixed scaling parameters, which often led to over- or under-provisioning. As the demand for agility grew, dynamic allocation techniques emerged, where system resources could be adjusted at runtime based on workload fluctuations. Virtualization and containerization technologies such as VMware, Docker, and Kubernetes transformed infrastructure flexibility by enabling resource isolation, replication, and orchestration Agrawal and Singh (2024).

With time the scope of managing resources changed to include not only utilization performance but also intelligent automation. As the distributed workloads and multi-tenant environments became more complex, some of the past research started to investigate the predictive and learning-based scaling methods. These methods rely on the history of utilization to predict future needs through the delivery of proactive as opposed to reactive scaling. This development incorporate a more paradigm shift to autonomous cloud systems where the infrastructure is self-learned, adapted and optimized so that it does not need much human intervention.

2.2 From Reactive to Predictive Autoscaling: The Need for Intelligence

Traditional autoscaling systems like AWS Auto Scaling or Azure VM Scale Sets rely on reactive events, scaling on the occurrence of preestablished limits (e.g. CPU utilization or memory) only Betti Pillippuge et al. (2025). These mechanisms are effective in constant workload but they cannot anticipate any irregular peaks or declines in demand and hence lag in performance and waste resources. When applied in a modern multi-cloud and containerized system, the above limitations directly impact the increased costs of operation and worsened the Quality of Service (QoS).

Predictive scaling is the one that brings in the idea of intelligence, since it predicts the future workload behavior Feng and Ding (2025). Predictive strategies forecast resource need with the help of statistics learning, regression models, and time-series forecasting Kashpruk et al. (2023). Nonlinear workload patterns can be represented in machine learning models, whereas deep learning models such as LSTM and BiLSTM can be used to examine sequential dependencies over time Kaim et al. (2023). Predictive systems are proactive in their allocation, learning the temporal associate of historical metric, to enhance the elasticity, decrease the latency, and sustain the performance even with fluctuating workloads. This is a shift to reactive elasticity to cognitive scalability, and scaling decisions are made on the basis of data, and constant improvement on the model.

2.3 Machine Learning for Cloud Autoscaling Optimization

One of the applications of cloud computing has been the change in the deployment of applications where management of the resources being deployed is of critical concern in order to ensure the optimal utilization of the resources, the compliance of SLA and the low cost of operation. Recent studies show the use of machine learning (ML) and reinforcement learning (RL) in the implementation of proactive and adaptive autoscaling policies. A predictive, application-agnostic autoscaling framework was proposed by Reshad and Jammal (2024) as a time series-based deep learning framework to predict resource utilization to optimize performance and reduce SLA violations, but with corresponding computational overhead costs. A reinforcement-based learning method, initially introduced by Mishra et al. (2024), dynamically adapts resources, circumventing the shortcomings of statistic mechanisms such as Kubernetes Horizontal Pod Autoscaler and is able to effectively deal with sudden workload bursts, although it needs a significant amount of training data and model pruning. Another study given by Joshi et al. (2025) has offered a comprehensive review of RL methods, e.g., deep reinforcement learning and policy gradients, explicit consideration of key issues in state representation, incentives, and scaling to heterogeneous cloud systems, and has shown its application in web services, container orchestration, and data analytics. In addition to these works, Ponnusamy and Khoje (2024)concentrated on predictive resource scaling with the help of ML on AWS and lowered the cost of clouds and over-provisioning by using past data and automated scaling choices, but it had issues in interpretability and compatibility with the current infrastructures. Together, these papers indicate that it is promising that ML-induced autoscaling can be used to improve resource usage, cost-effectiveness, and performance in dynamic clouds, and there is a need to enhance the transparency and generalization of models as shown in Table 1.

2.4 Deep Learning for Temporal Workload Forecasting

Accurate temporal workload and resource prediction is crucial across cloud computing, energy systems, and district heating networks to enable proactive, dynamic, and self-adaptive resource management, ensure service-level compliance, and improve operational efficiency. Recent studies have demonstrated the effectiveness of advanced deep learning techniques in capturing complex, nonlinear, and sequential dependencies in time series data. In Sabyasachi et al. (2024), a hybrid DCNN-LSTM model was proposed for cloud environments to anticipate CPU usage, energy consumption, and SLA violations, achieving improvements in the composite Energy SLA Violation metric over ARIMA-LSTM, CNN, LSTM, and ARIMA by 6.8%, 10.88%, 16.6%, and 22.4%, respectively. Similarly Yuan et al. (2024) proposed a VSBG framework to data centers in the clouds, which incorporates Variational Mode Decomposition, Savitzky-Golay filtering, BiLSTM and Grid LSTM to deal with nonstationary and noisy patterns of workloads, which achieves greater prediction accuracy and faster convergence than other existing algorithms. In energy demand forecasting, Pavlatos et al. (2023) used the LSTM networks bidirectionally to model both past and future temporal relationships in electrical load data and outperformed RNN, LSTM and GRU with MAE of 0.122 and RMSE of 0.022 which is a 46 percent improvement. Next, applying this method to district heating, Song et al. (2024) used BiLSTM and Temporal Convolutional Networks to fill the misleading data and yield high-level features of time, which reduced the error of mean absolute percentage to 2.47% and outperformed traditional LSTM, TCN, and SVR models. Taken together, these experiments prove that hybrid and bidirectional deep learning architectures can be considerably more effective at temporal workload forecasting, but still a number of issues exist as far as computational overhead and the reliance on high-quality data are concerned as indicated in Table 1.

Table 1: Summary Table

Study (Year)	Techniques / Models Used	Application / Focus Area	Key Findings / Results	Limitations / Challenges
Reshad and Jammal (2024)	Time series–based Deep Learning framework	Predictive, application-agnostic autoscaling in cloud systems	Improved system performance and reduced SLA violations	Computational overhead and high resource consumption
Mishra et al. (2024)	Reinforcement Learning (RL)–based adaptive scaling	Dynamic adaptation of cloud resources; comparison with Kubernetes HPA	Effectively handled sudden workload bursts	Requires large training datasets and model pruning
Joshi et al. (2025)	Deep Reinforcement Learning (Policy Gradient, Actor–Critic)	Cloud web services, container orchestration, and data analytics	Provided a unified RL-based autoscaling review emphasizing state representation and incentives	High complexity in heterogeneous cloud environments and training instability
Ponnusamy and Khoje (2024)	Machine Learning–based predictive resource scaling using AWS	Cloud cost optimization and overprovisioning minimization	Reduced cloud costs using historical workload data	Limited interpretability and compatibility with legacy systems
Sabyasachi et al. (2024)	Hybrid DCNN–LSTM model	Cloud CPU utilization, energy consumption, and SLA violation prediction	Outperformed ARIMA–LSTM, CNN, and LSTM with up to 22.4% improvement	Increased computational demand
Yuan et al. (2024)	VSBG framework (VMD + Savitzky–Golay + BiLSTM + Grid LSTM)	Cloud data center workload forecasting	Achieved higher prediction accuracy and faster convergence	Sensitive to noise and high data preprocessing cost
Pavlatos et al. (2023)	Bidirectional LSTM (BiLSTM)	Electrical energy demand forecasting	Outperformed RNN, LSTM, and GRU with MAE = 0.122 and RMSE = 0.022 (≈46% improvement)	Requires large-scale temporal datasets
Song et al. (2024)	Hybrid BiLSTM + Temporal Convolutional Network (TCN)	District heating load time-series forecasting	Reduced MAPE to 2.47%, outperforming LSTM, TCN, and SVR	Strong dependence on high-quality and complete data

2.5 Research Gaps Identified

Despite the existing literature that shows the performance of DL in cloud workload forecasting and autoscaling, there are still a number of gaps. Study given by (Dang-Quang and Yoo, 2022) determined that multivariate Bi-LSTM is better at autoscaling decisions than LSTM and CNN-LSTM, but it lacks attention to identify and focus on important temporal relationships. Also, they concentrate on data-level accuracy of autoscaling without multi-step short-term prediction or real cloud implementation checks. Current research is also focused on the accuracy of the prediction, and not on the runtime performance and cloud feasibility. Thus, the gap in research regarding the evaluation of attention-enhanced bidirectional models can be identified through their practical validation based on real Azure traces with practical AWS-based validation.

3 Methodology

In this chapter, the dataset of the Microsoft Azure VM and Azure Functions is described as the time-series workload analysis with the five-minute intervals. It describes the process of cleaning, preprocessing, feature engineering, Min -Max normalization, and the rolling window process with 12 successive values. Such steps are used to convert raw cloud traces into a supervised learning format, which can be used in deep learning models.

3.1 Dataset Description

Microsoft Azure Traces is an open dataset[1]found on GitHub that is a rich source of cloud computing, workload forecasting, and resource optimization research. It offers massive time-series traces, which reflect the actual operation behavior of the Microsoft Azure cloud infrastructure. The data set contains two major classes that are Virtual machine (VM) Traces and Azure functions Traces. The VM traces contain useful workload information of 2017 and 2019 as well as an extra VM request trace which is especially valuable in the analysis and optimization of resource packing algorithms. These traces are important in documenting the critical performance parameters like: CPU usage, memory usage, resource allocation trends on different virtual machines. The Azure Functions Traces, in their turn, include invocation data of serverless functions collected within a period of two weeks in 2019, including the data on accessing blobs collected in November and December 2020. These traces can be used to know how serverless environments will behave with variable workloads. The dataset is the most suitable one in terms of time-series analysis, demand forecasting, and autoscaling studies since it has high-frequency measurements (five minutes in between). Using this dataset, scientists are capable of creating predictive models that can be used to manage resources better, increase efficiency, and make intelligent decisions in dynamic clouds.

3.2 Data Cleaning, Preprocessing and Feature Engineering

Preprocessing and data cleaning are important factors that can be used to prepare the dataset to be used in training and analyzing the model Bala and Behal (2024). The raw traces of Microsoft Azure are first analyzed to identify and manage the absence of values or null values which maintains consistency and reliability of data. The process of reducing

[1]https://github.com/Azure/AzurePublicDataset

unwanted or unneeded type of columns that do not play a role in the performance of the model is performed to maximize the computational performance . This process of cleaning is to make sure that only pertinent features affecting the behavior of the workload are maintained. After that, preprocessing and feature engineering are carried out to increase the scope of analysis of the dataset. The timestamp columns are also turned into proper forms of datetimes in order to be able to perform proper time analysis and match events. Based on these datetime fields, other features like month, day, hour, and year are derived and it is possible to identify the pattern of workload seasonally or periodically.

3.3 Scaling the Data Using Min-Max Normalization

Scaling is an important preprocessing mode which attains consistency Dhawas et al. (2024) across features in datasets to enhance the accuracy and stability of machine learning and deep learning systems. Min-Max normalization is used in this work to normalize all numerical variables in a range of 0 to 1. This change reduces the effect of features whose magnitude is greater and the convergence of the model is quicker to train. Using this method of normalization, the contribution of every input variable to the learning process is proportionate, which lessens the bias of computation. The scaled data thereby gets more uniform and better suited to algorithms that are sensitive to data scale like LTM and BiLSTM networks.

3.4 Window Rolling with 12 Previous Values

A window rolling mechanism is implemented to attain temporal dependencies and time-related trends in the time-series data with the use of the last 12 values as historical input. This strategy allows the model to be learned based on short term and long-term workload pattern dependencies. Every rolling window is a predetermined length of historical observation and is utilized as the predictor of the next time step, which is a transformation of the data to supervised learning format in essence. This is a technique that improves predictability of deep learning models to discover periodic variations and temporal associations that can help in making reliable forecasts of cloud resources.

3.5 System WorkFlow

Figure 2: WorkFlow Diagram

The workflow begins with Microsoft Azure VM CPU time-series data which is followed by preprocessing and feature extraction to clean and structure inputs as shown in Figure 2. Data is normalized and converted into sliding windows for temporal learning. DL models are trained and evaluated by enabling multi-step CPU forecasting and final deployment on AWS for intelligent resource allocation.

4 Design Specification

In this chapter, the system architecture of the offered cloud resource optimization framework is provided, the data preprocessing, scaling, window rolling, model training, and evaluation processes are described. It describes the block and architectural schemes depicting the end-to-end flow of the work process of Azure data ingestion to AWS deployment. The chapter also identifies the BiLSTM-CATF algorithm of forecasting CPU use, the structure of the model, and the forecasting process.

4.1 System Components and Architecture

The proposed architecture shows the end-to-end workflow of the DeepLearning–based CPU usage forecasting framework for intelligent cloud resource allocation. It uses real-world cloud workload data, advanced preprocessing techniques, multiple DeepLearning models and practical deployment of the cloud components. Figure 3 shows the system architecture of proposed cloud resource optimization framework based on the deep learning models. It starts with the Microsoft Azure Dataset that has time-series data about workloads of virtual machines.

Data Preprocessing stage is used to clean and format the raw data, and then Feature Engineering is performed, which recognizes the temporal features like month, day, and hour. Data Scaling step involves normalizing the values using Min-Max to make all of the values within the same range, which improves the convergence of the model. After that, Window Rolling which has 12 past values is used to extract temporal dependencies and send the data to training phase. Model Evaluation measures their efficiency in terms of RMSE, R^2 and all.

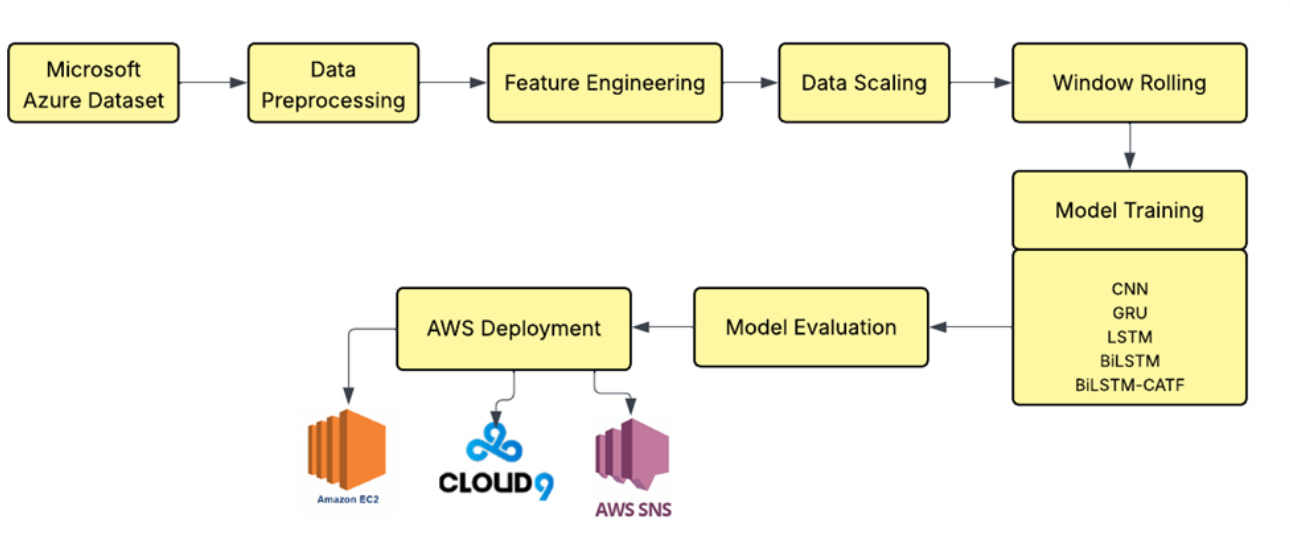

Figure 3: System Architecture Diagram

4.2 Algorithm: BiLSTM-CATF for CPU Usage Forecasting

Input:
Normalized time-series CPU usage data X, window size w=12
Output:
Predicted CPU usage y

Algorithm 1 BiLSTM with Cross-Attention for CPU Usage Forecasting

Input: CPU usage time-series data X, window size w
Output: Predicted CPU usage $\hat{y}$

Load CPU usage data X
if missing values exist in X **then**
    Clean data using forward-fill interpolation
end if
Normalize X using Min–Max scaling
for $i = w$ to length of X **do**
    Generate input sequences and corresponding labels
end for
Split dataset into training and testing sets (90% / 10%)
Feed input sequences into the model
Apply BiLSTM layer with 128 units and return_sequences = True
Apply Dropout with rate 0.5
Apply BiLSTM layer with 64 units and return_sequences = True
Apply Dropout with rate 0.5
Apply Cross-Attention Tensor Fusion on BiLSTM outputs
Flatten the attention output
Apply Dense layer with one neuron to generate prediction
Train the model using Adam optimizer and Mean Squared Error (MSE) loss
if forecasting is required **then**
    for $t = 1$ to 15 **do**
        Predict next CPU usage value recursively
        Append prediction to input sequence
    end for
end if
return Predicted CPU usage $\hat{y}$

Explanation: The BiLSTM-CATF algorithm forecasts short-term CPU usage by learning temporal patterns from normalised type of cloud workload time-series data with the help of a sliding window of 12 previous observations. Stacked BiLSTM layers capture both past and future dependencies, whereas dropout regularisation improves model generalisation. A Cross-Attention Tensor Fusion mechanism does show the most relevant temporal features to increase the accuracy of the prediction. The final output layer produces precise CPU usage forecasts by enabling proactive cloud resource allocation.

5 Implementation

This chapter showed the full deployment of the suggested deep learning-based CPU usage forecasting model with referencing to real-life time-series data of Azure VM. Several deep learning models were trained, and deployed using Python and cloud computing. The practicality and scalability of the proposed solution in the actual cloud environments were proven by the integration of attention mechanisms and AWS services.

5.1 Tools and Technologies

Table 2 shows the tools and techniques used throughout the implementation phase of the study. It shows the use of Python-based DL frameworks with cloud infrastructure to support accurate CPU usage forecasting and performance evaluation.

5.2 Implementation of DL Models

5.2.1 CNN Model

The CNN model was applied based on 1D convolutional neural network used to run time series data of CPU usage on the CNN model Ahmadzadeh et al. (2025). The model has three convolutional blocks with max-pooling in the middle in order to gradually decrease the number of features. After the initial convolution layer, batch normalization was used to stabilize the learning process, and ReLU activation was used to guarantee the extraction of non-linear features. Flattened, fully-connected dense layers with dropout were then added to stop overfitting and the final prediction is a single neuron to keep continuous prediction. The model has been assembled with Adam optimizer based on mean squared error loss and R^2 score measure. The training process included the use of 12 sequence of past time steps, data scaling with min-max normalization, and validation performance to reduce the dynamically changing learning rate. A performance assessment was done based on MSE, RMSE, MAE, R^2 score and comparison between predicted and actual trends of CPU usage. Such a structure enables the CNN to get local temporal features effectively.

5.2.2 GRU Model

GRU-based model is utilized to sequence learn CPU usage across time using gated recurrent units Niu et al. (2023). It has four GRU layers, the first three of them reuse sequences so that deeper temporal features could be extracted. Unable to memorize over sequential dependencies, dropout layers with probability 70 are implemented between each GRU layer. The last GRU layer only provides the final time step, which is then inputted into a dense layer to have the regression prediction. The model is assembled on the Adam optimizer, mean squared error loss, and R^2 metric. The input sequences are made ready with the help of 12 time-step windows and features that are normalized. Early termination during training had the effect of recovering optimal epoch weights, which maximized model generalization. The gating mechanism of GRU is effective to gather long-term relationships in time series data and the analysis is done using the standard metrics like loss, R^2 score, and graphical comparison of the real and predicted CPU values and the performance is evaluated to have strong prediction ability of the workloads.

Table 2: Implementation Tools, Techniques, and Their Purpose

Category	Tool / Technique	Purpose in Implementation
Programming Language	Python ($\geq$ 3.7.9)	Core language for data processing, modeling, and evaluation
Development Environment	Google Colab	Model development, training, and experimentation
Cloud IDE	AWS Cloud9	Cloud-based testing and deployment validation
Dataset	Microsoft Azure VM Traces	Real-world CPU usage time-series data
Data Handling	Pandas, NumPy	Data loading, cleaning, and preprocessing
Data Visualization	Matplotlib, Seaborn	Time-series plotting and performance visualization
Data Preprocessing	Min–Max Normalization	Scaling data between 0 and 1 for improved model stability
Feature Engineering	Sliding Window (12 steps)	Capturing temporal dependencies in CPU usage patterns
Deep Learning Framework	TensorFlow, Keras	Building and training deep learning models
Models Implemented	CNN, GRU, LSTM, BiLSTM, BiLSTM–CATF	CPU usage prediction and comparative analysis
Attention Mechanism	Cross-Attention Tensor Fusion (CATF)	Enhancing temporal feature importance and representation
Evaluation Metrics	MSE, RMSE, MAE, R^2	Measuring prediction accuracy and model performance
Performance Analysis	Latency, Throughput, Execution Time, Memory Usage	Assessing computational efficiency and scalability
Cloud Infrastructure	AWS EC2	Model execution and scalability testing
Monitoring	AWS SNS	Performance notifications and system alerts

5.2.3 LSTM Model

LSTM was created to perform sequential learning on the long short-term memory layers. A stack of three LSTM nets are used to learn the temporal dependencies in the CPU usage data so that the first two layers can provide Ahmadzadeh et al. (2025) a sequence back so that hierarchical features can be learned. Each LSTM layer is followed by dropout layers with probability as 80 to decrease overfitting due to recurrent connections. The LSTM layer has the hidden-output of a dense layer that has one neuron to make a continuous regression prediction. Compiling is done using Adam optimizer using a mean squared

error loss and the R^2 score metric. The training was done using input windows of 12 past values, which have been scaled using the min-max normalization, and early stopping was used to restart the optimal weights. The memory cells in the LSTM layers enable memory of the long-term time series dependencies. Measures of evaluation are loss, R^2 score, actual/ predicted graphs indicating the capacity to explain both short term and long term trends in CPU usage to successfully predict workload.

5.2.4 BiLSTM Model

The BiLSTM model is a combination of stacked layers which builds upon temporal dependencies of both forward and backward context Sirisha et al. (2023). The first LSTM layer is used to generate sequences and the second, third and fourth layers of the LSTM are used to expand the feature information. Probability of dropout layers 80 percent are used after every LSTM layer to avoid overfitting. The final dense layers consist of a 32 unit tanh-activated layer and a linear single-unit output to regression prediction. The model is summarized using Adam optimizer, mean squared error loss, and R^2 measure. The input data have 12-step sequences that have been normalized through min-max scaling. In training, early termination provides recovery of optimum epoch weights. The Bidirectional architecture permits the model to acquire both forward and backward temporal dependencies concurrently improving predictive accuracy. Evaluation of the models makes use of MSE, RMSE, MAE, R^2 score and graphical comparisons of the actual and predicted trends of the CPU usage.

5.2.5 BiLSTM-CATF Model

BiLSTM-CATF model is an extension of BiLSTM architecture that incorporates a cross-attention mechanism based on a tensor fusion. Two forward and backward time dependencies are learned using LSTMs, which are followed by 50 percent dropout regularization. The cross-attention layer gets attention weights by time steps and features by using Permute, Dense, Lambda, andMultiply layers which enable the model to concentrate on meaningful temporal patterns. The resulting attention output is flattened and fed into a single-neuron dense layer in order to get a regression prediction. This model is assembled using Adam optimizer, mean squared error loss, and R^2 score measure. The min-max scaling is used to normalize the input sequences of 12 time steps. Early termination replenishes the optimal weights during training. The cross-attention offers interpretability and performance of the model, as it finds the most important dependencies and the performance is evaluated by standard metrics and visual plots of predicted and real CPU usage to compare performance in future workloads prediction.

5.3 Multi-Step CPU Usage Forecasting (Next 15 Intervals)

This section implements recursive multi-step time series forecasting to predict CPU usage for the next 15 time steps (5-minute intervals) as shown in Figure 4. Scaled values are reshaped to match the model's input format and iteratively fed into the trained deep learning model. At each step the model has been predicted the next CPU value, which is appended to the sliding window and reused as input for subsequent predictions.

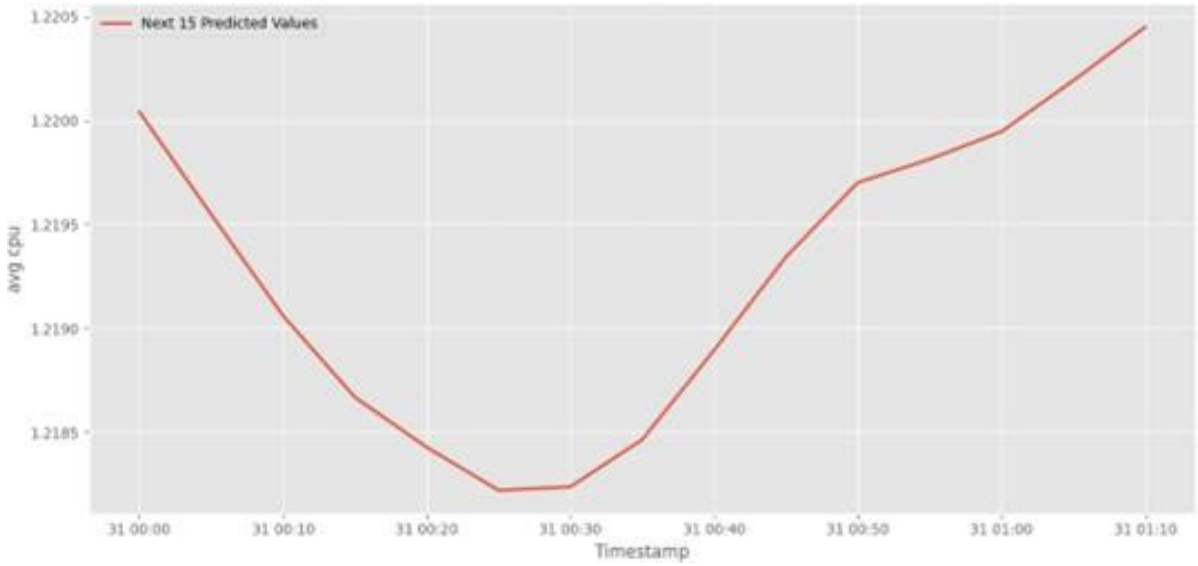

Figure 4: Next 15 step CPU usage forecast

5.4 Implementation of AWS

EC2 Instance: AWS EC2 instance serves as the backbone of our cloud infrastructure. This instance runs the virtual machine that handles data processing and model training. It has been configured with a public IP for easy access and uses AWS Compute Optimizer to have optimal performance. This setup is important for scaling the deep learning workloads and maintaining performance during resource allocation.

Cloud9 Environment: AWS Cloud9 IDE shows the development and coding happen. This cloud-based IDE allows smooth use with the EC2 instance by enabling direct code execution and debugging. It smooths the process of developing and testing the deep learning models which secures that it is both powerful and easy to manage.

SNS (Simple Notification Service): SNS setup has used to send email notifications about any type of performance for the models and their updates. This secures that any important events or results are communicated promptly by helping maintain transparency and timely decision-making.

6 Evaluation

In this chapter the standard regression and system performance measures is used to measure the predictive accuracy and the computational performance of all models implemented. The experiment results showed that the proposed BiLSTM-CATF model was always better than the baseline models in terms of accuracy and still had a reasonable latency and throughput. The performance and practical value of the suggested method of optimization of cloud resources were further confirmed by comparative analysis with a baseline study.

6.1 Evaluation Metrics

6.1.1 Mean Squared Error (MSE)

Mean Squared Error (MSE): This measures the average of squared differences between actual and predicted CPU usage values by penalizing larger errors more heavily You et al. (2023).

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{1}$$

6.1.2 Root Mean Squared Error (RMSE)

Root Mean Squared Error (RMSE): This represents the square root of MSE by providing the average prediction error in the same unit as CPU usage Uyeh et al. (2022).

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{2}$$

6.1.3 Mean Absolute Error (MAE)

Mean Absolute Error (MAE): This calculates the average absolute difference between predicted and actual values by giving measure less sensitive to outliers Uyeh et al. (2022).

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{3}$$

6.1.4 Coefficient of Determination (R^2 Score)

R^2 Score (Coefficient of Determination): This metric shows how well the model explains the variance in actual CPU usage with values closer to 1 showing better fit Daraghmeh et al. (2023).

$$R^2 = 1 - \frac{\sum_{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} \tag{4}$$

6.1.5 Latency and Throughput

Latency: This measures the average time taken by the model to generate a single CPU usage prediction Li et al. (2022).
Throughput: This represents the number of predictions processed per second showing the model's real-time inference capability Li et al. (2022).

$$\text{Throughput} = \frac{\text{Number of Predictions}}{\text{Total Inference Time}} \tag{5}$$

6.2 Experiment 1: Model Loss and R^2 graphs

The model loss plot shows that there is effective learning, as the training loss is monotonically decreasing as the CNN is taught to find spatial patterns in the time-windowed CPU utilization data as indicated in Figure 5 and 6. The validation loss is also small

and constant, which proves that the model does not memorize training examples and generalizes to unseen data. In line with this, the graph of the R^2 score depicts a steady improvement with an increase on the training and validation scores towards 1, which has a stronger predictive accuracy. The fact that training and validation curve closely resemble in the two graphs confirm the absence of overfitting. Convolutional filters used by CNN are effective to extract the local temporal patterns and dependencies of the CPU usage sequences and make strong predictions of features and high accuracy predictions.

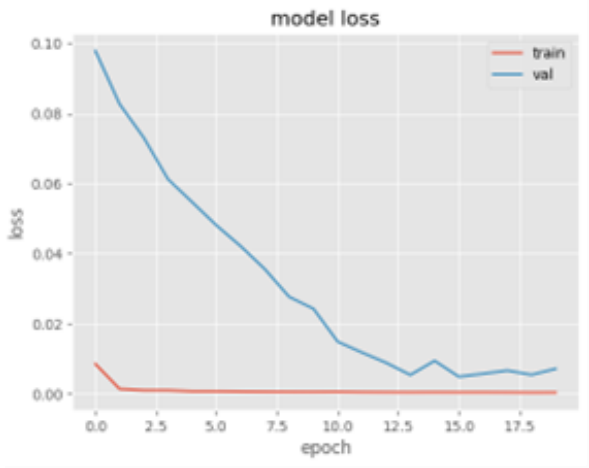

Figure 5: Model Loss for CNN

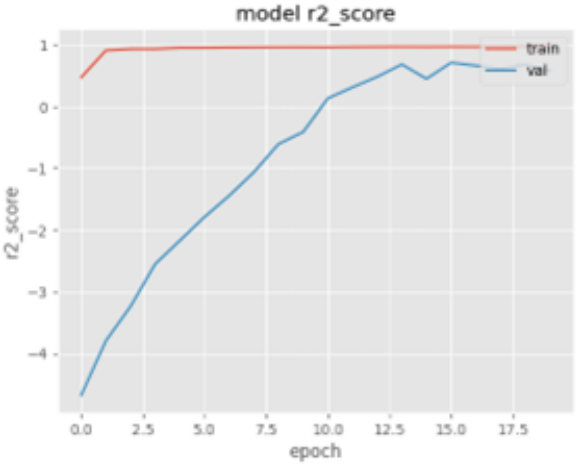

Figure 6: Model R^2 Graph for CNN

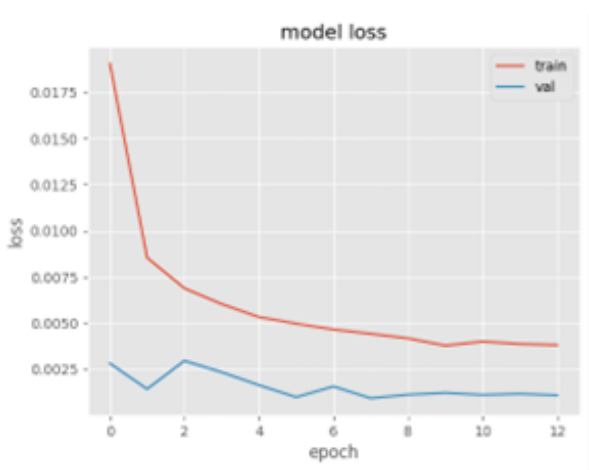

Figure 7: Model Loss for GRU

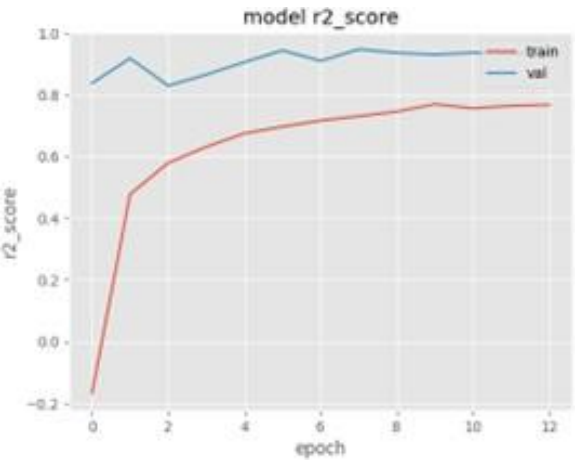

Figure 8: Model R^2 Graph for GRU

The model loss plot displays that the convergence is fast at the beginning with a slow increasing convergence until the gating mechanisms of the GRU achieves accuracy in storing the necessary temporal information and eliminating noise as illustrated in Figure 7 and 8. The loss of validation shows small changes, yet it is consistent, which indicates the capacity of the model to be applied to various times. The graph of R^2 scores shows that there is steady improvement in the predictive power whereby the validation performance is increasing gradually as the recurrent architecture acquires long-term dependencies. A moderate distance between the training and validation measures is a healthy behavior of the models. GRU is also highly effective in time series prediction due to its reset

and update gates effectively capturing the sequential patterns of CPU utilization at a computational cost, which is usually higher than the conventional LSTM architecture.

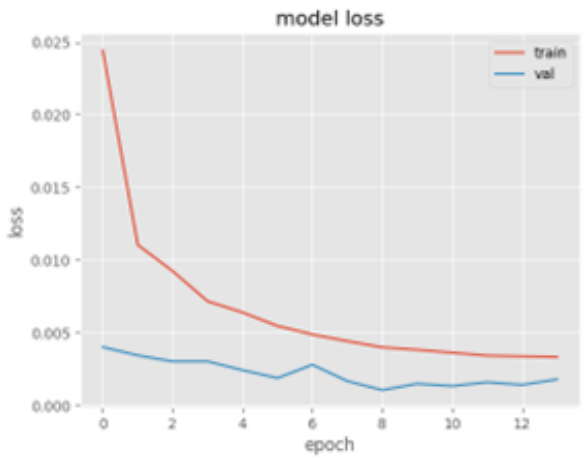

Figure 9: Model Loss for LSTM

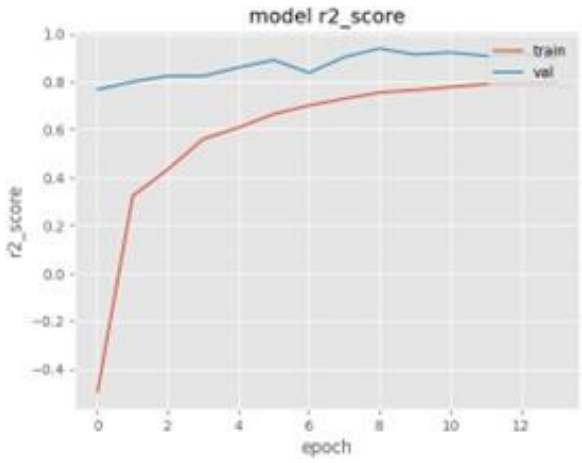

Figure 10: Model R^2 Graph for LSTM

The model loss graph shows good convergence of learning in which the training loss reduces rapidly at the onset of training when the LSTM memory cells are being trained to store important temporal information and it levels off at minimal levels as in Figure 9 and 10. Validation loss is also low and consistent with a few fluctuations that validate strong generalization with no overfitting. There is complementary improvement in the R^2 score graph, where both the training and validation scores are increasing towards the optimal values, which denotes improved predictive accuracy. Balanced model learning is indicated by the close tracking between training and validation curves. The advanced gating framework of LSTM, with an input, forget, and output gate, allows selective information retention in long sequences, which is essentially able to capture intricate time-dependent information and patterns in CPU utilization data, and thus produces superior forecasting accuracy in time series prediction tasks.

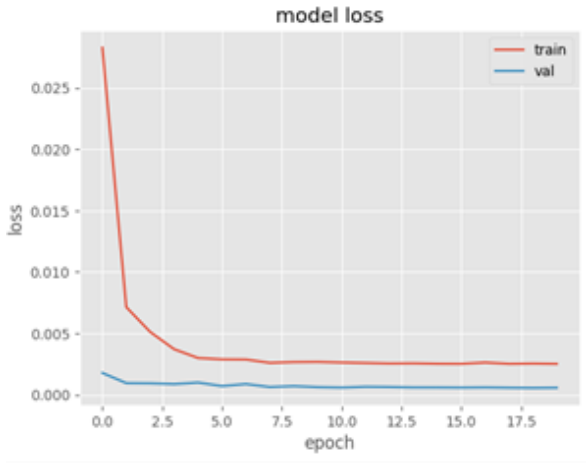

Figure 11: Model Loss for BiLSTM

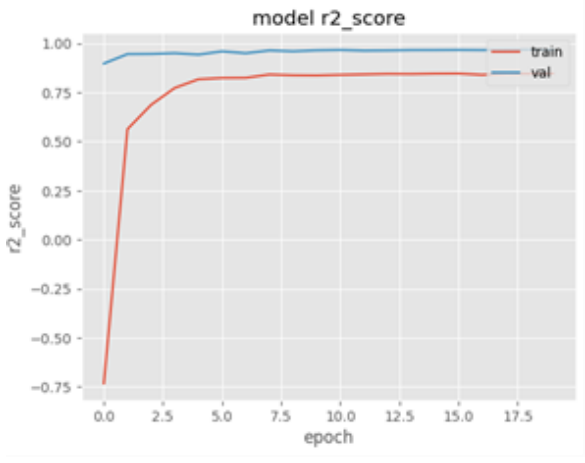

Figure 12: Model R^2 Graph for BiLSTM

The model loss curve converges at a high rate at the first several epochs because the architecture uses both forward and backward processing of the sequence simultaneously,

introducing the opportunity to recognize patterns of the time-span fully as was demonstrated in Figure 11 and 12. Training loss decreases sharply and then it levels while validation loss has low values, indicating high generalization ability. The high divergence between the training and validation means the optimal model complexity. The method is very effective in terms of multivariate time series forecasting because this bidirectional learning process has been proven to be good at complex interdependencies in CPU workload patterns.

The model loss curve shows a remarkably smooth convergence with the Cross-Attention Tensor Fusion mechanism that optimizes the traditional BiLSTM by dynamically adjusting the weights of the relevant temporal features of various time scales as shown in Figure 13 and 14. Both validation loss and training loss go down at a very high rate and approach very close values, which means they are going to feature integrate without overfitting. The R^2 score curve indicates the immediate improvement of the performance, which reaches almost the best predictive ability in a limited amount of epochs and is stable after that. The close parallelism between training and validation curves indicates that there is a strong learning performance of the model. The CATF component adds the mechanisms of attention that selectively focus on the key patterns of CPU utilization and integrate the multi-dimensional temporal representations with the use of tensors. This high architecture adds bidirectional context awareness and cross-attention refinement, providing the state-of-the-art forecasting accuracy in situations with complex prediction needs of cloud resources.

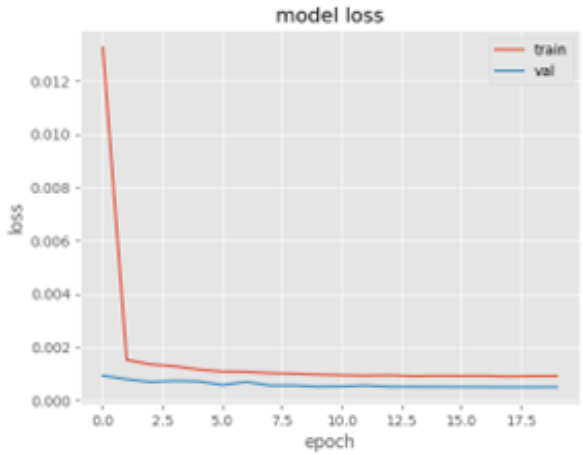

Figure 13: Model Loss for BiLSTM-CATF

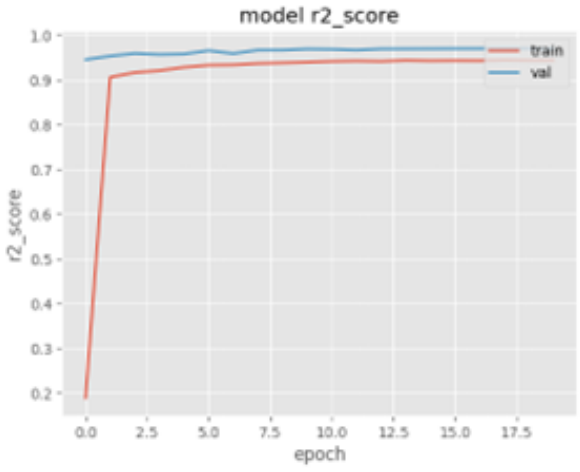

Figure 14: Model R^2 Graph for BiLSTM-CATF

6.3 Experiment 2: Latency and Throughput

In this experiment, the computational efficiency of various deep learning models is assessed in terms of latency and throughput, which is very important parameters of real-time cloud resource management as demonstrated in Table 3. The CNN model has the lowest latency and the highest throughput because it has a simpler architecture and has no recurrent connections, and thus it is computationally lightweight and rapid at inference. GRU and LSTM recurrent models are characterized by more latency since sequential processing and gating mechanisms add to the computational cost, but throughput is balanced

and can be used in near real-time system applications. BiLSTM scales the complexity in addition to processing the input sequence in both temporal directions, which have minor effects on latency but enhance contextual learning. The BiLSTM-CATF model has the largest latency and the lowest throughput, which can be mainly explained by the fact that the model has the additional cross-attention tensor fusion operations that provide an extra computation during prediction. Such a trade-off is however compensated by its superior predictive accuracy seen in previous experiments. In general, CNN is the most efficient when the inference is required to be performed at high speed and the BiLSTM-CATF is the most effective in the case when the priority is given to prediction accuracy rather than the speed of calculations in optimization of cloud scalability.

Table 3: Latency and Throughput Comparison of DL Models

Model	Latency per Prediction (ms)	Throughput (predictions/sec)
CNN	66.6659	7845.43
GRU	76.5907	5267.38
LSTM	68.4571	5474.52
BiLSTM	70.2410	5489.61
BiLSTM-CATF	70.8621	2468.72

6.4 Experiment 3: Performance Evaluation Metrics

The experiment comparatively assesses the performance of various deep learning architectures on predicting-CPU-usage on a full performance as presented in Table 4. CNN model presents a somewhat larger value of errors showing that it is a weak model in long term temporal dependencies that are evident in cloud workload time series data. GRU and LSTM are examples of recurrent models that can achieve significant improvements as their gated mechanisms allow learning the consecutive patterns and the workload variations. BiLSTM model also increases prediction accuracy by working in both forward and backward directions which allows it to capture more contextual information. The suggested BiLSTM-CATF model is superior to all other models, not only because of the quantifiable values of numerical error, but because of the cross-attention mechanism of the filtering of the most relevant temporal features through the use of the cross-attention tensor fusion. The fusion is based on attention which mitigates information loss, feature interaction and well as generalization ability. Consequently, BiLSTM-CATF provides more consistent and accurate predictions and is the best model to use to allocate cloud resources efficiently and optimize scalability.

Table 4: Performance Comparison of Deep Learning Models on CPU Usage Prediction

Model	MSE	RMSE	MAE	R^2-Score
CNN	0.004945	0.070326	0.058384	0.713227
GRU	0.000913	0.030216	0.023669	0.947059
LSTM	0.001052	0.032430	0.025568	0.939016
BiLSTM	0.000549	0.023427	0.018911	0.968178
BiLSTM-CATF	0.000501	0.022382	0.017975	0.970953

6.5 Comparative Discussion with Baseline

The baseline paper given by (Dang-Quang and Yoo, 2022) presents the evaluation of an effective multivariate autoscaling system with Bi-LSTM on real cloud workload datasets namely GWA-T-12 and GWA-T-13. The study compares this univariate Bi-LSTM and multivariate Bi-LSTM as well as continues to contrast this multivariate Bi-LSTM with multivariate LSTM and CNN-LSTM. The experimental findings presented in Table 5 indicate that the multivariate Bi-LSTM performs better in achieving the lowest MAE, MSE and RMSE values on both datasets, thus proving itself to be better at inter-feature dependencies in forecasting workloads and autoscaling the dataset. Although the Dang-Quang and Yoo (2022) concentrates on the concept of multivariate autoscaling performance within the framework of MAPE loop architecture, the proposed work builds upon this premise by utilizing the Azure VM time-series CPU utilization databases with 5 minutes intervals. Along with the assessment of deep learning models, the proposed study focuses on the multi-step forecasting and the validation of the models on AWS EC2 and Cloud9, which increases the practicality of the work in the optimization of performance and scalability.

Table 5: Comparison Table

Aspect	**Baseline Paper (Dang-Quang and Yoo, 2022)**	**Proposed Study**
Time-Series Type	Multivariate workload prediction	Time-series CPU usage prediction
Core Model Comparative Models	Bi-LSTM Univariate Bi-LSTM, Multivariate LSTM, CNN-LSTM	BiLSTM and BiLSTM-CATF CNN, GRU, LSTM, BiLSTM
Performance Metrics Key Result	MAE, MSE, RMSE Multivariate Bi-LSTM achieves lowest RMSE	MAE, MSE, RMSE, R^2 BiLSTM-based models outperform others
Focus Area	Autoscaling efficiency	Prediction accuracy and cloud performance
Forecasting Scope	Dataset-level prediction	Multi-step future forecasting
Deployment Validation	Not done	AWS EC2 and Cloud9 testing

7 Conclusion and Future Work

7.1 Conclusion

This study addressed and responded to the following research question of how using a cross-attention mechanism within a BiLSTM architecture affects short-term CPU usage forecasting accuracy and computational performance in cloud environments Based on Microsoft Azure VM time-series dataset on a real-world setting, various DL models were applied and tested. The obtained experimental results prove the proposed BiLSTM-CATF model to be the most successful with the lowest RMSE (0.0223) and the highest R2 (0.9709) measured among all models. The cross-attention integration helped the model to consider the most important temporal dependencies to promote prediction accuracy.

These findings endorse that a bidirectional modeling that is enhanced by attention is an effective way to support proactive and intelligent allocation of cloud resources.

7.2 Limitations of the Study

The experiments were carried out on a subset of Microsoft Azure VM traces that might not be fully representative of real-world cloud workloads that are highly heterogeneous and/or large scale. The test was conducted in a controlled AWS setting with limited possibilities to record actual network instability, workloads spikes and multi-cloud interoperability issues. Also, the BiLSTM-CATF model puts an increased computational burden and inference delay because of the attention mechanism. The study was mainly based on the prediction of CPU usage, with other essential measures of resources like memory usage, energy consumption, and cost-sensitive optimization being omitted, which may play a role in determining holistic cloud resource management.

7.3 Future Works

The further study will be devoted to the expansion of the proposed framework to include other cloud resource metrics to include memory usage, energy efficiency, and cost prediction to realize the holistic resource optimization. The light and efficient attention mechanism can be investigated to minimize the inference latency at the cost of the high prediction accuracy. Moreover, a combination of decision-making that is based on reinforcement learning and the use of predictive forecasting models would allow allocating cloud resources fully autonomously and adaptively. The framework may also be tested on the real-time production cloud environments and generalized to the hybrid and multi-cloud platforms to measure the scalability, robustness and generalization under various workload conditions. Future work can explore the use of advanced time-series forecasting architectures such as Transformer-based models and hybrid approaches to capture long-range temporal type of dependencies than recurrent networks in a good manner.

References

Agrawal, S. and Singh, D. (2024). Study containerization technologies like docker and kubernetes and their role in modern cloud deployments, *2024 IEEE 9th International Conference for Convergence in Technology (I2CT)*, IEEE, Pune, India, pp. 1–5.

Ahmadzadeh, M., Zahrai, S. M. and Bitaraf, M. (2025). An integrated deep neural network model combining 1d cnn and lstm for structural health monitoring utilizing multisensor time-series data, *Structural Health Monitoring* **24**: 447–465.

Akoh Atadoga, U. J., Umoga, U. J., Lottu, O. A. and Sodiya, E. O. (2024). Evaluating the impact of cloud computing on accounting firms: A review of efficiency, scalability, and data security, *Global Journal of Engineering and Technology Advances* **18**: 065–075.

Ali, N. N. and Zeebaree, S. R. M. (2024). Distributed resource management in cloud computing: A review of allocation, scheduling, and provisioning techniques, *Indonesian Journal of Computer Science* **13**.

Bala, B. and Behal, S. (2024). A brief survey of data preprocessing in machine learning and deep learning techniques, *2024 8th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)*, IEEE, Kirtipur, Nepal, pp. 1755–1762.

Betti Pillippuge, T. L., Khan, Z. and Munir, K. (2025). Horizontal autoscaling of virtual machines in hybrid cloud infrastructures: Current status, challenges, and opportunities, *Encyclopedia* **5**: 37.

Dang-Quang, N.-M. and Yoo, M. (2022). An efficient multivariate autoscaling framework using bi-lstm for cloud computing, *Applied Sciences* **12**: 3523.

Daraghmeh, M., Agarwal, A. and Jararweh, Y. (2023). A multilevel learning model for predicting cpu utilization in cloud data centers, *2023 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, IEEE, pp. 1016–1023.

Demirbaga, Ü ., Aujla, G. S., Jindal, A. and Kalyon, O. (2024). Cloud computing for big data analytics, *Big Data Analytics*, Springer Nature Switzerland, Cham, pp. 43–77.

Dhawas, P., Dhore, A., Bhagat, D., Pawar, R. D., Kukade, A. and Kalbande, K. (2024). Big data preprocessing, techniques, integration, transformation, normalisation, cleaning, discretization, and binning, *in* D. Darwish (ed.), *Advances in Business Information Systems and Analytics*, IGI Global, pp. 159–182.

Feng, B. and Ding, Z. (2025). Application-oriented cloud workload prediction: A survey and new perspectives, *Tsinghua Science and Technology* **30**: 34–54.

Hamzah Khan, M., Habaebi, M. H. and Rafiqul Islam, M. (2024). A systematic literature review of cloud brokers for autonomic service distribution, *IEEE Access* **12**: 131164–131187.

Joshi, V., Patel, P., Chandwani, N., Bhatia, J. and Kumhar, M. (2025). Intelligent auto-scaling in cloud infrastructure using machine learning and reinforcement learning, *Advances in Data-Driven Computing and Intelligent Systems*, Lecture Notes in Networks and Systems, Springer Nature Singapore, Singapore, pp. 217–239.

Kaim, A., Singh, S. and Patel, Y. S. (2023). Ensemble cnn attention-based bilstm deep learning architecture for multivariate cloud workload prediction, *Proceedings of the 24th International Conference on Distributed Computing and Networking*, ACM, Kharagpur, India, pp. 342–348.

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**: 12374.

Li, P., Wang, X., Huang, K., Huang, Y., Li, S. and Iqbal, M. (2022). Multi-model running latency optimization in an edge computing paradigm, *Sensors* **22**(16): 6097.

Mishra, R., Kshetri, N. and Kumar Jha, S. (2024). Leveraging blockchain technology for making secure iot networks, *Blockchain Technology for Cyber Defense, Cybersecurity, and Countermeasures*, CRC Press, Boca Raton, pp. 17–33.

Niu, Z., Zhong, G., Yue, G., Wang, L.-N., Yu, H., Ling, X. and Dong, J. (2023). Recurrent attention unit: A new gated recurrent unit for long-term memory of important parts in sequential data, *Neurocomputing* **517**: 1–9.

Pavlatos, C., Makris, E., Fotis, G., Vita, V. and Mladenov, V. (2023). Enhancing electrical load prediction using a bidirectional lstm neural network, *Electronics* **12**: 4652.

Ponnusamy, S. and Khoje, M. (2024). Optimizing cloud costs with machine learning: Predictive resource scaling strategies, *2024 5th International Conference on Innovative Trends in Information Technology (ICITIIT)*, IEEE, Kottayam, India, pp. 1–8.

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**: 619.

Reshad, A. and Jammal, M. (2024). Optimizing cloud and iot resource utilization: A proactive, application-agnostic auto-scaling technique using machine learning, *2024 IEEE/ACM 17th International Conference on Utility and Cloud Computing (UCC)*, IEEE, Sharjah, United Arab Emirates, pp. 489–496.

Sabyasachi, A. S., Sahoo, B. M. and Ranganath, A. (2024). Deep cnn and lstm approaches for efficient workload prediction in cloud environment, *Procedia Computer Science* **235**: 2651–2661.

Shi, F. and Lin, J. (2022). Virtual machine resource allocation optimization in cloud computing based on multiobjective genetic algorithm, *Computational Intelligence and Neuroscience* **2022**: 1–10.

Sirisha, B., Goud, K. K. C. and Rohit, B. T. V. S. (2023). A deep stacked bidirectional lstm (sbilstm) model for petroleum production forecasting, *Procedia Computer Science* **218**: 2767–2775.

Song, J., Li, W., Zhu, S., Zhou, C., Xue, G. and Wu, X. (2024). Predicting hourly heating load in district heating system based on the hybrid bi-directional long short-term memory and temporal convolutional network model, *Journal of Cleaner Production* **463**: 142769.

Uyeh, D. D., Iyiola, O., Mallipeddi, R., Asem-Hiablie, S., Amaizu, M., Ha, Y. and Park, T. (2022). Grid search for lowest root mean squared error in predicting optimal sensor location in protected cultivation systems, *Frontiers in Plant Science* **13**: 920284.

You, D., Lin, W., Shi, F., Li, J., Qi, D. and Fong, S. (2023). A novel approach for cpu load prediction of cloud server combining denoising and error correction, *Computing* **105**(3): 577–594.

Yuan, H., Bi, J., Li, S., Zhang, J. and Zhou, M. (2024). An improved lstm-based prediction approach for resources and workload in large-scale data centers, *IEEE Internet of Things Journal* **11**: 22816–22829.