



PDF Download
3773276.3774300.pdf
22 January 2026
Total Citations: 0
Total Downloads: 70

 Latest updates: <https://dl.acm.org/doi/10.1145/3773276.3774300>

RESEARCH-ARTICLE

Balancing Compression and Prediction: A Hybrid Autoencoder–LSTM Framework for Cloud Workloads

LIDIA KIDANE, Umeå University, Umea, Västerbotten, Sweden

PAUL TOWNEND, Umeå University, Umea, Västerbotten, Sweden

THIJS METSCH, Sun Microsystems, Santa Clara, CA, United States

ERIK ELMROTH, Umeå University, Umea, Västerbotten, Sweden

Open Access Support provided by:

Umeå University

Sun Microsystems

Published: 01 December 2025

[Citation in BibTeX format](#)

BDCAT '25: IEEE/ACM 12th International Conference on Big Data Computing, Applications and Technologies
December 1 - 4, 2025
Nantes, France

Conference Sponsors:
SIGARCH

Balancing Compression and Prediction: A Hybrid Autoencoder–LSTM Framework for Cloud Workloads

Lidia Kidane
Umeå University
Umea, Sweden
lkidane@cs.umu.se

Thijs Metsch
Sun Microsystems
Neubiberg, Germany
tmetsch@gmail.com

Paul Townend
Umeå University
Umea, Sweden
pault@cs.umu.se

Erik Elmroth
Umeå University and Elastisys
Umea, Sweden
elmroth@cs.umu.se

Abstract

Accurate future workload prediction is an essential step for proactive resource allocation and efficient provisioning in cloud computing environments. Deep learning strategies have proven successful for this task, but they face challenges due to the high dimensionality of monitoring data, extensive preprocessing requirements, and computational overhead. In this paper, we propose a hybrid framework that integrates autoencoders for workload compression with Long Short-Term Memory (LSTM) networks for time-series forecasting. Unlike prior studies, our approach systematically analyzes the trade-off between compression ratio and predictive accuracy, demonstrating how dimensionality reduction can improve both scalability and robustness. Thereby reducing the computational burden associated with processing massive-scale monitoring data. Experiments conducted on both synthetic and real-world datasets demonstrate that the proposed method achieves up to 60% data compression with minimal reconstruction loss, while also improving prediction accuracy compared to baseline LSTM models. We evaluate the overall performance of the framework using various metrics, including data reduction ratio, prediction accuracy, and the effects of different compression stages on predictive performance. Additionally, we quantify the computational savings in terms of CPU usage, memory footprint, and training/inference times, confirming the framework's feasibility for real-world deployment. These results underscore the potential of integrating compression and prediction to achieve scalable, accurate, and resource-efficient management of cloud workloads.

Keywords

Cloud computing, Workload prediction, Data compression, Information extraction, Autoencoders

ACM Reference Format:

Lidia Kidane, Paul Townend, Thijs Metsch, and Erik Elmroth. 2025. Balancing Compression and Prediction: A Hybrid Autoencoder–LSTM Framework for Cloud Workloads. In *IEEE/ACM 12th International Conference*

on Big Data Computing, Applications and Technologies (BDCAT '25), December 01–04, 2025, Nantes, France. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3773276.3774300>

1 Introduction

Cloud computing enables users to utilize a wide range of services without needing to manage or worry about the underlying infrastructure [1, 21]. It has become an essential solution in various fields, including Information Technology, by providing equipment and support that enable developers to focus on development without worrying about infrastructure challenges [7]. Cloud providers are responsible for providing users with resources that meet the quality of service standards and ensure compliance with the conditions outlined in the Service Level Agreement (SLA). In a modern cloud computing environment, it is crucial to predict future workload patterns for efficient and proactive resource allocation and system optimization [2]. However, the high dimensionality and high volume of monitoring data generated by cloud environments pose challenges for efficient preprocessing and prediction of future workloads [4]. Traditional methods often struggle to handle large-scale monitoring data, resulting in inaccurate predictions and high computational needs. To address this problem, a robust cloud workload prediction model is required that leverages data compression and accurate prediction. Predicting future workload within a cloud computing environment is an important task that directly affects resource allocation and system performance. An accurate forecast enables the timely provision of resources, prevents bottlenecks, and ensures the efficient use of resources [17]. In addition, predictive models have proven to improve and timely resource allocation in cloud companies with high demand and service level agreement (SLA) constraints [18]. Although deep learning techniques have shown success in this area, their application is often impeded by the significant effort required to preprocess the vast volumes of high-dimensional, noisy monitoring data generated by cloud systems. This data presents challenges related to computational complexity [5]. However, two major challenges affect effective workload forecasting:

1. The high-dimensional and noisy monitoring data produced by cloud environments increase preprocessing demands and hinder model generalization.
2. The computational overhead of deep learning models restricts scalability in large-scale, resource-constrained deployments.



This work is licensed under a Creative Commons Attribution 4.0 International License. *BDCAT '25, Nantes, France*

© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2286-8/25/12
<https://doi.org/10.1145/3773276.3774300>

Prior research has explored machine learning and deep learning methods, including LSTM and GRU models, as well as dimensionality reduction techniques such as PCA and autoencoders. While these approaches achieve good accuracy, they often treat compression and prediction as separate tasks, leaving unexplored the balance between data reduction, prediction quality, and computational efficiency.

Big data compression and dimensionality reduction have emerged as a significant research area in recent years [14], due to the increasing complexity and volume of data generated across diverse domains. Data compression techniques are essential to enhance storage efficiency and reduce computational demands in resource-constrained environments. To address the challenges in cloud workload data, this paper proposes a novel framework that integrates autoencoder-based workload compression with Long Short-Term Memory (LSTM) modeling to improve workload prediction and resource management. The framework utilizes autoencoders to reduce the dimensionality of monitoring data, thereby reducing the computational cost of training deep learning models and optimizing storage. This compression step is crucial for accurately processing large datasets, extracting meaningful representations, and training the LSTM model for precise workload forecasting.

1.1 Proposed method

In this paper, we address a gap in the literature by proposing a hybrid framework that combines autoencoders and LSTM (Long Short-Term Memory) networks for joint optimization of compression and forecasting. Our approach utilizes autoencoders to extract compact and noise-resistant representations of workloads, which are then used to train LSTM models for time-series predictions. In addition to assessing accuracy, we systematically evaluate the trade-off between compression ratio and predictive performance. We also quantify the computational advantages in terms of resource usage and training/inference times.

1.2 Contribution of this paper

- (1) A novel workload forecasting framework that integrates autoencoder-based compression with LSTM prediction.
- (2) A systematic evaluation of the trade-off between compression and accuracy, utilizing both synthetic and real-world datasets.
- (3) Evaluate the proposed framework for computational efficiency, including CPU usage, memory footprint, and execution time.
- (4) Demonstration of up to 60% storage savings and improvement in accuracy compared to baseline methods, confirming the framework's suitability for practical deployment.

1.3 Significance

The proposed framework has significant implications for improving the efficiency and performance of cloud computing systems. By enabling more accurate workload predictions while reducing the computational burden associated with processing massive monitoring data, the framework can facilitate proactive resource allocation, improve system scalability, and optimize resource utilization in

cloud environments. Ultimately, this research contributes to the advancement of cloud computing technologies by addressing critical challenges in workload prediction and resource management.

The remainder of this paper is organized as follows: Section 2 provides an overview of state-of-the-art research in this field. Sections 3 and 4 detail the proposed methodology and experimental setup, respectively. Finally, Sections 5 and 6 present the experimental evaluation and the conclusion.

2 Literature Review

The study of cloud workload prediction can be generally categorized into classical machine learning and deep learning approaches. First, we summarize the key findings and contributions from state-of-the-art machine learning-based cloud resource prediction studies, followed by an overview of deep learning approaches. Furthermore, we summarize dimensionality reduction and compression methods for cloud workloads.

2.1 Deep learning-based methods

Deep Neural Network (DNN) is introduced in [6] to classify workloads in a multi-virtual machine cloud environment. In [30] the authors have proposed an efficient supervised learning-based Deep Neural Network (esDNN) that combines DNN with a Gated Recurrent Unit (GRU) to tackle the high variability and high dimensional feature of cloud workloads. They have achieved improved performance over the baseline GRU. Moreover, the authors in [11] have proposed a Bidirectional Gated-Recurrent Unit (BiGRU) and attention mechanism to predict future workload. They utilize Discrete Wavelet Transformation (DWT) to decompose and extract patterns from the non-linear and non-stationary input data. Several LSTM-based approaches exist for workload prediction. In [19], Long Short-Term Memory (LSTM) has been introduced to capture the long-term dependencies of cloud workloads. On the other hand, [23] has proposed an LSTM to predict storage workload in the cloud. They further investigated the effect of hyperparameters in the LSTM model. Tang [26] proposed a two-dimensional LSTM network based on week-based dependence and weight parallelization. They achieved real-time performance with high accuracy. Furthermore, [13] has utilized a multi-layer Bidirectional Long Short-Term Memory (Bi-LSTM) to predict task and job failure in the cloud. In [24] the authors proposed Feature Enhanced Multi-Task cloud workloads (FEMT-LSTM) to predict the turning point of trend in cloud workloads. A bidirectional long short-term memory (BiLSTM) based on multivariate time series is proposed in [27]. The authors considered predicting multi-dimensional virtual machine resource parameters simultaneously to capture the relationship between application requirements and resources. The authors in [22] demonstrated that LSTM achieved higher accuracy in predicting future workload compared to DNN, Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN).

2.2 Dimensionality reduction and compression methods

Principal Component Analysis (PCA) [29] is a widely adopted technique for reducing the dimensionality of high-dimensional datasets.

Table 1: Comparison of Related Work on Cloud Workload Prediction

Study	Method	Datasets	Strengths	Limitations
Bhagtya et al. (2021) [6]	DNN for workload classification	Multi-VM synthetic	Handles diverse workloads	Ignores temporal dependencies, no compression
Dogani et al. (2023) [11]	BiGRU + DWT	Host load traces	Captures non-linear workload patterns	High preprocessing cost; no dimensionality reduction
Kumar et al. (2018) [19]	LSTM	Cloud datacenter traces	Models long-term dependencies	High computational cost; scalability not addressed
Ruan et al. (2023) [23]	LSTM for storage workload	Storage systems	Explores hyperparameter effects	Limited to storage workloads; no compression
Ruan et al. (2022) [24]	FEMT-LSTM (feature-enhanced multitask)	Cloud workloads	Predicts workload turning points	Complex model; high training cost
Tang (2019) [26]	2D parallel LSTM	Large-scale workloads	Real-time performance	Parallelization needed; resource-intensive
Ullah et al. (2023) [27]	BiLSTM for multivariate prediction	VM resource data	Captures multi-dimensional patterns	No compression; high complexity
Alqahtani (2023) [3]	Sparse Autoencoder + GRU	Cloud workloads	Reduces dimensionality; good accuracy	Focused on GRU; lacks compression–accuracy trade-off
Chen et al. (2020) [9]	Deep learning on high-dimensional workloads	Synthetic workloads	High accuracy on variable workloads	Computationally expensive
El Sayed et al. (2023) [12]	Deep AE for compression & anomaly detection	Telemetry data	Achieves high compression ratio	Focus on anomaly detection, not forecasting
This Work (Proposed)	Autoencoder + LSTM	Synthetic (Kubernetes), (Wikipedia)	Joint compression & prediction; systematic compression–accuracy trade-off; computational analysis	future work: diverse real-world traces

PCA works by projecting the original data onto principal components that are mutually orthogonal. With advancements in deep learning, autoencoders have emerged as powerful alternatives for dimensionality reduction, enabling the learning of compact data representations. A convolutional autoencoder-based feature extraction and compression approach is proposed to capture daily and seasonal variants for customer load analysis [25]. Moreover, in [9], a top sparse autoencoder (TSA) based method is utilized to extract representations from high-dimensional workload data. They achieved good accuracy on high-dimensional and highly variable workloads. Similarly, the authors in [3] utilized Sparse Auto-Encoders (SAE) and Gated Recurrent Units (GRU) to reduce dimensionality and predict cloud workload data. Furthermore, they enhanced the GRU model by incorporating a step-wise learning rate scheduler. In [20], the authors have proposed two power prediction models for data centers based on an Autoencoder (AE). A fine-grained model based on recursive AE and a coarse-grained model based on AE to encode extensive historical data and reduce dimensionality. Additionally, authors in [16] proposed a lightweight Inception-based AE Network to compress noisy time series data generated by the Internet of Things (IoT). They evaluated the methods based on compression and feature extraction capabilities. The authors in [12] proposed deep autoencoders for adaptive telemetry data compression and anomaly detection, demonstrating a compression ratio of up to 64% with less than 1% reconstruction error in their experiments.

As illustrated in Table 1, existing studies mainly focus on achieving high accuracy in predicting future cloud workloads. However, the high dimensionality of the data poses significant challenges,

particularly in balancing prediction accuracy with the computational demands of training deep learning models. Dimensionality reduction techniques have demonstrated the ability to extract essential representations from high-volume workloads. This enables the training of deep learning models with reduced computational requirements and improved accuracy. Additionally, traditional compression methods primarily address data reduction for transportation purposes. In this work, we integrate cloud workload compression and prediction to achieve an approach that delivers high accuracy, reduced computation, and lower storage requirements.

3 Methodology

This section outlines the methodology used to develop and evaluate the proposed framework for workload compression and prediction in cloud computing environments. The approach involves data compression using autoencoders [15] and workload prediction using Long Short-Term Memory (LSTM) [16] model. The methodology is divided into key components: data compression using an autoencoder, workload prediction using LSTM models, and evaluation metrics. The architecture is shown in Fig. 1. The proposed method is shown in Algorithm 2.

3.1 Data Compression and dimensionality reduction using Autoencoder

We employ autoencoders for data compression to address the challenge of processing massive monitoring data. Autoencoders are a neural network designed to learn efficient representations of input

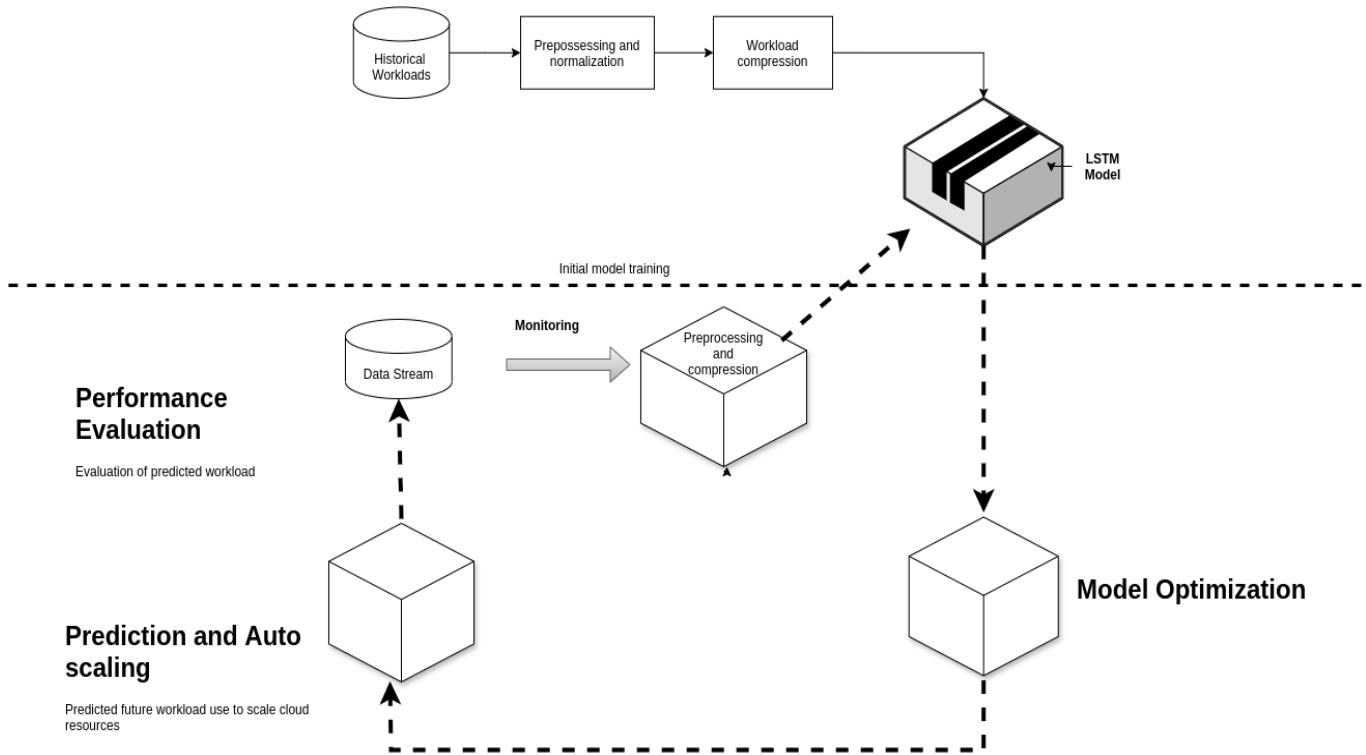


Figure 1: The architecture of the autoscaling mechanism in a cloud workload environment alongside the proposed workload prediction approach.

data by encoding it into a compressed form and then reconstructing it back to its original form [15].

3.1.1 Model architecture. The autoencoder architecture consists of an encoder, which compresses the input data, and a decoder, which attempts to reconstruct the original data from the compressed representation. The model is trained to minimize the reconstruction loss, ensuring that essential information is preserved during compression.

3.1.2 Training. The autoencoder is trained on the collected monitoring data, optimizing its parameters to achieve the desired level of data compression. During training, various compression ratios are performed. The whole training process of the autoencoder is illustrated in algorithm ??.

3.1.3 Latent dimension. This indicates the data reduction ratio by the encoder. Once trained, the autoencoder is used to compress the test data, and the data reduction ratio is calculated by comparing the compressed data size to the original data size. This metric is crucial in evaluating the efficacy of the autoencoder in reducing storage needs. The complete flow of compressing data with an autoencoder is illustrated in algorithm 3.

3.2 Workload Prediction using LSTM Models

With the compressed data obtained from the autoencoder, we proceed to the workload prediction phase using LSTM models. LSTM

[19] is a type of recurrent neural network (RNN) that is particularly well-suited for sequence prediction tasks due to its ability to capture long-term dependencies in time series data.

3.2.1 Model architecture. The LSTM model is designed with input layers corresponding to the compressed data features, hidden layers with LSTM units, and an output layer that generates the workload predictions.

3.2.2 Training and validation. The LSTM model is trained on the compressed training data. A validation set is used to tune hyperparameters such as the number of LSTM units, learning rate, and batch size to prevent overfitting and enhance the model’s generalization capabilities.

3.2.3 Prediction accuracy. The trained LSTM model is evaluated on the test set, and its prediction accuracy is measured against the ground truth workload observations. This assessment ensures that the model can reliably predict future workloads.

4 Experimental Setup

All experiments are performed on a machine with 32GB of RAM, an Intel Core i7-7700 3.6 GHz CPU with four cores on the Ubuntu Linux 20.04 operating system. We implement our LSTM and AE model using Keras [10].

Algorithm 1 Training an Autoencoder

Input: Training dataset $\mathcal{X} = \{x^{(i)} \mid i = 1, \dots, N\}$, encoder f_θ , decoder g_ϕ , learning rate η , number of epochs E ,

Output: Optimized parameters θ and ϕ

Initialize parameters θ, ϕ randomly

for epoch = 1, 2, ..., E **do**

for each $x \in \mathcal{X}$ **do**

Forward pass:

 Compute the latent representation: $z = f_\theta(x)$ (*Encoder*)

 Reconstruct the input: $\hat{x} = g_\phi(z)$ (*Decoder*)

 Compute reconstruction loss: $\mathcal{L} = \|x - \hat{x}\|^2$

Backpropagation:

 Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial z} \cdot \frac{\partial z}{\partial \theta}, \quad \frac{\partial \mathcal{L}}{\partial \phi} = \frac{\partial \mathcal{L}}{\partial \hat{x}} \cdot \frac{\partial \hat{x}}{\partial \phi}$$

Update parameters:

$$\theta \leftarrow \theta - \eta \frac{\partial \mathcal{L}}{\partial \theta}, \quad \phi \leftarrow \phi - \eta \frac{\partial \mathcal{L}}{\partial \phi}$$

end for

end for

4.1 Datasets

The proposed methods are evaluated using synthetic and real-world cloud workload datasets across all experiments. The synthetic dataset is generated using Kubernetes [8] and a custom load generator. Additionally, real-world workload datasets are employed to validate the effectiveness of the methods in practical applications. In this context, workload forecasting focuses on key performance metrics, specifically CPU utilization and memory utilization, to ensure relevance to real-world operational needs.

4.1.1 Synthetic workload. For this study, synthetic data was generated from a system deployed on a Kubernetes cluster, spanning a two-day observation period and yielding 172,800 records. This dataset includes diverse metrics such as latency, CPU usage, memory usage, network bandwidth, and storage utilization. These metrics are critical for the detailed analysis of workload patterns and system performance, enabling a comprehensive evaluation of the proposed methods.

4.1.2 Wiki dataset. The Wikipedia dataset used in this study represents 10% of HTTP requests recorded between September 19, 2007, and December 31, 2013 [28]. This dataset aggregates HTTP requests over one-hour intervals across all languages supported by Wikipedia. The dataset fields include an identifier (ID), timestamp, and the number of HTTP requests, providing a comprehensive basis for evaluating the proposed forecasting methods. The properties of the datasets are presented in Table 2.

Algorithm 2 Autoencoder + LSTM for Cloud Workload Prediction

Input: Historical monitoring data $\mathbf{X} \in \mathbb{R}^{n \times m}$, where n is the number of time steps and m the number of features.

Output: Predicted workload $\hat{\mathbf{Y}}$.

Step 1: Autoencoder Training

- Train an autoencoder model on \mathbf{X} .
- **Encoder:** Map input \mathbf{X} to a compressed representation $\mathbf{Z} \in \mathbb{R}^{n \times d}$, where $d \ll m$.
- **Decoder:** Reconstruct \mathbf{X} from \mathbf{Z} by minimizing the reconstruction loss.
- Save the trained encoder for later dimensionality reduction.

Step 2: Dimensionality Reduction

- Use the trained encoder to transform \mathbf{X} into compressed data \mathbf{Z} .

Step 3: LSTM Training

- Train an LSTM model using \mathbf{Z} as input data.
- **Input:** compressed historical data \mathbf{Z} .
- **Output:** future workload prediction $\hat{\mathbf{Y}}$.
- **Objective:** minimize the prediction error (e.g., mean squared error).

Step 4: Prediction

- Use the trained LSTM model to predict future workload $\hat{\mathbf{Y}}$ based on \mathbf{Z} .

Step 5: Evaluation

- Compare predictions $\hat{\mathbf{Y}}$ with the actual workload \mathbf{Y} to evaluate accuracy.
- Assess storage savings by comparing the sizes of \mathbf{X} and \mathbf{Z} .
- Measure computational efficiency for both training and inference.

End of Algorithm.

Algorithm 3 Compression of Historical Data Using a Pretrained Autoencoder

Input: Historical data $\mathbf{X} \in \mathbb{R}^{n \times m}$, where n is the number of time steps and m the number of features.

Output: Compressed data $\mathbf{Z} \in \mathbb{R}^{n \times d}$, where $d \ll m$.

Step 1: Load the Pretrained Autoencoder

- Load the encoder component of the pretrained autoencoder.
- Ensure the encoder maps $\mathbb{R}^m \rightarrow \mathbb{R}^d$ correctly.

Step 2: Preprocess the Input Data

- Handle missing data in \mathbf{X} (imputation or removal).
- Normalize or standardize \mathbf{X} according to the training configuration.

Step 3: Apply the Encoder to Compress Data

- Compute $\mathbf{Z} = f_{\text{enc}}(\mathbf{X})$, where f_{enc} is the encoder function.
- Verify that \mathbf{Z} has dimensions (n, d) .

Step 4: Save the Compressed Data

- Store \mathbf{Z} for further analysis or workload prediction tasks.
- Optionally calculate storage savings: ratio = $\text{size}(\mathbf{Z}) / \text{size}(\mathbf{X})$.

End of Algorithm.

Table 2: Wikipedia dataset properties

Number of Instances	Min	Min	Standard deviation	Mean
50710	6	445592	57842.3	80277.2

4.2 Evaluation metrics

- (1) Data reduction ratio: Measure the percentage of reduction achieved in the size of monitoring data after compression using autoencoders.
- (2) Prediction accuracy: Evaluate the accuracy of workload predictions generated by the LSTM model trained on compressed data compared to ground truth observations.
 - (a) Root Mean Square Error (RMSE)

$$\text{RMSE} = \sqrt{\left(\frac{1}{n}\right) \sum_{i=1}^n (y_i - x_i)^2}$$

where x_i and y_i denote actual values and predicted results, respectively, and n represents the number of samples

- (3) Computational efficiency: to analyze the computational resources required by the proposed model for training and inference processes, including CPU utilization, memory usage, and execution time.
 - (a) CPU % usage utilized on training and inference by the proposed model.
 - (b) Memory % utilized on training and inference by the proposed model.
 - (c) Model training time (s).
 - (d) Model inference time (s).

5 Evaluation

In this section, we thoroughly evaluate the effectiveness and efficiency of the proposed research framework using a series of evaluation metrics. The performance of the proposed framework is benchmarked against traditional workload prediction methods and the LSTM model. The comparison involves two key aspects:

- Prediction accuracy: The accuracy of predictions made by the baseline LSTM model is compared to the proposed method, providing a measure of the superiority or parity of the proposed approach.
- Computational efficiency: The computational efficiency of the framework, considering both data compression and prediction processes, is evaluated relative to the baseline methods. This comparison highlights the potential of the proposed framework to reduce computational overhead while maintaining or improving prediction accuracy.

5.1 Prediction Accuracy

The predictive accuracy of the proposed Long LSTM model, trained on compressed datasets, is evaluated against ground truth observations to validate its reliability. This assessment highlights the practical applicability of the framework in real-world scenarios. Table 3 presents a comparative analysis of the average prediction errors between the baseline LSTM and proposed models, utilizing

synthetic and real-world Wikipedia datasets. The results demonstrate a marked improvement in predictive accuracy for the proposed approach across both datasets, underscoring its efficacy in enhancing the performance of conventional LSTM-based prediction frameworks.

In addition to evaluating overall predictive accuracy, the performance of the proposed model was also assessed for its ability to predict long-term horizons. Table 4 summarizes the prediction errors for horizons of 1, 10, 20, 30, and 40 steps into the future, evaluated on both synthetic and Wikipedia datasets. In the synthetic dataset, the horizons correspond to 1-minute intervals, while in the Wikipedia dataset, the frequency is 1 hour. The table shows that the prediction error gradually increases as the horizon lengthens. However, despite this trend, the results demonstrate the reliability of the proposed model in accurately forecasting over extended horizons, thereby reinforcing its robustness and suitability for long-term predictive tasks.

Table 3: Prediction performance of baseline LSTM model and proposed model on synthetic and real-world datasets

Dataset	LSTM (rmse)	Proposed (rmse)
Synthetic	0.32	0.02
Wiki	0.13	0.06

Table 4: Prediction performance of various prediction horizons of the proposed model on synthetic and real-world dataset

Prediction horizon	Synthetic	Wiki
1	0.020	0.050
10	0.024	0.071
20	0.028	0.074
30	0.035	0.078
40	0.037	0.081

5.2 Experiment with Various Compression Ratios

The proposed research framework undergoes rigorous experimentation with different compression ratios, specifically 20%, 30%, 40%, and 50%. This experimentation aims to assess the trade-off between data compression and prediction accuracy across varying levels of data reduction. By systematically varying the compression ratios, we analyze how the compression level influences the LSTM model's predictive performance and the framework's overall efficacy. This study aims to evaluate how the LSTM model captures information from compressed data representations. This evaluation provides valuable insights into the optimal balance between data compression and prediction accuracy, enabling the identification of the most suitable compression ratio for practical deployment scenarios. Through this additional evaluation, we expand our understanding of the proposed framework's behavior under different compression settings, facilitating informed decision-making regarding the selection of an appropriate compression ratio tailored to

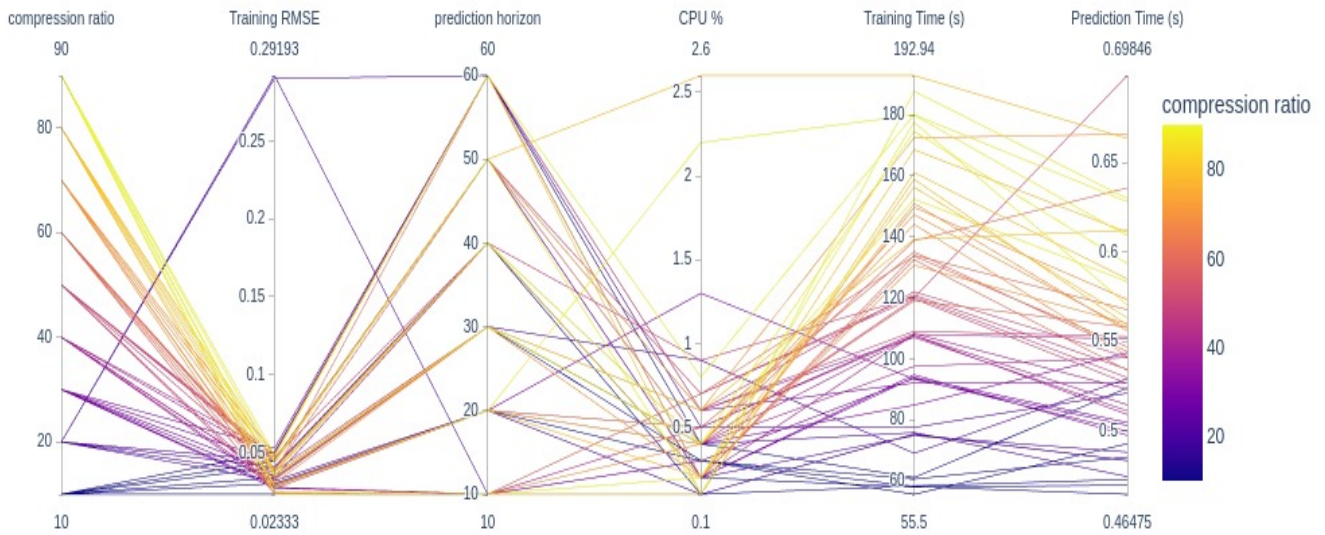


Figure 2: Proposed model performance on synthetic dataset

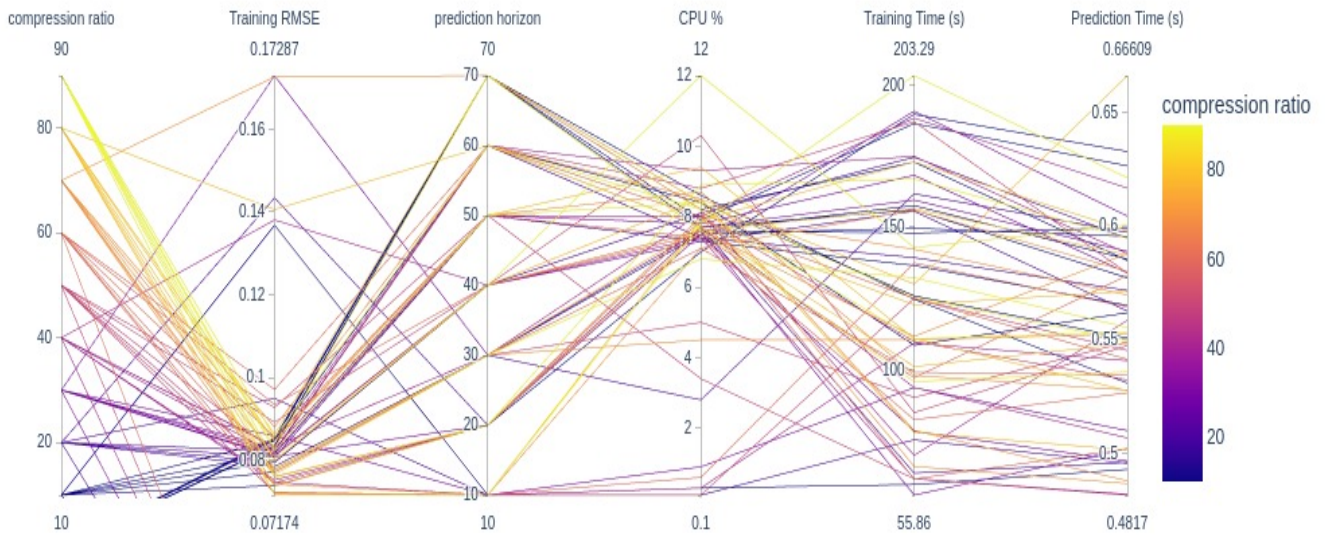


Figure 3: Proposed model performance on wiki dataset

specific application requirements. The key findings are summarized as follows.

- The LSTM model, trained with compressed representations from the encoder, achieved acceptable accuracy in all cases.

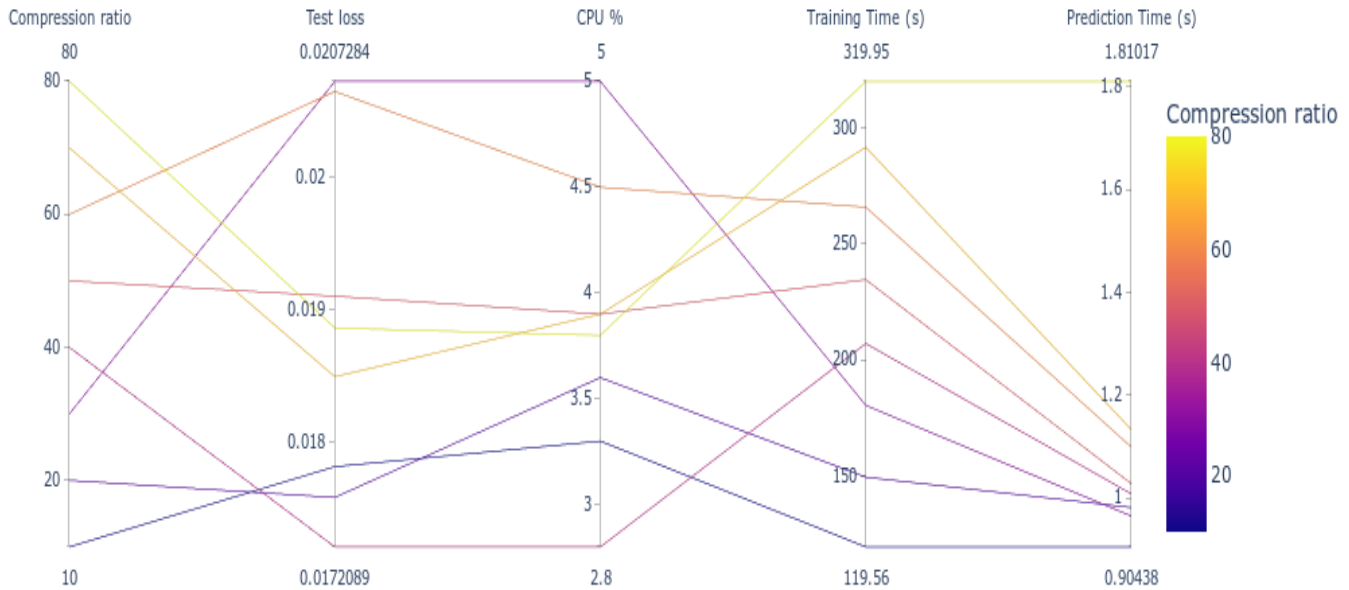


Figure 4: The computational performance of the proposed model across various compression ratios on synthetic dataset.

- The LSTM model, trained with compressed representations achieving up to 60% compression, produced the lowest prediction error in both synthetic and real-world workloads.
- Smaller latent dimensions, specifically 10%, 20%, and 30%, resulted in relatively higher prediction errors across both datasets. This trade-off can be adjusted based on specific requirements and the nature of the data being used.

5.2.1 Impact of Data Compression Ratio. In this section, we evaluate the model’s effectiveness in compressing data and examine how different levels of workload compression impact the LSTM prediction model. Additionally, we analyze the influence of the compression ratio on model performance, focusing on prediction accuracy, long-term forecasting capabilities, and the computational overhead. This includes examining CPU and memory utilization, as well as the model’s training and inference times. We look at both synthetic and real-world datasets.

Synthetic data: We conducted experiments using various compression ratios, specifically 10%, 20%, 30%, 40%, and up to 90%. As shown in Fig. 2, all compression ratios produced low prediction errors. However, in terms of computational overhead, higher compression ratios resulted in increased CPU usage compared to lower compression ratios. A similar trend was observed for both training and prediction times.

Real world data: Similarly, we examined the effect of different compression ratios on real-world workloads. As depicted in Fig.

3, higher compression ratios resulted in lower prediction errors compared to lower compression ratios. This indicates that highly compressed data provided the essential representation required for the LSTM model to capture the data’s characteristics effectively. Consistent with the results from synthetic data, we observed that higher compression ratios resulted in increased CPU usage, as well as longer training and prediction times.

We measure the efficacy of data compression using autoencoders by calculating the percentage reduction in the size of the monitoring data. This metric serves as a pivotal indicator of the model’s ability to condense large volumes of data while preserving crucial information.

5.3 Computational Assessment

In this evaluation, an analysis of computational resources required for both training and inference processes is conducted. This assessment encompasses CPU utilization percentage, memory usage percentage, model training time, and inference time. By inspecting the computational demands, we gain valuable insights into the scalability and resource efficiency of the proposed framework, facilitating informed decisions regarding its deployment in resource-constrained environments.

Figure 4 illustrates the computational requirements during the training and inference stages of the proposed method on synthetic data, with the prediction horizon set to 1. The analysis spans data compression levels ranging from 10% to 80%, highlighting the corresponding prediction performance, CPU usage, and computation

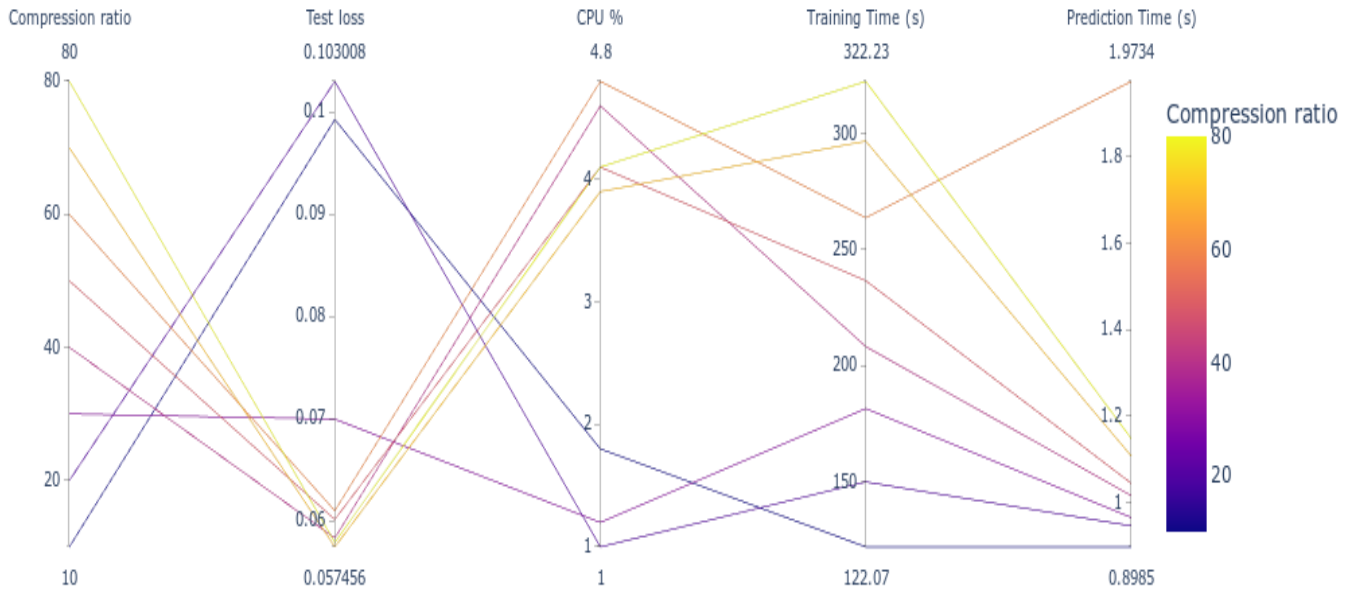


Figure 5: The computational performance of the proposed model across various compression ratios on Wiki dataset.

time. The results indicate that lower compression ratios require less CPU usage and shorter training and inference times, thereby reducing resource consumption. This is because a lower compression ratio, which corresponds to a larger latent space, requires more parameters and, consequently, consumes more computational power. However, the overall prediction loss remains relatively consistent across all compression ratios. Notably, smaller compression percentages resulted in lower CPU usage, while a clear distinction was observed in the time required for training and inference. Specifically, the 10% and 20% compression ratios exhibited shorter training and prediction times compared to higher compression ratios.

Similarly, the results in Figure 5 show computational performance evaluation on the Wiki dataset. The analysis spans data compression levels ranging from 10% to 80%, showing the corresponding prediction performance, CPU usage, and computation time. The results indicate that lower compression ratios demand less CPU usage and shorter training and inference times, thereby reducing resource requirements. However, the model achieves higher accuracy with greater compression. Specifically, compression levels of 10% and 20% yielded a test loss of approximately 0.1, while higher compression ratios achieved a lower prediction error, at around 0.06. This highlights a trade-off between computational efficiency and prediction accuracy, necessitating a balance based on application requirements. This is because a higher compression ratio forces the model to capture only the most essential features of the data. It removes redundancy and less relevant information, focusing on the

underlying structure of the data that is most important for making predictions.

Finally, through the extensive evaluation of the computational requirement, we gained insight into not only the effect of workload compression on prediction accuracy but also how each setting chosen affects the system's overall performance and its validation in addressing the identified research objectives.

6 Conclusion

The study presents a novel cloud workload compression and prediction frameworks that combine an autoencoder with LSTM models. The framework aims to enhance efficiency by reducing storage and computational overhead while maintaining high accuracy in workload prediction. By using an autoencoder, the framework effectively reduced the dimensionality of the workload while preserving the necessary features. The LSTM model trained with the compressed representation of the original workload achieved up to a 30% improvement in prediction accuracy.

In addition to improving the predictive performance, the compression level achieved through the autoencoder contributes to substantial storage savings by allowing compressed historical data instead of original, high-dimensional monitoring data. A comprehensive evaluation demonstrated the framework's effectiveness in data reduction, prediction accuracy, and calculation efficiency. Comparison with traditional methods emphasized its ability to handle high-volume cloud workloads and improve prediction accuracy.

The assessment further confirmed its feasibility for a real-world cloud environment, achieving a significant reduction in training and inference times, alongside storage optimization. Thus, it is a crucial step to a robust and scalable solution for the challenges of workload prediction in a resource-intensive cloud system. Through a comprehensive evaluation of the computational requirements needed to train models to predict future workload, valuable insights were gained into the significance of workload compression on prediction accuracy and overall system performance. Notably, a higher compression ratio resulted in more accurate prediction models, albeit with a slight increase in computational demand. Moreover, the framework allows for up to a 60% reduction in storage requirements for workload data. In a resource-constrained environment, a trade-off can be made between storage, computational requirements, and workload predictive accuracy, depending on the specific use case. Future directions include exploring more robust methods with better compression techniques for cloud workloads.

Acknowledgments

Financial support has been provided in part by the Knut and Alice Wallenberg Foundation under grant KAW 2019.0352 and by the eSENCE Programme under the Swedish Government's Strategic Research Initiative.

References

- [1] Mahfooz Alam, Suhel Mustajab, Mohammad Shahid, and Faisal Ahmad. 2023. Cloud computing: architecture, vision, challenges, opportunities, and emerging trends. In *2023 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*. IEEE, 829–834.
- [2] Tanweer Alam. 2020. Cloud Computing and its role in the Information Technology. *IAIC Transactions on Sustainable Digital Innovation (ITSDI)* 1, 2 (2020), 108–115.
- [3] Dalal Alqahtani. 2023. Leveraging Sparse Auto-Encoding and Dynamic Learning Rate for Efficient Cloud Workloads Prediction. *IEEE Access* 11 (2023), 64586–64599. doi:10.1109/ACCESS.2023.3289884
- [4] Fadi Alzhour, Suhil Bani Melhem, Anjali Agarwal, Mustafa Daraghme, Yan Liu, and Sadeq Younis. 2020. Dynamic resource management for cloud spot markets. *IEEE Access* 8 (2020), 122838–122847.
- [5] Maryam Amiri and Leyli Mohammad-Khanli. 2017. Survey on prediction models of applications for resources provisioning in cloud. *Journal of Network and Computer Applications* 82 (2017), 93–113.
- [6] Paras Bhagtya, S Raghavan, and K Chandrasekran. 2021. Workload classification in multi-vm cloud environment using deep neural network model. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*. 79–82.
- [7] Irena Bojanova, Jia Zhang, and Jeffrey Voas. 2013. Cloud computing. *IT Professional* 15, 2 (2013), 12–14.
- [8] Eric A. Brewer. 2015. Kubernetes and the path to cloud native. In *Proceedings of the Sixth ACM Symposium on Cloud Computing (Kohala Coast, Hawaii) (SoCC '15)*. Association for Computing Machinery, New York, NY, USA, 167. doi:10.1145/2806777.2809955
- [9] Zheyi Chen, Jia Hu, Geyong Min, Albert Y. Zomaya, and Tarek El-Ghazawi. 2020. Towards Accurate Prediction for High-Dimensional and Highly-Volatile Cloud Workloads with Deep Learning. *IEEE Transactions on Parallel and Distributed Systems* 31, 4 (2020), 923–934. doi:10.1109/TPDS.2019.2953745
- [10] François Chollet. 2015. keras. <https://github.com/fchollet/keras>.
- [11] Javad Dogani, Farshad Khunjush, and Mehdi Seydali. 2023. Host load prediction in cloud computing with discrete wavelet transformation (dwt) and bidirectional gated recurrent unit (bigru) network. *Computer Communications* 198 (2023), 157–174.
- [12] Ahmad El Sayed, Marc Ruiz, Hassan Harb, and Luis Velasco Esteban. 2023. Deep Learning-Based Adaptive Compression and Anomaly Detection for Smart B5G Use Cases Operation. *Sensors* 23 (01 2023), 1043. doi:10.3390/s23021043
- [13] Jiechao Gao, Haoyu Wang, and Haiying Shen. 2020. Task failure prediction in cloud data centers using deep learning. *IEEE transactions on services computing* 15, 3 (2020), 1411–1422.
- [14] Muhammad Habib ur Rehman, Chee Sun Liew, Assad Abbas, Prem Prakash Jayaraman, Samee Khan, and Teh Wah. 2016. Big Data Reduction Methods: A Survey. *Data Science and Engineering* 1 (12 2016). doi:10.1007/s41019-016-0022-0
- [15] Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *science* 313, 5786 (2006), 504–507.
- [16] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural computation* 9 (12 1997), 1735–80. doi:10.1162/neco.1997.9.8.1735
- [17] Tahseen Khan, Wenhong Tian, Guangyao Zhou, Shashikant Ilager, Mingming Gong, and Rajkumar Buyya. 2022. Machine learning (ML)-centric resource management in cloud computing: A review and future directions. *Journal of Network and Computer Applications* 204 (2022), 103405.
- [18] In Kee Kim, Jinho Hwang, Wei Wang, and Marty Humphrey. 2020. Guaranteeing performance SLAs of cloud applications under resource storms. *IEEE Transactions on Cloud Computing* 10, 2 (2020), 1329–1343.
- [19] Jitendra Kumar, Rimsha Goomer, and Ashutosh Kumar Singh. 2018. Long short term memory recurrent neural network (LSTM-RNN) based workload forecasting model for cloud datacenters. *Procedia computer science* 125 (2018), 676–682.
- [20] Yuanlong Li, Han Hu, Yonggang Wen, and Jun Zhang. 2016. Learning-based power prediction for data centre operations via deep neural networks. In *Proceedings of the 5th International Workshop on Energy Efficient Data Centres*. 1–10.
- [21] P Mell. 2011. The NIST Definition of Cloud Computing. *Recommendations of the National Institute of Standards and Technology* (2011).
- [22] Yashwant Singh Patel and Rajiv Misra. 2018. Performance comparison of deep VM workload prediction approaches for cloud. In *Progress in Computing, Analytics and Networking: Proceedings of ICCAN 2017*. Springer, 149–160.
- [23] Li Ruan, Yu Bai, Shaoning Li, Shuibing He, and Limin Xiao. 2023. Workload time series prediction in storage systems: a deep learning based approach. *Cluster Computing* (2023), 1–11.
- [24] Li Ruan, Yu Bai, Shaoning Li, Jiaxun Lv, Tianyuan Zhang, Limin Xiao, Haiguang Fang, Chunhao Wang, and Yunzhi Xue. 2022. Cloud workload turning points prediction via cloud feature-enhanced deep learning. *IEEE Transactions on Cloud Computing* 11, 2 (2022), 1719–1732.
- [25] Seunghyoung Ryu, Hyungeun Choi, Hyoseop Lee, and Hongseok Kim. 2019. Convolutional autoencoder based feature extraction and clustering for customer load analysis. *IEEE Transactions on Power Systems* 35, 2 (2019), 1048–1060.
- [26] Xiaoyong Tang. 2019. Large-scale computing systems workload prediction using parallel improved LSTM neural network. *IEEE Access* 7 (2019), 40525–40533.
- [27] Farman Ullah, Muhammad Bilal, and Su-Kyung Yoon. 2023. Intelligent time-series forecasting framework for non-linear dynamic workload and resource prediction in cloud. *Computer Networks* 225 (2023), 109653.
- [28] Guido Urdaneta, Guillaume Pierre, and Maarten Van Steen. 2009. Wikipedia workload analysis for decentralized hosting. *Computer Networks* 53, 11 (2009), 1830–1845.
- [29] Svante Wold, Kim Esbensen, and Paul Geladi. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems* 2, 1-3 (1987), 37–52.
- [30] Minxian Xu, Chenghao Song, Huaming Wu, Sukhpal Singh Gill, Kejiang Ye, and Chengzhong Xu. 2022. esDNN: deep neural network based multivariate workload prediction in cloud computing environments. *ACM Transactions on Internet Technology (TOIT)* 22, 3 (2022), 1–24.