An Adaptive Deep Learning Framework with Enhanced Attention for Precise Load Forecasting in Cloud Computing Environments

Kusuma Nidadavolu
Department of Computer Science and Systems Engineering, School of Computer Science and Engineering, GITAM (Deemed to be University), Hyderabad, India
knidadav@gitam.edu (corresponding author)

Somasekhar Giddaluru
Department of Computer Science and Systems Engineering, School of Computer Science and Engineering, GITAM (Deemed to be University), Hyderabad, India
sgiddalu@gitam.edu

Received: 10 January 2026 | Revised: 31 January 2026, 7 February 2026, and 15 February 2026 | Accepted: 16 February 2026

 DOI: https://doi.org/10.48084/etasr.17454

ABSTRACT

With the rapid growth of cloud computing services, accurate load forecasting and balancing have become critical for efficient resource utilization and system responsiveness. This paper proposes an adaptive deep learning model that integrates Convolutional Neural Networks (CNN), Bidirectional Long Short-Term Memory (Bi-LSTM), and an attention mechanism for precise multi-step load prediction. The CNN extracts local spatial features, the Bi-LSTM captures bidirectional temporal dependencies, and the attention mechanism dynamically focuses on the most relevant input segments, improving sensitivity to sudden load variations and anomalies. Evaluated on multi-step horizons (1–15 steps ahead) using a real-world IoT gateway and the Google Cluster Trace datasets, the model achieves average values of MSE at 0.0023, MAE at 0.035, and R^2 at 0.92 across horizons. These results indicate superior accuracy and explained variance compared to state-of-the-art baselines, making the approach highly suitable for cloud providers aiming to optimize resource provisioning and reduce operational costs.

Keywords-attention mechanism; convolutional neural networks; cloud computing; deep learning; resource provision; time-series data

I. INTRODUCTION

Cloud computing has revolutionized the administration of digital resources by offering on-demand access to computing resources without active management [1]. Its history traces back to the 1960s with time-sharing concepts, while the 1990s growth in telecommunication bandwidth laid the foundation for the development of cloud computing technology as a reasonable commercial technology. The early 2000's witnessed the emergence of the concept of using computing resources with the help of internet connectivity. Load prediction in cloud computing involves forecasting resource requirements to ensure that they are appropriately allocated so that there are no shortages or surpluses.

Traditional statistical methods, such as ARIMA, remain effective in scenarios with relatively stable or seasonal patterns. However, they often struggle with the high dynamic, nonlinear, and burst workloads of modern cloud environments. Machine learning and deep learning models, such as Support Vector Machines, Neural Networks, and LSTMs, have transformed the load prediction process based on big data [2]. Cloud systems have dynamic workloads due to fast and frequent changes. To balance these variations and handle the various features of cloud resources and cost-effectiveness, powerful algorithms and real-time data processing are necessary [3]. The integration of AI, IoT, and edge computing technologies is crucial in load prediction and balance schemes [4, 5]. Load prediction and balancing techniques are vital in making cloud computing more efficient and effective, as they ensure that it remains a scalable, reliable, and affordable solution to a number of computing needs. This study presents a new method of load prediction and balancing with the following objectives:

- Design a model for accurate load demand forecasting, improving MSE and MAE metrics.
- Integrate CNN with Bi-LSTM and an attention mechanism to enhance load fluctuation sensitivity.
- Ensure optimal computational resource utilization and reduce CPU/memory variances.
- Employ attention-based feature prioritization for anomaly identification with a lower false-positive rate.

II. RELATED WORK

Early load balancing relied on rule-based approaches such as round robin and random allocation [6-8], which lacked adaptability to dynamic cloud workloads. Statistical methods, including ARIMA models, emerged to forecast future loads using historical patterns [9, 10]. Machine learning algorithms (e.g., SVMs and Decision Trees) improved prediction capabilities [11-13], while deep learning models, particularly LTSMs, demonstrated exceptional efficacy in processing large data volumes and acquiring complex patterns [14, 15]. A CNN-GRU hybrid model was effective with noisy and missing data in real-life hourly loads, performing better than single models in terms of its RMSE and MAPE [16]. Recent trends focus on hybrid approaches that combine multiple algorithms and federated learning for privacy-preserving scenarios.

Despite these advances, existing models face limitations including domain-specific applicability (e.g., energy sector only), insufficient fine-grained temporal adaptation, lack of federated/decentralized support, and limited fault tolerance. Table I summarizes key gaps in state-of-the-art approaches compared to the proposed model, which addresses these limitations by integrating CNN-Bi-LSTM with attention mechanisms, edge-cloud coordination, and adaptive learning capabilities. This proposed CNN-Bi-LSTM model uses an attention mechanism to improve contextual understanding, non-handling, dynamic anomaly detection, and programmed feature learning of high-dimensional data to address these problems.

TABLE I. RESEARCH GAPS IN EXISTING SOTA MODELS VS. THE PROPOSED MODEL

Ref.	Technique	Identified gaps	Advantages of the proposed model
[2]	DL-based electricity forecasting	Energy sector focus; lacks dynamic scaling and adaptation to bursts	Generalizes to the cloud with adaptive workload handling
[6]	ML+ Optimization	Probabilistic; Lacks temporal adaptation; No coverage of load forecasting	CNN-LSTM for spatio-temporal dynamics
[8]	CNN-BiLSTM	Low-carbon focus; No fault tolerance	Trust-aware; Fault-tolerant with explainability
[17]	Real-time prediction	Lacks federated settings	Includes federated learning and trust estimation

III. MATERIALS AND METHODS

The proposed system integrates edge and cloud computing layers for efficient IoT data processing, as shown in Figure 1. The IoT layer collects device data, edge devices perform basic processing, edge data centers perform advanced tasks, and the cloud provides high-level computational resources for storage, analytics, and decision-making. The updated model parameters learned in the cloud are periodically synchronized with the edge nodes, ensuring coordinated and adaptive forecasting while reducing network traffic and latency.

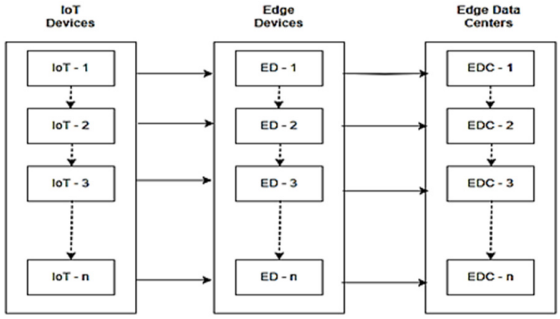

Fig. 1. Overview of the proposed IoT network integrated with cloud computing.

A comprehensive mathematical model is developed for the proposed CNN-Bi-LSTM with the attention mechanism. The CNN first extracts the local spatial features from raw time series inputs, providing a denoised informative sequence to Bi-LSTM for long-range bidirectional temporal modeling. Let $X \in R^{T \times F}$ represent an input matrix where T is the number of time steps and F is the number of features. $W_c \in R^{K \times F}$ represents the weighted matrix of the CNN, where K is the kernel size. $b_c \in R$ is the bias term in the CNN, and $C \in R^{T-K+1}$ is the convolved feature after applying the CNN filter. The convolution operation in the CNN layer can be represented as:

$$C_t = ReLU\left(\sum_{i=0}^{K-1}\sum_{j=0}^{F-1} W_c[i,j]\ .\ X[t+i,j] + b_c\right)$$
$$\text{for } t = 0, 1, \ldots, T-K \quad (1)$$

where $ReLU$ is the rectified linear unit activation function defined as:

$$ReLU(x) = \max(0, x) \quad (2)$$

In the BiLSTM, for each LSTM unit, W_f, W_i, W_o, and W_c are the weighted matrices for the forget gate, input gate, output gate, and cell state, respectively, in both directions. U_f, U_i, U_o, and U_c are the recurrent weight matrices, b_f, b_i, b_o, and b_c are the bias terms. At any time instant t, the forward LSTM can be represented as :

$$f_t = \sigma\left(W_f C_t + U_f h_{t-1} + b_f\right) \quad (3)$$
$$i_t = \sigma(W_i C_t + U_i h_{t-1} + b_i) \quad (4)$$
$$o_t = \sigma(W_o C_t + U_o h_{t-1} + b_o) \quad (5)$$
$$\tilde{C_t} = tanh(W_c C_t + U_c h_{t-1} + b_c) \quad (6)$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C_t} \quad (7)$$
$$h_t = o_t \odot \tanh(C_t) \quad (8)$$

Here, σ represents the sigmoid function, and $\odot$ represents element-wise multiplication. The above sequence is modeled using a Bi-LSTM network as

$$h_t C_t = LSTM\left(X_{conv}(t), \vec{h}_{t-1}, \vec{C}_{t-1}\right) \quad (9)$$

where $X_{conv}(t)$ is the input to the LSTM at time t, which is derived from the feature map of the CNN. The attention mechanism in the CNN and Bi-LSTM model is used for

processing time series data. By focusing on different features of the input data, the model understands the significance of each segment and then enables better decision-making. This approach is particularly useful in domains where understanding the model's reasoning is crucial. The attention mechanism also enhances the model's performance on tasks involving sequences, particularly in noisy or irrelevant datasets. Let $H \in R^{T\times 2D}$ represent the concatenated hidden states from the BiLSTM, where D is the hidden layer size. Let W_a, U_a, and V_a represent the weighing matrices for the attention mechanisms and b_a, c_a be the bias terms. The attention mechanism is defined as

$$e_t = \tanh(W_a H_t + U_a h_{t-1} + b_a) \quad (10)$$

$$\alpha_t = \frac{exp(v_a^T e_t + c_a)}{\sum_{t'} exp(v_a^T e_{t'} + c_a)} \quad (11)$$

Here, α_t is the attention weight for time step t. This aggregation of attention weights leads to the final attention-based representation given by :

$$A = \sum_{t=1}^{T} \alpha_t H_t \quad (12)$$

The attention mechanism of the proposed hybrid model assigns weights α_k to each Bi-LSTM hidden state h_k and calculates intermediate scores that are normalized using Softmax to generate a context vector that emphasizes the most informative time steps. This dynamic weighting not only represents sudden changes in sequences necessary to detect anomalies but also provides context vectors to the prediction layer to ensure better accuracy and interpretability. The predictions obtained can be used to allocate resources and provide load balancing in a cloud computing setup. Gaussian residual analysis identifies the anomalies in the time series by examining the deviations in the time series, and the errors give a glimpse into the model performance and error patterns. To reduce the effect of overfitting and improve accuracy, dropout regularization with tuned rates of [0.01-0.3] was used after the CNN blocks, the Bi-LSTM layer, and before the final dense prediction layer. Figure 2 illustrates the anomaly detection process.

The residual at each time step t is given by

$$e_t = y_t - \hat{y}_t \quad (13)$$

Standard deviation σ_r provides a measure of the variability in the residuals, given by:

$$\sigma_r = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(e_t - \mu_e)^2} \quad (14)$$

Depending upon the specific domains and applications, the sensitivity parameter k adjusts the severity of the anomaly detection:

$$\sigma_r \times k = \theta \quad (15)$$

A sample at any instant of time t is flagged as an anomaly if:

$$if\ (|e_t|) > \theta \quad (16)$$

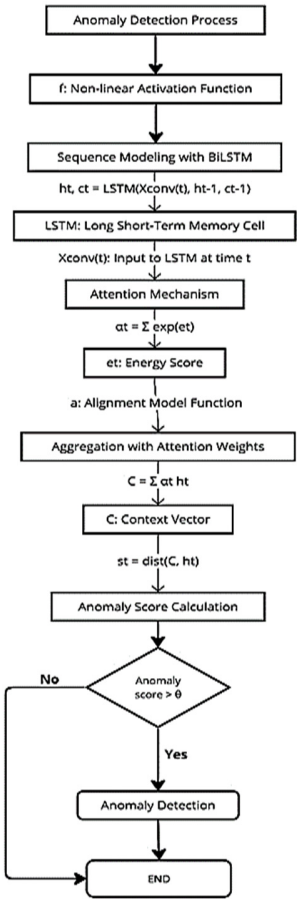

Fig. 2. Anomaly detection using residual analysis in the proposed model.

The following performance metrics were used to compare the proposed model with other existing models:

- Average CPU Utilization is the proportion of CPU capacity actively used over a given period, indicating how efficiently the processing power is being utilized by the system.

$$Avg\ CPU\ Utilization = \frac{1}{n}\sum_{i=1}^{n}\frac{CPU\ usage\ on\ node\ i}{Total\ CPU\ capacity\ on\ node\ i} \times 100 \quad (17)$$

- Average Memory utilization is the proportion of available memory that is consistently in use over a given period, reflecting the efficiency of memory resource management.

$$Avg\ Memory\ Utilization = \frac{1}{n}\sum_{i=1}^{n}\frac{Memory\ usage\ on\ node\ i}{Total\ Memory\ capacity\ on\ node\ i} \times 100 \quad (18)$$

- CPU load variance (var_{CPU}) measures how evenly the computational load is distributed across the CPU over time; lower variance indicates solid load balancing and resource allocation.

$$var_{CPU} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{CPU\ usage\ on\ node\ i}{Total\ CPU\ capacity\ on\ node\ i} - Avg\ CPU\ Utilization\right)^2 \quad (19)$$

- Throughput per node in the network is given by:

$$Throughput\ per\ node\ = \frac{Number\ of\ tasks\ completed\ on\ node\ i}{Total\ time} \quad (20)$$

- Response time is the time it takes for a system to process a request or task and return the result, with lower response times indicating faster and more efficient task control.

$$Response\ time\ per\ task\ = End\ time\ of\ task\ -\ Start\ time\ of\ task \quad (21)$$

- Node's Queue length is given by:

$$Queue\ length\ =\ Number\ of\ tasks\ waiting\ on\ node\ I \quad (22)$$

- Migration overhead is given by:

$$Migration\ overhead\ =\ \frac{Time\ spent\ on\ migration\ activities}{Total\ system\ time} \quad (23)$$

- Total energy consumption in the network is given by:

$$Total\ energy\ consumption\ = \sum_{i=1}^{n} Energy\ consumed\ by\ node\ i \quad (24)$$

IV. RESULTS

A. Model Evaluation

Performance was evaluated using two datasets. Simulations were conducted on a Dell XPS 15 laptop (Intel Core i7-10750H @2.6 GHz, 16 GB DDR4 RAM, 512 GB SSD, Windows 10 Pro) using Python 3.8.5, TensorFlow 2.3.0, Keras 2.4.3, Pandas 1.1.3, Numpy 1.19.2, Matplotlib 3.3.1, and Seaborn 0.11.0. The first dataset comprised 8622 samples of a 30-day time series of a production IoT gateway, having a sampling period of 5 minutes. Samples are 11-dimensional vectors of features with CPU cores, provisioned/used CPU (MHz %), memory (KB), disk I/O (KB/s), and network I/O(KB/s), reflecting short-term variations and long-term patterns. The model was also tested on Google Cluster Trace v.2.1 [18], a publicly available dataset of 12,500 machines, 2 days long, with 500,000 multivariate records condensed to 5 minutes.

A comprehensive grid search over all major hyperparameters was conducted for hyperparameter tuning, using validation-set MSE as the selection criterion. Learning rates of $\eta \in \{10^{-4}, 5\times10^{-4}, 10^{-3}\}$. and batch sizes of {32, 64, 128} were tested along with CNN filter counts {16, 32, 64}, kernel sizes {3×3, 5×5}, and pooling windows {2×2}. LSTM units varied across {50, 100, 150}, dropout rates {0.3, 0.5}, activation functions, and attention types. All training was performed with the Adam optimizer to minimize MSE and employing early stopping $patience =$ 10 on validation loss. Inputs were normalized using Min-Max scaling, and anomalies were flagged above the 95th percentile. The learning rate of 0.001, batch size = 64, 25 epochs, 32 CNN filters, 3×3 kernels, 2×2 pooling, 100 LSTM units, 0.5 dropout, and scaled dot-product attention yielded the lowest validation MSE, and therefore, was used for all reported experiments.

Data was split chronologically (70% training, 15% validation, 15% testing) to prevent leakage. Features were Min-Max normalized. The hybrid CNN-Bi-LSTM-Attention model was trained with 60-minute CNN patches and ±120-minute Bi-LSTM context, ensuring robust generalization to dynamic cloud workloads. Figure 3 compares actual and predicted CPU consumption over various samples. The red dots indicate the anomalies highlighted. It is clear that the proposed model tracks performance fairly accurately and in line with the actual data. Anomaly detection highlights unusual behavior. The model handles most spikes and is accurate in capturing sudden spikes or drops.

Table II compares the proposed model with state-of-the-art approaches; the model achieves an MSE of 0.0023, MAE of 0.035, and R^2 of 0.92, demonstrating superior capture of spatiotemporal dependencies. In load balancing, these metrics directly enhance resource allocation; lower MSE minimizes extreme over- or under-provisioning events and wasted capacity, reduced MAE enables tighter capacity matching and higher steady-state utilization, and higher R^2 ensures more reliable and consistent forecasts for stable provisioning decisions.

TABLE II. COMPARISON OF MODEL PERFORMANCE ON THE DATASET

MODEL	MSE	MAE	R^2
Proposed CNN-Bi-LSTM-Attention	0.0023	0.035	0.92
Transformer Forecaster	0.0031	0.042	0.88
GRU	0.0035	0.045	0.85
ARIMA+XGBoost	0.0042	0.052	0.8

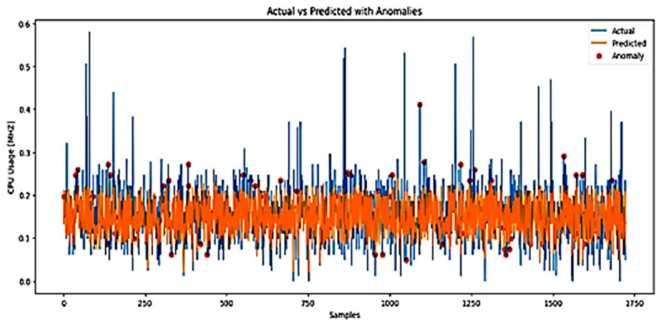

Fig. 3. Anomaly detection during CPU consumption slots in the proposed model.

Table III summarizes the superiority of the proposed model in predicting and balancing cloud computing demand. Using the reduced variance of the predictions of the proposed model, the safety margin (2σ) can be reduced from 0.1114 to 0.0959. This decreases the average provisioned capacity from 0.6114 to 0.5959 normalized units. This corresponds to a 2.5% drop in the total allocation. The observed decrease can further reduce the hourly provisioning cost, and a 13.9% reduction in overprovisioning is calculated as a relative decrease in excess provisioned capacity. Such empirical results show how resource utilization accuracy, throughput, and cost-effectiveness of response time are improved. This paradigm offers an important breakthrough in addressing the complexity and needs of modern cloud infrastructures.

TABLE III. PROVISIONING COST BY FORECASTING MODEL

MODEL	Safety margin (2σ)	Avg. Provisioned (Units)
Baseline Transformer	0.1114	0.6114
Proposed CNN +Bi-LSTM+Attention	0.0959	0.5959
GRU	0.1183	0.6183
ARIMA+XGBoost	0.1296	0.6296

B. System Analysis and Discussion

1) Computational Cost

Table IV summarizes a comparative analysis in terms of computational cost and resource usage of various models. Both the Transformer forecaster and the GRU model require more computation time and memory. The ARIMA+XGBoost ensemble is lighter but less accurate. All models ran on the same Dell XPS 15 (Intel i7-10750H, 6 cores/12 threads @2.6 GHz, 16 GB RAM), using only CPU (GPU < 5 % utilization). This confirms that the proposed model is computationally effective.

TABLE IV. COMPUTATIONAL RESOURCE PROFILE FOR FORECASTING MODELS ON A DELL XPS 15 LAPTOP

Model	Total training time	Peak CPU Usage	Peak GPU usage	Peak memory usage
Baseline Transformer	3.2 hours	95 %	< 5 %	14.1 GB
CNN-Bi-LSTM-Attention (proposed)	2.1 hours	92 %	< 5 %	12.3 GB
GRU	2.5 hours	90 %	< 5 %	11.8 GB
ARIMA+XGBoost	1.2 hours	70 %	< 5 %	8.4 GB

2) Error Analysis

Table V summarizes the incremental improvement in anomaly detection. The evaluation is performed on a fixed self-collected test set of 8,622 samples consisting of 8,450 normal instances and 172 labeled anomalies. The Bi-LSTM-only baseline correctly flags 132 out of 172 true anomalies and misses 40, along with 50 false alarms. When the CNN front end was added, true positives increased to 140, misses decreased to 32, and false positives were reduced to 45, highlighting its ability to extract discriminative spatial features. The addition of the attention mechanism yielded the largest improvement, detecting 162 anomalies with only 10 misses, while false alarms dropped to 15. True negatives in this case increased to 8,435. Each part increases recall and precision, resulting in a reliable, low false-alarm system appropriate for practical IoT applications. Cross-dataset tests on Google Cluster Trace [18] and synthetic anomaly injection (Gaussian noise+spikes) confirm strong generalization (recall > 0.90, low false positives), allaying worries about overfitting to the IoT gateway dataset despite the extremely high recall (162/172).

TABLE V. CONFUSION MATRIX FOR ANOMALY DETECTION

Model	TP	FN	FP	TN
Bi-LSTM only (baseline)	132	40	50	8,400
+CNN front end (CNN-Bi-LSTM)	140	32	45	8,405
+Attention (full CNN-Bi-LSTM-Attention)	162	10	15	8,435

3) Ablation Study

Table VI presents ablation experiments starting from a Bi-LSTM-only baseline (MSE 0.0035, MAE 0.045). Adding the CNN front end reduces MSE by 20.0% and MAE by 13.3%, demonstrating the value of localized cross-sensor feature learning. Incorporating the attention layer on top of the CNN-Bi-LSTM yields further gains—34.3% lower MSE and 22.2% lower MAE versus the baseline. This reflects that dynamic weighting of time steps is critical for capturing key anomaly and load-shift events.

TABLE VI. ABLATION STUDY QUANTIFYING THE IMPACT OF ADDING CNN FEATURE EXTRACTION AND ATTENTION

Model variant	MSE	MAE	ΔMSE (%)	ΔMAE (%)
BiLSTM only (baseline)	0.0035	0.045	0.0	0.0
+ CNN front end (CNN–BiLSTM)	0.0028	0.039	20.0	13.3
+ Attention (full CNN–BiLSTM–Attn)	0.0023	0.035	34.3	22.2

V. CONCLUSION

Cloud computing efficiency increases significantly through advanced load prediction and balancing techniques that optimize resource allocation, enhance performance, and lower operational costs. The proposed hybrid model combines CNN for spatial feature extraction and short-term dependency capture, Bi-LSTM for comprehensive bidirectional modeling of complex temporal relationships (unlike unidirectional LSTMs), and an attention mechanism to prioritize the most informative time steps. Empirical evaluations demonstrate clear superiority over existing approaches: the model reduces MSE by approximately 34% compared to a Bi-LSTM-only baseline (via ablation study) and by 26–45% across a range of state-of-the-art models (e.g., Transformer, GRU, ARIMA + XGBoost), as shown in Table II. It also achieves higher R^2 values (up to 15% relative improvement vs. weaker baselines), confirming more reliable and consistent forecasts. This framework effectively addresses the challenges of dynamic, bursty workloads in modern cloud environments, offering a practical advancement for adaptive resource management.

REFERENCES

[1] R. Atat, L. Liu, J. Wu, G. Li, C. Ye, and Y. Yang, "Big Data Meet Cyber-Physical Systems: A Panoramic Survey," *IEEE Access*, vol. 6, pp. 73603–73636, 2018, https://doi.org/10.1109/ACCESS.2018.2878681.

[2] A. Perçuku, D. Minkovska, and N. Hinov, "Enhancing Electricity Load Forecasting with Machine Learning and Deep Learning," *Technologies*, vol. 13, no. 2, Feb. 2025, Art. no. 59, https://doi.org/10.3390/technologies13020059.

[3] D. A. Shafiq, N. Z. Jhanjhi, and A. Abdullah, "Load balancing techniques in cloud computing environment: A review," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 7, pp. 3910–3933, July 2022, https://doi.org/10.1016/j.jksuci.2021.02.007.

[4] P. Vyas, K. M. Ragothaman, A. Chauhan, and B. Rimal, "Data augmentation and generative machine learning on the cloud platform," *International Journal of Information Technology*, vol. 16, no. 8, pp. 4833–4843, Dec. 2024, https://doi.org/10.1007/s41870-024-02104-5.

[5] Y. Himeur, A. N. Sayed, A. Alsalemi, F. Bensaali, and A. Amira, "Edge AI for Internet of Energy: Challenges and perspectives," *Internet of Things*, vol. 25, Apr. 2024, Art. no. 101035, https://doi.org/10.1016/j.iot.2023.101035.

[6] P. W. Tien, S. Wei, J. Darkwa, C. Wood, and J. K. Calautit, "Machine Learning and Deep Learning Methods for Enhancing Building Energy Efficiency and Indoor Environmental Quality – A Review," *Energy and AI*, vol. 10, Nov. 2022, Art. no. 100198, https://doi.org/10.1016/j.egyai.2022.100198.

[7] S. Verma and A. Bala, "ETSA-LP: Ensemble Time-Series Approach for Load Prediction in Cloud," *Computing and Informatics*, vol. 43, no. 1, pp. 64–93, 2024, https://doi.org/10.31577/cai_2024_1_64.

[8] H. Zhang, J. Li, and H. Yang, "Cloud computing load prediction method based on CNN-BiLSTM model under low-carbon background," *Scientific Reports*, vol. 14, no. 1, Aug. 2024, Art. no. 18004, https://doi.org/10.1038/s41598-024-68339-1.

[9] J. Wang, J. Pan, F. Esposito, P. Calyam, Z. Yang, and P. Mohapatra, "Edge Cloud Offloading Algorithms: Issues, Methods, and Perspectives," *ACM Computing Surveys*, vol. 52, no. 1, pp. 1–23, Jan. 2020, https://doi.org/10.1145/3284387.

[10] M. I. Khaleel, "A dynamic weight–assignment load balancing approach for workflow scheduling in edge-cloud computing using ameliorated moth flame and rock hyrax optimization algorithms," *Future Generation Computer Systems*, vol. 155, pp. 465–485, June 2024, https://doi.org/10.1016/j.future.2024.02.025.

[11] N. Hogade and S. Pasricha, "A Survey on Machine Learning for Geo-Distributed Cloud Data Center Management," *IEEE Transactions on Sustainable Computing*, vol. 8, no. 1, pp. 15–31, Jan. 2023, https://doi.org/10.1109/TSUSC.2022.3208781.

[12] S. Verma and A. Bala, "Efficient Auto-scaling for Host Load Prediction through VM migration in Cloud," *Concurrency and Computation: Practice and Experience*, vol. 36, no. 4, Feb. 2024, Art. no. e7925, https://doi.org/10.1002/cpe.7925.

[13] S. Simaiya *et al.*, "A hybrid cloud load balancing and host utilization prediction method using deep learning and optimization techniques," *Scientific Reports*, vol. 14, no. 1, Jan. 2024, Art. no. 1337, https://doi.org/10.1038/s41598-024-51466-0.

[14] M. Valizadeh and S. J. Wolff, "Convolutional Neural Network applications in additive manufacturing: A review," *Advances in Industrial and Manufacturing Engineering*, vol. 4, May 2022, Art. no. 100072, https://doi.org/10.1016/j.aime.2022.100072.

[15] A. Goel, A. K. Goel, and A. Kumar, "The role of artificial neural network and machine learning in utilizing spatial information," *Spatial Information Research*, vol. 31, no. 3, pp. 275–285, June 2023, https://doi.org/10.1007/s41324-022-00494-x.

[16] S. Zairi and M. Freihat, "Electric Load Forecasting using Machine Learning for Peak Demand Management in Smart Grids," *Engineering, Technology & Applied Science Research*, vol. 15, no. 3, pp. 23335–23346, June 2025, https://doi.org/10.48084/etasr.10687.

[17] H. Toumi, Z. Brahmi, and M. M. Gammoudi, "RTSLPS: Real time server load prediction system for the ever-changing cloud computing environment," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 2, pp. 342–353, Feb. 2022, https://doi.org/10.1016/j.jksuci.2019.12.004.

[18] "google/cluster-data." Google, [Online]. Available: https://github.com/google/cluster-data.