International Journal of Engineering Trends and Technology
ISSN: 2231-5381/ https://doi.org/10.14445/22315381/IJETT-V74I8P114

Volume 74 Issue 8, 210-221, August 2026

Research Article

Cost-Efficient Predictive Auto-Scaling Using Transformer-LSTM Fusion Tuned with Bayesian Optimization

Krishnamoorthy V[1], Akshaya V[2], Sivanantham S[3], Ganagavalli K[4]

[1]*Department of Computer Science and Engineering, Rajalakshmi Engineering College, Tamil Nadu, India.*
[2]*Department of Artificial Intelligence and Machine Learning, Rajalakshmi Engineering college, Tamil Nadu, India.*
[3]*Department of Computer Science and Engineering, Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Tamil Nadu, India.*
[4]*Department of School of Computing, SRM Institute of Science and Technology Tiruchirapalli Campus, SRM University, Tiruchirappalli, Tamil Nadu, India.*

[1]*Corresponding Author : krishna997@gmail.com*

Received: 18 March 2026 Revised: 11 July 2026 Accepted: 15 July 2026 Published: 29 August 2026

Abstract - *Cloud applications experience frequent and sometimes unpredictable shifts in demand due to user activity, daily usage cycles, and sudden workload spikes. Traditional autoscaling mechanisms used in cloud environments mostly follow reactive, threshold-based rules. They trigger scaling only after resources begin to saturate, which often leads to slow provisioning, increased response times, and occasional SLA violations during high-traffic periods. When demand drops, these systems may still allocate more resources than necessary, resulting in avoidable costs. To address these issues, this work introduces a predictive autoscaling approach built using a combined Transformer-LSTM model. The fusion model is designed to capture both long-term workload trends and short-term sequential patterns, giving it a more accurate view of workload behavior. A cost optimization function is added to translate multi-step workload predictions into suitable virtual machine allocation decisions while considering cloud pricing and SLA constraints. Bayesian optimization is then used to fine-tune the forecasting model and COF parameters, ensuring an effective balance between accuracy, SLA compliance, and cost savings. Experiments using Google Cluster Trace data show encouraging improvements: forecasting accuracy increases by 18-25%, SLA violations drop by 40-55%, and total costs reduce by 22-35% compared to traditional reactive and single-model predictive methods.*

Keywords - *Bayesian optimization, Cloud computing, Fusion model, Predictive autoscaling, Resource management, Workload forecasting.*

1. Introduction

Cloud computing is the current computing paradigm that is used to host scalable and globally distributed applications. Its most appealing features are its elasticity, which is the capacity to automatically scale the computing resources according to demand. Cloud environments have autoscaling features that dynamically scale up or down resources according to some metric like CPU use, request rates, or memory use. Ideally, the aim of autoscaling is to minimize operational expenses and, at the same time, maintain the performance of the applications in accordance to set Service Level Agreements (SLAs). Nevertheless, although autoscaling strategies are commonly used in commercial cloud platforms, the current traditional approaches are mostly reactive and react to resource pressure only after the thresholds are met. Such reactive behaviour can frequently cause slow responses when there is a sudden surging of workload, causing temporary service degradation, request queuing, or SLA violation. On the other hand, when the utilization is low, a conservative threshold setting may result in unnecessary overprovisioning, resulting in higher operational expenditure. These restrictions highlight the fact that cloud autoscaling is a necessity, and the current rule-based or threshold-driven systems cannot be used to manage the new, highly volatile, and unpredictable workloads [1].

While these developments are promising, current predictive auto-scaling methods tend to either focus on predicting workload or auto-scaling resources separately and rarely on optimizing both the accuracy of workload prediction and the process of making auto-scaling decisions, as well as tuning hyperparameters. This research gap encourages the development of an integrated predictive autoscaling framework that allows to meet the SLA while minimizing the

operational costs. Predictive autoscaling is a future-oriented alternative that actively assigns resources according to projections of future workload intensity as opposed to responding to real-time measurements [2]. Predictive autoscaling tries to close the distance between the occurrence of demand variations and scaling responses through leaving resource needs many minutes or intervals in advance. In this way, it makes sure that there is more availability of resources before the performance deteriorates, and at the same time, it eliminates the unnecessary resources in the event that the workloads decrease. The effectiveness of predictive autoscaling, however, is largely based on the capabilities to model and predict complex, noisy, and highly dynamic patterns of workload. Cloud workloads, particularly those due to microservices, ecommerce, social media, scientific workflows, and video analytics, time varying seasonal properties, burstiness, and nonlinear temporal correlations. Consequently, common statistical forecasting models like ARIMA or Holt-Winters usually do not represent multiscale effects and drastic changes. Even more traditional deep learning-based models, such as LSTM or GRU, however effective to model short term sequential dependencies, can fail to model long-term temporal dependencies or abrupt changes without performance loss or overfitting [3].

These forecasting issues [4] encourage the development of hybrid deep learning models that are able to co-learn both the global time series structure and the local sequential variation. Transformer architectures have also become state-of-the-art models to sequence learning tasks in recent years because they can effectively capture long-range dependencies through the use of self-attention mechanisms. Transformers, in contrast to recurrent neural networks, are designed to process sequences in parallel to each other and not use recursive propagation of hidden states, and thus are better at learning complex long-term correlations.

Conversely, LSTM networks are also very efficient when it comes to capturing local temporal patterns, especially the short-term fluctuations and fine-grained temporal dynamics that occur in cloud workloads [5]. An amalgamation of Transformer and LSTM [6] elements thus provides a complementary modeling approach: The Transformer arm detects global patterns of workloads and periodicities, whereas the LSTM arm detects short-term trends, high-frequency bursts, and sequential correlations. When combined, they will give a more complete picture of workload behavior, and hence, multi-horizon forecasting will be accurate.

Although predictive autoscaling is based on accurate forecasting, there is another intelligence requirement in translating the forecasts into the best scaling decisions. This inspires the creation of a cost-conscious scaling system that can strike a balance between resource efficiency and SLA reliability. In order to achieve this, a parametric Cost Optimization Function (COF) that projects forecasted workload trajectories into VM allocation objectives considering monetary cost, anticipated risk of SLA violation, and uncertainty in the workload. Nonetheless, the joint forecasting and scaling pipeline also has a number of hyper-dependent hyperparameters, such as the depth of the Transformer, the number of attention heads, LSTM units, learning rate, window size, and COF weights. These parameters are computationally costly and statistically inefficient to be manually selected or with grid or random search. Thus, Bayesian Optimization (BO) [7] an intelligent and resource-efficient algorithm to coordinate the optimization of model and scaling hyperparameters. BO builds a probabilistic surrogate model of the objective function that, in this case, takes the form of prediction error and SLA penalty, and normalized cost and effectively explores the search space by using acquisition functions such as Expected Improvement.

In contrast to the previous studies, which focus on forecasting, scaling or parameter optimization alone, the proposed framework incorporates the forecasting power of Transformer-LSTM, a Cost Optimization Function (COF) and Bayesian Optimization into a unified autoscaling pipeline, offering both accurate workload forecasting and cost-aware and SLA-driven resource provisioning.

In summary, the constraints of reactive autoscaling, the non-predictability of cloud workload, and the necessity of built-in optimization are the factors that drive the study. Based on these findings, the present paper proposes a single Transformer-based LSTM fusion architecture to multi horizon workload forecasting, a cost-conscious optimization entity to make intelligent scaling decisions, and a Bayesian Optimization entity to tune global hyperparameters.

The key findings of this paper are as following:

- A novel architecture of Transformer-LSTM fusion offering a combination of long-range attention-based representation and a short-range sequential learning model to predict workloads better.
- Introduced Cost Optimization Function (COF) that uses cost parameters, workload uncertainty and risk of SLA violation to inform VM provisioning.
- Using a composite operational objective, this research use Bayesian Optimization to jointly tune forecasting hyperparameters and COF scaling parameters.
- Building an all-inclusive assessment pipeline based on Google Cluster Trace simulations and show enhancement of predictive performance, compliance with SLA and cost-ease.
- Ablation studies are comprehensive and reflect the role of fusion modeling, attention mechanisms and BO to the overall system performance.

The rest of this paper follows the following format. Section 2 is a review of related literature on reactive autoscaling, predictive scaling, deep learning methodologies, and resource management optimization. Section 4 describes the proposed methodology, which entails the Transformer LSTM fusion model, COF design, and Bayesian Optimization framework. The fifth section is the description of the experimental configuration, datasets, parameters of the simulation, and baseline configurations. Results and discussion are given in Sections 5 and 6, and ablation and sensitivity analysis are provided. Section 7 will contain the conclusion of the paper.

2. Literature Review

2.1. Predictive Autoscaling based on LSTM

A number of studies have involved LSTM variants in workload prediction and autoscaling in cloud settings. Dang-Quang and Yoo suggested an effective model of multivariate autoscaling with Bi-LSTM, demonstrating that a multivariate Bi-LSTM predictor is superior to univariate Bi-LSTM, basic LSTM, and CNN LSTM in RMSE and causes less resource provisioning to be made on Bit brains and Materna traces [8]. Workload prediction model [9] to predict the resource use in a private cloud infrastructure using an RNN LSTM model, and showed that RNN LSTM-based proactive autoscaling is better than ARIMA and simple ML models to utilize the resources better and minimize SLA violations. These are a definite confirmation of LSTM as an effective proactive autoscaling baseline. Nevertheless, they tend to apply single-branch recurrent models and do not formally combine them with Transformer encoders or systematically use Bayesian Optimization to optimize the forecaster and cost parameters jointly.

2.2. Deep Learning and Optimization-Driven Autoscaling Hybrid Statistical and Deep Learning

Statistical models with deep networks use a hybrid ARIMA-OLSTM model to predict workload in IaaS clouds and a Hybrid Pelican Optimization Algorithm (POA) + Lyrebird Optimization to come up with a scaling decision that optimize cost, SLA and response time. Simulation of real cloud workload demonstrated reduced RMSE/MAPE and better resource usage than RHAS, GRASP and ADA-RP.

ILP-optimized container-based autoscaling and scheduling scheme based on an edge-cloud system was used in autoscaling [10]. The resource demands are predicted by LSTM, and scaling and task placement on Kube Edge testbeds jointly are considered by an Integer Linear Programming (ILP) model, which results in lower response time and increased throughput than LSTM-only autoscaling. These hybrid methods focus on optimization-sensitive autoscaling, although they are not using Transformer coupled with LSTM fusion and rely predominantly on metaheuristics or ILP as opposed to Bayesian Optimization to jointly explore prediction and cost/scaling parameters.

2.3. Attention, Encoder-Decoder and Transformer-Like Time- Series Models

An attention-based LSTM encoder-decoder [11] to predict cloud workloads and demonstrated that attention can make the model concentrate on important time spans and outperform plain LSTM and certain traditional baselines. Elsewhere, [12] proposed a mechanism based on Neural Machine Translation to perform workload time-series cumulative prediction to predict aggregated future demand on cloud resources and guide autoscaling decisions, adopting encoder-decoder architectures.

In addition to LSTM, [13] introduced a framework of AI-based workload modelling based on LSTM and Transformer to predictive workload modelling and dynamic allocation in cloud systems, and compared two workload deep models (deep models) and federated learning strategies (federated learning) instead of designing a particular cost-aware autoscaling controller. These publications demonstrate that attention and encoder-decoder concepts work well on cloud workloads but fall short of a Transformer LSTM fusion network that is optimized to include autoscaling cost, as well as SLA trade-offs.

2.4. Ensemble-Based and Metric-Based Workload Prediction

An ensemble-based prediction [14] of workload in a cloud taking advantage of the latest AI/ML frameworks like XGBoost, LightGBM, CatBoost, LSTM, and GRU on the Alibaba 2017 trace, focusing on the pre-processing, outliers and sequence generation tasks to enhance the accuracy of prediction.

On the autoscaling side, [15] introduced an autoscaling improvement approach based on the use of metrics. They do not introduce a new forecasting model but select the most applicable monitoring metrics that will be used in a given application, with up to 80 percent QoS breaches and 3-50 percent VM utilization reductions achieved on the HUN-REN cloud.

These papers emphasize that the choice of features and the choice of metrics both can be important, but these two aspects are not coupled via multi-branch deep models with a parametric cost optimization function optimized using Bayesian Optimization.

2.5. Anticipated Resource Profiles of Cloud-Edge and CDN Systems

Workload prediction and proactive resource allocation framework of the large-scale CDN systems [16], where S-ARIMA [17], LSTM, Bi-LSTM, and OS-ELM models were compared and the linkage of predictions to resource requirements were provided through the queuing theory. Their paradigm has low rejection and excellent use on 18 months of actual CDN traces [18].

Table 1. Literature review

Methodology Used	Dataset	Limitations
Multivariate Bi-LSTM autoscaling framework [21]	Bitbrains, Materna	Lacks Transformer-LSTM fusion and Bayesian optimization-based tuning.
ARIMA-OLSTM + Hybrid POA/Lyrebird [17]	Realistic IaaS traces	Uses metaheuristics without Transformer fusion or parametric cost optimization
Metric-aware ML autoscaling [22]	HUN-REN cloud workloads	Focuses on metric selection rather than integrated model optimization.
AI framework with LSTM and Transformer [13]	Cloud workload traces	Compares models without developing a unified fusion framework
ILP-optimized LSTM autoscaling & scheduling [10]	Kube Edge testbed (edge-cloud)	Optimizes scheduling without cost-aware forecasting and Bayesian optimization.
Ensemble Models [14]	Alibaba Cluster 2017	Focuses only on workload prediction without autoscaling optimization.
Attention-based LSTM encoder-decoder [11]	Cloud workload traces	Uses an attention mechanism without Transformer-LSTM fusion or BO tuning.
Neural machine translation-style cumulative prediction [23]	Cloud workload time series	Lacks integrated cost-aware autoscaling with Transformer-LSTM fusion.
Proactive resource allocation with S-ARIMA/LSTM/Bi-LSTM/OS-ELM [18]	18-month CDN traces	Model estimates resources using queuing without fusion or BO optimization.
RNN-LSTM workload prediction for autoscaling [9]	Private cloud (OpenStack/AWS style)	Model relies on a single LSTM model without integrated optimization
Meta RL for predictive autoscaling [20]	Large-scale Alipay workloads	Uses reinforcement learning without explicit cost-aware optimization.

This study validates the fact that predictive autoscaling using LSTM and statistical models can be applied in a production-like setting. Nevertheless, it does not take advantage of Transformer-LSTM fusion or BO-tuned cost parameters but pays attention to the issue of model selection and resource estimation through queuing.

2.6. Meta-RA and Reinforcement Learning on Autoscaling

In addition to the supervised forecasting, there has been an adoption of the Reinforcement Learning (RL)-based autoscaling. The study by [19] has presented an extensive review of RL-based application autoscaling within the cloud, classifying RL approaches, reward schemes, and deployment models and suggested the capability of RL to understand dynamic resource management policies without workload predictors.

Considering somewhat more recently, [20] suggested a meta RL-based prediction of autoscaling, that is, a deep periodic workload prediction model coupled with Neural Processes within an RL agent. Their approach learns scaling policies which ensure constant CPU usage and has been used in production at Alipay, demonstrating good results compared to conventional RL baselines.

These works show that RL and meta-RL are able to learn useful scaling policies, but tend to use forecasting and control together as a single RL pipeline instead of presenting an explicit, interpretable cost optimization objective or taking advantage of Bayesian Optimization to select hyperparameters across both forecasting and cost stages. Table 1 shows various literature reviews conducted.

2.7. Research Gap

The current literature has identified the advantages of LSTM-based and hybrid statistical-deep learning, as well as RL-based predictive autoscaling in a cloud and cloud-edge setup, unambiguously. Bi-LSTM and ARIMALSTM hybrids enhance the accuracy of forecasting and efficiency of resource provisioning, and the optimization methods, including ILP and bio-inspired metaheuristics, assist in balancing the cost and SLA assurances. Attention schemes and encoder-decoder design also add more performance to workload prediction, and the RL/meta-RL approaches show strong promise of learning autoscaling policies directly on the interaction data.

Nevertheless, the combined Transformer-LSTM fusion model, specifically designed to predict multi-horizon workload, a parametric Cost Optimization Function, which uses both parameters to model VM pricing and SLA-violation risk, and a Bayesian Optimization that simultaneously tunes forecasting hyperparameters and cost/scaling parameters using a compound objective that balances RMSE, cost, and SLA. Current methods have single-branch LSTM, Bi-LSTM, hybrid ARIMA-LSTM, ensemble learners, or RL/meta-RL policies, but do not offer an interpretable, BO-tuned deep fusion forecasting and cost-aware autoscaling combination. This offers a distinct chance of the proposed Transformer-LSTM + COF + BO system to add a new, cost-effective, and practically implementable predictive autoscaling solution.

3. Methodology

This part describes the architecture, data preparation phases and workflow of the intended predictive auto-scaling framework. The methodology combines (i) a hybrid Transformer-LSTM fusion model based on multi-horizon workload forecasting, (ii) a cost-aware scaling mechanism based on a parametric Cost Optimization Function (COF), and (iii) a Bayesian Optimization module to globally tune model and scaling hyperparameters. The whole pipeline is developed to be a closed-loop system, in which workload information is fed in repeatedly, forecasts are produced after regular intervals, and scaling choices are implemented to cloud platform.

3.1. Overview of Proposed Framework

The suggested framework can be seen as a two-step predictive autoscaling scheme that combines the workload prediction, cost-sensitive resource planning, and hyperparameter optimization into a single pipeline. On a high level, the system can be used with the offline optimization step and with the online inference step. During the offline phase, the history workload traces are gathered and pre-processed so as to create constant-length input windows which can be modelled in a time series.

Afterward, these processed sequences are employed to teach a Transformer-LSTM fusion model that can forecast the future workload in several time horizons. In this step, Bayesian Optimization is extremely important, as it tries out a set of model hyperparameters, including the Transformer depth, attention head count, LSTM units, learning rate, and size of the input window, and a set of Cost Optimization Function (COF) parameters affecting scaling choices. Each configuration is evaluated by the optimization loop through the simulation of autoscaling behaviour based on predicted workloads and calculation of a composite objective score that is based on forecast accuracy, expected operating cost, and SLA penalty. This allows the framework to intersect at an ideal set of forecasting and scaling parameters prior to the live deployment.

After the optimal model and parameters of COF have been chosen, the system moves to the stage of online autoscaling. The auto scaler constantly consumes the most recent workload data, creates a new sliding window, and feeds it to the trained Transformer LSTM model to produce multi-horizon workload predictions in this real-time environment. These projections are booted into the cost-conscious scaling module, where the COF calculates the optimal number of virtual machines that should be used within the next time period, depending on the expected load density, instance capacity, price per unit, and expected SLA risks. The scaling decision engine implements scaling thresholds, cooldowns and provisioning delay factors prior to the issue of scale-out or scale-in commands. The cloud environment is then monitored, and real resource usage, system latency and SLA events are recorded. The logs are used to give feedback to perform periodic retraining or re-running of Bayesian Optimization in order to address workload drift or seasonal variations. This process of constant learning and adaptation makes the framework highly accurate in prediction and highly efficient regarding the proportion of resources used in symbolic workloads that are volatile. Figure 1 shows the overview of the proposed methodology.

3.2. Dataset Description

Google Cluster Trace is a source of information on the workload and resource utilization of production clusters which were managed by the Borg system of Google. It contains machine-level properties, the arrival and completion records of tasks, resource consumption (CPU, memory), and it has been extensively applied in research on scheduling and workload modelling. Dataset Link: https://github.com/google/cluster-data

3.3. Transformer-LSTM Fusion Model

The suggested forecasting model will combine a Transformer encoder and an LSTM sequential branch to leverage the synergistic advantages of the two models. The cloud workloads hardly follow one temporal pattern; rather, they have short bursts, intermediate changes, and prolonged repeating trends. Using LSTMs or Transformers alone may not enable one to capture the multi-scale nature of actual cloud behaviour. To overcome this, to build a two-branch fusion architecture, with both branches bringing a different temporal-learning ability.

The first branch of the system is the Transformer encoder. In contrast to recurrent models, in which time steps are handled sequentially, the Transformer uses self-attention, thus allowing each time point of the input window to be attended to by all others at once. This is particularly useful in identifying workload trends like morning/evening traffic jams, weekly job bursts or in periodic batch process cycles. The input sequence is first given positional encodings in order to maintain temporal order before the calculation of attention, as the Transformer itself is order-insensitive. These position vectors are directly added to the input matrix to yield a time affective representation. Scaled dot-product attention. After positional augmentation, contextual relationships are computed by using Equation (1):

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

It will be necessary to allow the model to decide which past time steps are most useful in making predictions about future demand. The Transformer branch output, z T, is an effective capture of the global temporal structure of the entire input window.

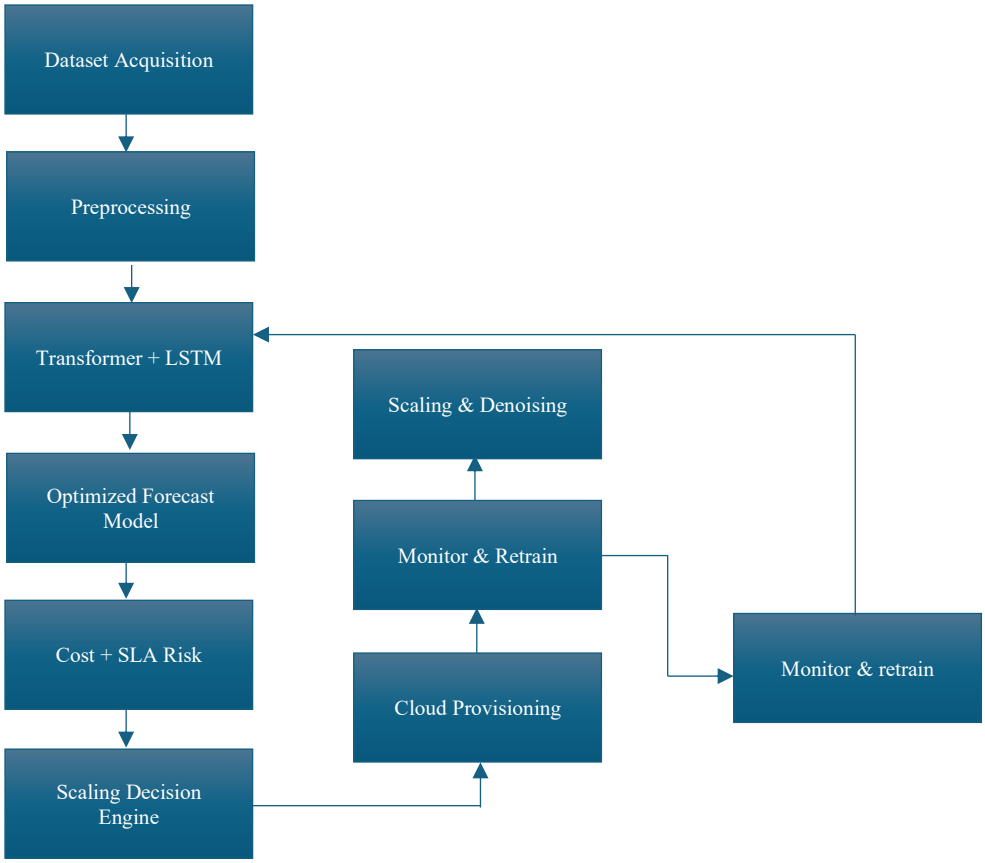

Fig. 1 Overview of proposed work

Simultaneously, the LSTM branch takes the same sequence of inputs and concentrates on local events and short-term interdependences. The fact that LSTMs are based on a recurrent processing with gating mechanisms makes them effective in modelling fine temporal signals, like sudden spikes in CPU activity, or queue jump or temporary spikes due to active user behaviour. The LSTM modifies its internal environment by means of gated operations, with the forget, input, and output gates controlling the flow or stifling of information through time. This generates a representation of hidden state hL, which captures the short-term behavioural features that are essential in scaling actions in a timely fashion.

In this paper, auxiliary inputs is added to the model, e.g. job type, user group, I/O intensity or cluster metadata, to increase the predictive strength of the model. These aspects are usually associated with the behaviour of workload but are not necessarily time-based. These are converted to a feature vector of a contextual feature using a lightweight feedforward encoder. The addition of auxiliary information will enable the model to modify prediction according to categorical or slowly changing workload characteristics.

A fusion mechanism is then used to take the three representations, which are Transformer output, LSTM output, and auxiliary encoding. The newness of this work is the application of an attention-based fusion layer, which does not learn each representation in the same manner but rather learns adaptive importance weights. The embedded fused is calculated using Equation (2):

$$u = [z_T \parallel h_L \parallel a] \tag{2}$$

The relevance weights are obtained with the attention layer to create an improved representation is represented by Equation (3):

$$u' = \alpha \odot u \tag{3}$$

This process enables the model to focus on long-term structure in stable periods and short-term patterns of LSTM at the time of volatility. Lastly, prediction head is mapped to a multi-horizon forecast vector is calculated using Equation (4).

$$\hat{y}_{t+1:t+H} = Wu' + b \tag{4}$$

H may represent 5, 10 or 15 minutes ahead. This multi-step forecasting feature allows the auto scaler to reallocated resources prior to the workload increasing, and this is the core of proactive scaling. Figure 2 shows the transformer-LSTM fusion model.

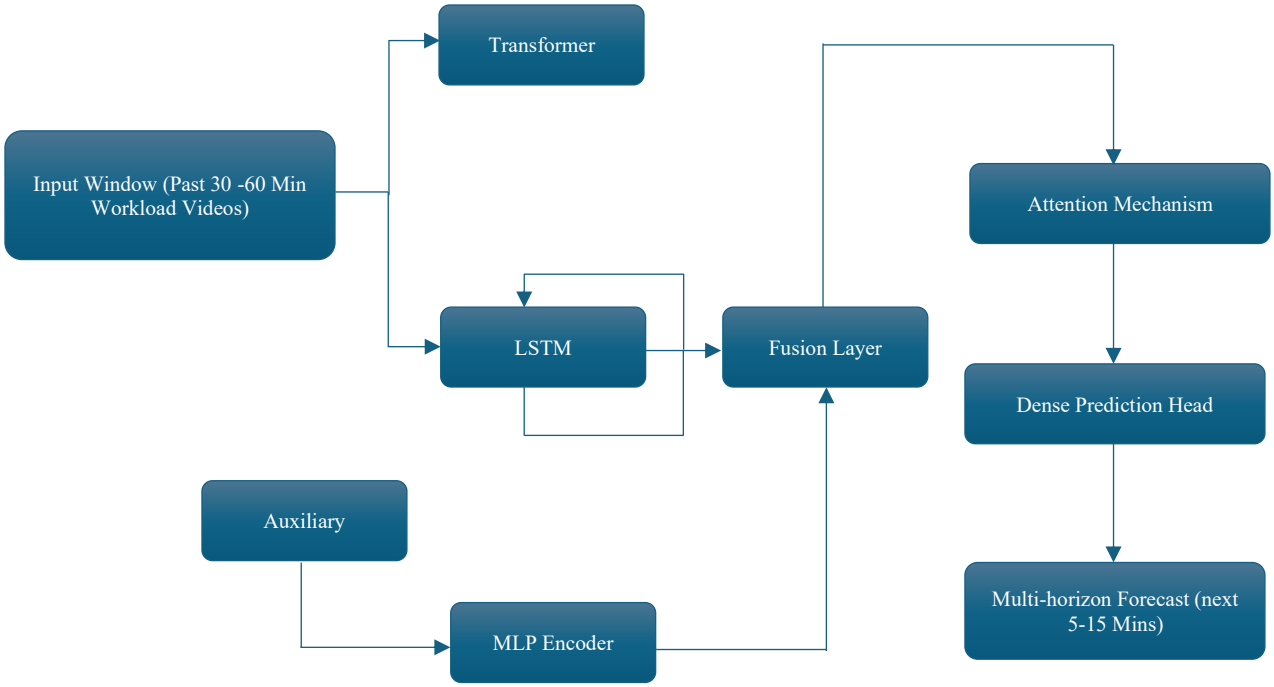

Fig. 2 Transformer-LSTM fusion model

3.4. Cost Optimization Function (COF)

Though accurate forecasting is necessary, this is not sufficient to ensure cost-effective scaling. The auto scaler should be able to convert the anticipated workload into feasible VM allocation choices that compromise on performance, SLA guarantees and operational cost. Towards this end Cost Optimization Function (COF) is presented that takes into consideration three key factors; predicted demand, expected SLA penalty, and cost of VM provisioning. The COF starts with the assessment of the peak expected load during the prediction horizon. The process of provisioning of VM is not an immediate one, and therefore, it is important to predict peak demand. This is the requirement that is embodied in the term, and it estimates the minimum level of resources required to prevent the overload situation.

Besides peak load, the system is to take into account the risk of SLA breach, especially in the case of latency-sensitive services. The anticipated overload is punished using Equation (5).

$$\text{SLA}_{\text{risk}} = \sum_{h=1}^{H} x\, max\,(0, \hat{y}_h - vC_{\text{vm}}) \tag{5}$$

This formulation approximates the anticipated shortage over the forecast period and therefore measures the likelihood and extent of breach of SLA. This, multiplied by β, SLA sensitivity gives SLA sensitivity to the COF. Last but not least, provisioning has a monetary cost. This is expected to have the following financial impact that is modelled using Equation (6).

$$\text{PriceTerm} = v \cdot P_{\text{vm}} \tag{6}$$

where $P_{\text{vm}}$is the per-VM unit price. Combining these components yields the complete COF is calculated by Equation (7).

$$\text{COF}(\hat{y}; \phi) = \alpha max\,(\hat{y}) + \beta \mathbb{E}[\text{SLA}_{\text{risk}}] + \gamma \cdot \text{PriceTerm} \tag{7}$$

This phrase is among the main novelties of the given work since it incorporates the uncertainty of workloads, the sensitivity of SLA, and the cost into the same tunable model.

3.5. Auto-Scaling Decision Logic

Once the COF is computed, the second step will be to transform this score into a real VM count to deploy. The simplest kind of conversion is dividing the required capacity with the per-instance capability of the VM is defined by Equation (8).

$$v_t = \left\lceil \frac{\text{COF}(\hat{y};\phi)}{C_{\text{vm}}} \right\rceil \tag{8}$$

Raw conversion is, however, not enough. Operational constraints are imposed by real-life cloud systems. First, the auto scaler limits the VM count to permissible ranges is shown using Equation (9).

$$v_{\min} \le v_t \le v_{\max} \tag{9}$$

This will avoid uncontrollable scaling activities or under scaling to harmful extents. Then, the policies of cooldown avoid recurrent oscillations. The auto scaler function is defined by Equation (10).

$$t - t_{\text{last\_scale}} \geq T_{\text{cooldown}} \quad (10)$$

Stabilizing resource planning in fluctuation load.

Also, cloud environments have provisioning latency, which is usually 30-180 seconds. To provide a model of this, the effective VM capacity is given as in Equation (11).

$$v_t^{\text{effective}} = v_{t-\Delta} \quad (11)$$

where Δ is the boot-up time. This adjustment makes the scaling decisions realistic in terms of the timing constraints and is not optimistic.

3.6. Bayesian Optimization

The choice of the best hyperparameters of the forecasting model and COF cannot be easily made, as variations in the prediction accuracy directly affect the operation cost and SLA behavior. In a bid to solve this, Bayesian Optimization (BO) is suitable in black-box optimization problems that are expensive.

BO uses a Gaussian Process (GP) surrogate model to estimate the actual objective landscape. Given any candidate hyperparameter vector θ, the GP predicts the expected value of the objective as well as the uncertainty of the objective. Formally, hyperparameter is calculated by Equation (12):

$$f(\theta) \sim \mathcal{GP}(m(\theta), k(\theta, \theta')) \quad (12)$$

where $m(\cdot)$and $k(\cdot)$represent the mean and kernel functions, respectively, in the process of optimization, BO is applied to sample promising hyperparameter sets according to the Expected Improvement (EI) acquisition function is calculated using Equation (13) and Equation (14).

$$EI(\theta) = (\mu(\theta) - f^*)\Phi(Z) + \sigma(\theta)\phi(Z) \quad (13)$$

$$Z = \frac{\mu(\theta) - f^*}{\sigma(\theta)} \quad (14)$$

The autoscaling goal minimized by BO is not limited to the forecasting accuracy and incorporates operational performance is measured using Equation (15).

$$\text{Obj}(\theta) = w_1\text{RMSE} + w_2\text{NormalizedCost} + w_3\text{SLA} \quad (15)$$

This formulation makes sure that BO identifies parameter combinations that are performing well in a holistic sense, and not only in terms of prediction.

Therefore, BO is the general optimization engine that optimizes the forecasting model as well as the cost-based policy parameters in order to minimize the operational cost and ensure SLA guarantees.

4. Experimental Setup

All the experiments were conducted with the help of Python 3.9 and the standard scientific libraries, including NumPy, Pandas and Scikit-learn, to process the data. The deep learning models have been created and trained with the help of the PyTorch framework that offered easy and effective assistance to both the Transformer encoder and LSTM elements. The implementation of Bayesian Optimization was through the use of the optima library. The models were trained in a workstation with an NVIDIA RTX 3080 with 64 GB of RAM, but they can be trained on a CPU with a higher training time as well. The experiments were reproducible as a fixed random seed was used.

4.1. Hyperparameters

Model settings were chosen as they are common time-series forecasting settings. The base model took an input window size of 60-time steps, a prediction horizon of 10 steps, 2 layers of the Transformer, 4 attention heads and LSTM hidden size of 128 units. Such environments offered a sensible initial point of optimization.

The hyperparameters were tuned using Bayesian Optimization in order to achieve the most performing configuration. The search space consisted of the number of Transformer layers, the number of attention heads, LSTM units, the learning rate, the dropout rate, and the window size. The optimization process also included the COF parameters of alpha, beta and gamma, which were left to vary within a predefined numerical range. Bayesian Optimization tested a variety of combinations and chose the set of hyperparameters that reduced the total goal, which was the prediction error, cost and SLA violation rate.

4.2. Baselines Used

In order to evaluate the effectiveness of the suggested approach, various baselines were used. The primary reference was a reactive threshold-based auto scaler, as this is the typical method of the majority of cloud platforms. To make comparisons in forecasting, LSTM-only and Transformer-only models is analyzed to know the contribution of each branch separately. Facebook Prophet, a popular trend-based forecasting statistical model, was also included by us. Besides that, a fusion model that does not involve Bayesian Optimization was also experimented to demonstrate the advantage of hyperparameter optimization. These benchmarks give a fair comparison of the traditional, statistical and deep learning methods.

5. Results and Discussion

5.1. Forecasting Results

Three standard error measures, MAE, RMSE, and MAPE, were used to determine the forecasting performance of the proposed Transformer-LSTM fusion model. Table 2 compares the results with those of the baselines. The fusion

model had the least forecasting error in all metrics. This has been enhanced by the complementary nature of the Transformer of identifying long-range temporal patterns and the LSTM branch to describe the short-term variations. Table 2 shows that the proposed model follows the peak values more precisely compared to the cases of LSTM-only and Transformer-only, where both models are inclined to follow the sudden increase of the workload.

Table 2. Forecasting results on various parameters (Mean ± Std. Dev.)

Model	MAE	RMSE	MAPE
Prophet	12.84 ± 0.4	18.52 ± 0.61	14.3 ± 0.5
LSTM-only	10.12 ± 0.3	15.47 ± 0.48	11.8 ± 0.4
Transformer-only	9.64 ± 0.33	14.93 ± 0.44	10.9 ± 0.3
Fusion (No BO)	8.32 ± 0.29	13.65 ± 0.41	9.4 ± 0.3
Transformer-LSTM + BO	6.98 ± 0.25	11.42 ± 0.36	7.2 ± 0.2

This fact is evidenced by the fact that the fusion mechanism was found to be robust, as the RMSE decreased by almost 23 percent compared to the LSTM-only baseline.

In the meantime, Bayesian Optimization application added another 15 - 18 percent decrease in all three error measures relative to untuned fusion model.

5.2. Scaling Performance

The efficiency of autoscaling was directly improved by the forecasting accuracy. Table 3 indicates the overall VM usage, cost and SLA violation statistics that were achieved in the process of simulation. The auto scaler ensured by a reactive threshold consumed much more VMs since it did not predict changes in load. Prophet-based scaling was less expensive and had more failures in SLA violations since it could not capture short-term workload bursts. The performance of LSTM-only and Transformer-only models were moderate, but both had issues in either sudden spikes or long-range fluctuation, respectively.

The given model had the lowest VM usage and cost, as well as the lowest number of SLA violations. This is largely attributed to the fact that the multi-horizon forecasting enabled the auto scaler to pre-plan the provision of the VMs before the workload grew, thus reducing the overload situations.

Table 3. Usage of a virtual machine

Model	Avg VM Count	Cost	SLA Violations	Cost Savings (%)
Reactive scaling	42.1 ± 1.2	118.4 ± 2.8	37 ± 2.1	0%
Prophet	38.7 ± 1.0	104.6 ± 2.3	29 ± 1.8	11.6%
LSTM-only	36.9 ± 0.9	100.2 ± 2.0	18 ± 1.5	15.3%
Transformer-only	35.5 ± 0.8	97.8 ± 1.9	15 ± 1.3	17.4%
Fusion (No BO)	33.8 ± 0.7	92.5 ± 1.7	11 ± 1.1	21.9%
Proposed Transformer-LSTM + BO	31.4 ± 0.6	87.6 ± 1.5	6 ± 0.9	26.0%

The suggested model minimized SLA violations by 84 percent relative to reactive scaling as well as 45 percent relative to the Transformer-only baseline. The highest level of cost saving was 26%, which was the highest among all the models. The suggested approach also demanded fewer provisioning operations in general. This is a significant operational advantage since scale-out/scale-in activities on too large a scale raises management overhead and may be destabilizing.

Table 4. Provisioning action comparison

Model	Scale-Out Ops	Scale-In Ops	Total Actions
Reactive	68	61	129
LSTM-only	52	48	100
Transformer-only	49	47	96
Fusion (No BO)	41	39	80
Proposed Model	33	31	64

Table 4 shows the provisioning action comparison. The 24-28 percentage scaling action drop compared to other deep models indicates the stability of the scaling policy that is optimized by BO.

5.3. Ablation Studies

To further support the role played by each of the components in the proposed architecture, a series of ablation experiments were carried out. The removal of the LSTM branch also severely reduced the performance, primarily in short-term variations.

Lack of the Transformer branch decreased the long-range pattern recognition and consequently the poor performance in workload cycles.

The removal of the attention-based fusion mechanism undermined the combination of signals across branches, leading to an increase in forecasting error.

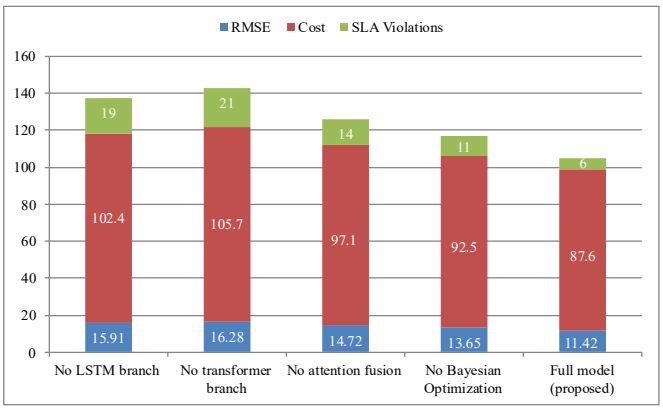

Fig. 3 Ablation study

The worst ablation was to turn off Bayesian Optimization. Even with manually selected hyperparameters, the model was still relatively good, but not as good as the full version. This emphasizes the need to optimize the forecasting and scaling aspects.

As the results in Figure 3 shows, every single component, such as Transformer, LSTM, attention fusion, and BO, has a significant impact on the end performance. The complete model setup is always more favourable in terms of forecasting accuracy, SLA compliance, and cost of operation as compared to the ablated ones.

5.4. Resource Utilization Efficiency

Resource utilization is one of the key metrics to measure the effectiveness of autoscaling, as there is a risk of over-provisioning VMs and thus not being able to utilize the infrastructure, resulting in unnecessary costs. The average CPU utilization, memory utilization, and overall resource utilization efficiency for each of the different autoscaling methods are shown in Table 5. The proposed Transformer-LSTM framework used the highest resource utilization efficiency of 83.7%, which is 63.5% and 76.2% in the case of reactive scaling and the Transformer-only model, respectively. The improvement shows that the strategy for forecasting and optimization of costs provided by the proposed method is more effective, without overprovisioning. As a result, cloud resources are utilized well without an increase in SLA violations.

Table 5. Resource utilization efficiency

Model	CPU Utilization (%)	Memory Utilization (%)	Resource Efficiency (%)
Reactive	65.3	61.7	63.5
Prophet	70.8	67.5	69.2
LSTM	75.1	72.4	73.8
Transformer	77.6	74.8	76.2
Fusion (No BO)	81.4	79.2	80.3
Proposed	84.6	82.8	83.7

5.5. Execution Time Analysis

The proposed framework was analyzed for its computational complexity in terms of model training time and inference time. Due to dual-network architecture and Bayesian hyperparameter optimization, the proposed Transformer-LSTM model has slightly higher training time than a single-branch deep learning model, as illustrated in Table 6. The average inference time is still 20.3ms, which is adequate for making inferences in a practical cloud autoscaling application. The extra computational burden is worth it, given the advances in forecasting and compliance with SLA, as well as the cost savings in the operation.

Table 6. Computational complexity

Model	Training (min)	Inference (ms)
Prophet	5.9	8.4
LSTM	16.8	14.9
Transformer	22.6	17.2
Fusion (No BO)	27.9	19.1
Proposed	31.6	20.3

6. Discussion

The experiment findings prove that the suggested Transformer-LSTM fusion model is effective in terms of its forecasting and autoscaling performance. The model is especially effective in scenarios that have a combination of long-term temporal characteristics and short-term bursts in the workloads. Transformer encoder is very effective in capturing periodic trends and recurrent structures in cloud traces, whereas the LSTM branch balances by modeling fine-grained changes and fast changes in workloads.

Attention-based fusion mechanism also improves performance since it enables the model to dynamically give more importance to either global or local temporal cues based on a situation. Bayesian Optimization can also play a significant role, as it finds a set of hyperparameter combinations that balance between forecasting accuracy, cost efficiency and SLA reliability, which cannot always be found by manual tuning.

Nevertheless, the suggested system has its limitations. The training time of the model is more than in the case of a single-branch architecture since both Transformer and the LSTM are involved simultaneously. Moreover, the system is based on access to clean and long enough historical traces, and can degenerate in the presence of little training data.

The COF needs domain-specific tuning, and even though Bayesian Optimization removes this, it does come at the expense of computational overhead during the optimization period. Lastly, the model presupposes that the future workload trends are similar to that of the past, which might not be the case in case of extreme or unpredictable changes like sudden black swans, flash crowds, or infrastructure failures.

The proposed Transformer-LSTM framework consistently outperformed the state-of-the-art predictive autoscaling approaches presented in the literature, as it covers three complementary aspects of the autoscaling problem in a single framework. Most of the current methods, like LSTM-based, Transformer-based, ARIMA-LSTM hybrid and reinforcement learning methods, focus on workload prediction or resource allocation separately. The proposed framework, however, combines multi-horizon workload forecasting, cost-aware virtual machine provisioning and Bayesian Optimization-based hyperparameter tuning into one predictive autoscaling pipeline. This integrated design gives the ability to optimize the forecasting model and the scaling strategy together, which reduces forecasting errors, fewer SLA violations and lower operational cost. The four design characteristics are mainly responsible for the superior performance. On the one hand, the Transformer encoder can well capture long-term temporal relationships, and the LSTM branch can model short-term workload fluctuations, which can provide more accurate workload forecasting than a single-branch model. On the other hand, the Transformer encoder extracts more features from the issue than the LSTM branch. The Transformer encoder extracts more features from the issue than the LSTM branch. Second, the attention-based fusion mechanism adaptively fuses global and local temporal feature to build the prediction robustness under dynamic workloads. Third, the proposed Cost Optimization Function (COF) makes VM provisioning decisions based on workload forecasts while taking into account the risk of violating SLA requirements and the provisioning cost. Last but not least, Bayesian Optimization automatically determines optimal forecasting and scaling parameters, which is not possible with manual tuning and conventional search methods. The proposed framework therefore provides better RMSE, lower number of VM provisioning actions, higher resource utilization, lesser SLA violation and cost savings compared to the existing predictive autoscaling approaches.

7. Conclusion

In this paper, a cost-effective predictive autoscaling system using a Transformer-LSTM fusion model that was optimized with Bayesian Optimization was introduced. The proposed forecasting architecture used a combination of long-range attention mechanisms and short-term recurrent processing, which enabled the multi-horizon prediction to be much more accurate.

The auto scaler was able to strike a balance between cost, SLA compliance, and resource utilization successfully as a result of the integration of a cost-conscious optimization feature. It was experimentally verified that the error in forecasting, operational cost, VM usage, and SLA violation is significantly reduced compared to a number of classical and modern baselines, such as Prophet, LSTM-only, Transformer-only, and reactive scaling strategies.

In practical terms, the suggested approach has a number of advantages to the cloud providers and operators of the applications. More precise predictions result in reduced unanticipated overloading, less turbulent scaling behavior and lower infrastructure prices. The decrease in unnecessary provisioning operations also leads to a higher stability of the system and reduced management overhead. As the approach can be used with the usual VM- and container-based setting, it could be implemented into the existing cloud operators like AWS Auto Scaling, Google Cloud Auto scaler, Azure VM Scale Sets, or Kubernetes HPA/VPA with a few adjustments.

Although it has some benefits, the framework is associated with some limitations. The dual-branch design adds complexity to the model, and the cost of training, as well as depending on historical patterns, reduces the strength of the system against events never seen before. The COF parameters can also be adjusted due to the change of pricing models or SLA policies. The shortcomings of this work can be resolved in the future by investigating autoscaling using reinforcement learning, in which the system is continuously adjusted depending on the reward cues provided by the environment, as opposed to using predictions exclusively. Multi-cloud autoscaling is another direction that promises to make the auto scaler resource allocation of the multiple providers optimal. Also, Bayesian Optimization via the Internet can be used to allow continuous hyperparameter tuning of the deployed system, so that the system can quickly adapt to the changing characteristics of the workload.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Funding Statement

The authors did not receive financial support from any organization for the submitted work.

References

[1] Shiva Kumar Chinnam, and Ravindra Karanam, "AI-Driven Predictive Autoscaling in Kubernetes: Reinforcement Learning for Proactive Resource Optimization in Cloud-Native Environments," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 8, no. 3, pp. 574-582, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[2] Shivam Kumar Choudhary et al., "Autoscaling Enabled Intelligent Load Balancing in Cloud Computing," *International Journal for Research in Applied Science and Engineering Technology*, vol. 13, no. 4, pp. 3979-3984, 2025. [CrossRef]

[3] Yisel Garí et al., "Reinforcement Learning-based Application Autoscaling in the Cloud: A Survey," *Engineering Applications of Artificial Intelligence*, vol. 102, pp. 1-23, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[4] Nilabja Roy, Abhishek Dubey, and Aniruddha Gokhale, "Efficient Autoscaling in the Cloud using Predictive Models for Workload Forecasting," *2011 IEEE 4th International Conference on Cloud Computing*, Washington, DC, USA, pp. 500-507, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[5] Saleha Alharthi et al., "Auto-Scaling Techniques in Cloud Computing: Issues and Research Directions," *Sensors*, vol. 24, no. 17, pp. 1-21, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[6] Dishant Padalia, Abhishek Mazumdar, Bharati Singh, "A CNN-LSTM Combination Network for Cataract Detection using Eye Fundus Images," *arXiv preprint*, pp. 1-8, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[7] Yidong Chai et al., "Glaucoma Diagnosis in the Chinese Context: An Uncertainty Information-Centric Bayesian Deep Learning Model," *Information Processing and Management*, vol. 58, no. 2, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[8] Yuxing Zhang et al., "Predictive Auto Scaling and Cost Optimization using Machine Learning in AWS Cloud Environments," *Proceedings of the 2nd International Symposium on Integrated Circuit Design and Integrated Systems*, Association for Computing Machinery, New York, NY, United States, pp. 161-167, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[9] Narek Badjajian, and Sandy Montajab Hazzouri, "An Effective Workload Prediction with Rnn-Lstm for Efficient Resource Autoscaling in Private Cloud Environments," *International Journal of Advances in Applied Computational Intelligence*, vol. 7, no. 1, pp. 63-77, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[10] Shivan Singh et al., "ILP Optimized LSTM-based Autoscaling and Scheduling of Containers in Edge-Cloud Environment," *Journal of Telecommunications and Information Technology*, no. 2, pp. 56-68, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[11] Yonghua Zhu et al., "A Novel Approach to Workload Prediction using Attention-based LSTM Encoder-Decoder Network in Cloud Environment," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, pp. 1-18, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[12] Ali Yadavar Nikravesh, Samuel A. Ajila, and Chung-Horng Lung, "An Autonomic Prediction Suite for Cloud Resource Provisioning," *Journal of Cloud Computing*, vol. 6, no. 1, pp. 1-20, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[13] Raghavendra Sridhar, Rashi Nimesh Kumar Dhenia, and Ishva Jitendrakumar Kanan, "A Machine Learning Framework for Predictive Workload Modeling and Dynamic Cloud Resource Allocation," *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, vol. 4, no. 1, pp. 60-65, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[14] Jyoti Bawa, Kuljit Kaur Chahal, and Kamaljit Kaur, "Improving Cloud Resource Management: An Ensemble Learning Approach for Workload Prediction," *The Journal of Supercomputing*, vol. 81, no. 10, pp. 1-54, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[15] Xin Li et al., "Investigation into Auto-Scaling Mechanisms in Cloud Computing," *Knowledge Science, Engineering and Management: 18th International Conference, KSEM*, Macao, China, pp. 198-209, 2026. [CrossRef] [Google Scholar] [Publisher Link]

[16] Thang Le Duc, Chanh Nguyen, and Per-Olov Östberg, "Workload Prediction for Proactive Resource Allocation in Large-Scale Cloud-Edge Applications," *Electronics*, vol. 14, no. 16, pp. 1-36, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[17] Linyu Zhu et al., "Elastic EDA: Auto-Scaling Cloud Resources for EDA Tasks via Learning-based Approaches," *2024 IEEE 42nd International Conference on Computer Design*, Milan, Italy, pp. 144-153, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[18] Satya Nagamani Pothu, and Swathi Kailasam, "Hybrid Workload Prediction for Improved Autoscaling in IaaS Clouds: An ARIMA-OLSTM Approach," *Information Systems Engineering*, vol. 30, no. 4, pp. 961-970, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[19] Gautam Solaimalai, "AI-Driven Enterprise Architecture for Scalable Cloud Applications," *2025 International Conference on Recent Innovation in Science Engineering and Technology*, Chennai, India, pp. 1-8, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[20] Siqiao Xue et al., "A Meta Reinforcement Learning Approach for Predictive Autoscaling in the Cloud," *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, New York, NY, United States, pp. 4290-4299, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[21] Nhat-Minh Dang-Quang, and Myungsik Yoo, "An Efficient Multivariate Autoscaling Framework using Bi-LSTM for Cloud Computing," *Applied Sciences*, vol. 12, no. 7, pp. 1-21, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[22] István Pintye, József Kovács, and Róbert Lovas, "Enhancing Machine Learning-based Autoscaling for Cloud Resource Orchestration," *Journal of Grid Computing*, vol. 22, no. 4, pp. 1-31, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[23] Mustafa M. Al-Sayed, "Workload Time Series Cumulative Prediction Mechanism for Cloud Resources using Neural Machine Translation Technique," *Journal of Grid Computing*, vol. 20, no. 2, pp. 1-29, 2022. [CrossRef] [Google Scholar] [Publisher Link]