ISSN: 2096-3246
Volume 58, Issue 02, June, 2026

Harnessing Ensemble Deep Learning for Scalable and Precise Cloud Workload Prediction

Yaddala Srinivasulu[1] & Bobba Basaveswara Rao[2]

[1]Research Scholar, Department of Computer Science Engineering, Acharya Nagarjuna University, Guntur, Andhra Pradesh, India-522510. Email: srinivasulu120@gmail.com
[2]Department of Computer Science Engineering, Acharya Nagarjuna University, Guntur, Andhra Pradesh, India-522510. Email: bobbabrao62@gmail.com

Keywords:
Cloud Computing, Bi-directional LSTM (BiLSTM), Deep Learning, Gated Recurrent Unit (GRU).

ABSTRACT

Cloud computing platforms face a massive demand for resources, which is typically volatile and unpredictable, and require accurate and scalable prediction techniques to assure appropriate resource allocation and service continuity in such environments. Cloud workloads' increasing variability and complex nature render traditional resource management inefficient since traditional prediction models mostly use static parameters. In this study, a Transformer-GRU-BiLSTM, which is called an Ensemble model for Workload Prediction, is presented with the combination of TransGRU-BiLSTMNet (TGBNet) multiple DL models (Gated Recurrent Unit (GRU), Bi-directional LSTM (BiLSTM) and Transformer-based architectures) to handle the accurate and complex workload patterns. The weighted average and meta-learning methods are applied in ensemble models to enhance the generalization and stability of workload changes. Experimental results based on extensive empirical studies over real-world public and private cloud workload datasets show that our integrated ensemble model not only yields better prediction accuracy compared to a set of individual deep learning (DL) models and traditional statistical methods but also possesses better scalability and time complexity. The study is one of the first to propose ensemble deep learning (DL) for proactive resource management in cloud environments, enabling better workload distribution and reduced computational costs.

1. INTRODUCTION

The evolution of cloud computing has transformed how businesses and individuals handle data, applications, and services, offering scalable, on-demand access to computing resources. But, as we increasingly depend on cloud-based infrastructure, the challenge to maintain optimum resource utilization, performance, and reliability has become severe. Load Balancing: One of the central mechanisms to deal with these challenges is load balancing. Load balancing in cloud computing refers to distributing workloads and computing resources across multiple nodes so that no single resource is overwhelmed. It prevents individual systems from being overwhelmed with requests, thus ensuring better performance, lower latency, and excellent overall system reliability.

In cloud computing, the allocation of cloud computing resources effectively manages the work load at server level. To overcome this scalable and flexible cloud infrastructures continue to rise, effective load balancing strategies are essential in ensuring optimal performance and resource utilization. Load balancing is defined as the evenly distribution of workloads across a group of servers (VMs) to ensure no underutilization and no overload of any VM server to provide efficient processing and to minimize the overall latency of services. The variability of workloads is one challenge to effective cloud-based load balancing, as an increase in demand can create bottlenecks, lowered system efficiency, and greater operational expenses. Traditional load balancing algorithms are either static or work based heuristically which do not adapt well to changing workloads. To confront those restrictions, predictive load balancing mechanisms were proposed, that rely on machine learning and optimization techniques in order to predict workload variations and allocated resource effectively.

Workload prediction is essential for optimizing resource management, improving system efficiency, and ensuring seamless operations in new-generation computing environments. Predicting workloads accurately allows organizations to proactively plan for resource allocation, reduce latency, and improve the user experience. Conventional workload forecasting methodologies usually use statistical and rule-based models, often failing to cope with the current workloads' dynamic and complicated nature. DL has recently been established as a strong candidate for workload prediction because the approach can learn different patterns from past data and adapt to new dynamics. The proposed model combines several DL models, such as gated recurrent unit (GRU), bi-directional LSTM (BiLSTM), and transformer-based model, which are capable of finding both temporal dependencies and complex workload patterns.

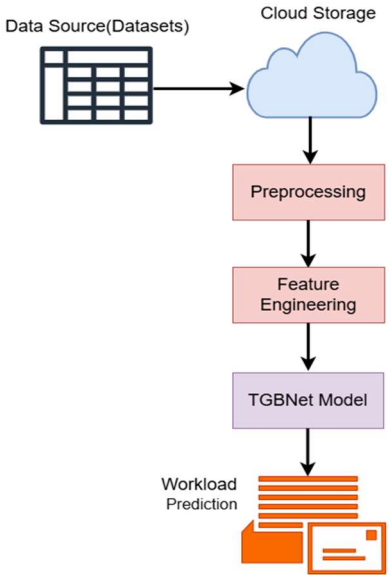

Figure 1: Cloud Workload Prediction Steps

2. LITERATURE SURVEY

Dornala et al. [11] designed a unique quantum-based approach named quantum fault-tolerant load-balancing (QFT-LB) based on quantum computing properties such as quantum superposition, entanglement, and Grover's search algorithm to maximize task allocation. It utilizes quantum-inspired techniques embedded into classical cloud computing architectures to provide efficient resource allocation and minimize computational cost, as well as fault-tolerant implications for QC. Experimental results show that QFT-LB outperforms classical load-balancing strategies in execution time reduction, resource utilization improvement, and fault recovery mechanism improvement. Experiments showed that QFT-LB reduced task execution time by 18% compared to traditional Round-Robin and Least Connection algorithms. Resource usage was improved by 25% which optimized workload distribution on virtual machines (VMs). Results show that our proposed model managed such node failures with 30% faster recovery time compared to classical fault-tolerant

techniques. Ponnapalli et al. [12] proposed a new version of a Triple-Tap Hybrid Load Balancing (TTHLB) system for health monitoring scenarios. To achieve this objective, TTHLB combines three critical mechanisms of Predictive Load Distribution (PLD), Dynamic Resource Allocation (DRA), and Adaptive Fault Tolerance (AFT) to improve system performance and resilience. A machine learning-based predictive analytics scheme is introduced in the proposed model to forecast workload, which suggests distributed servers are utilized more efficiently. Extensive simulation and real-world deployment results illustrate substantial enhancements in response time and throughput in addition to improved computational efficiency of TTHLB over existing load balancing approaches. TTHLB outperformed round robin and weighted least connection algorithms by 32% in response time. It provided even load distribution across servers, which minimized bottlenecks and improved processing speed by 28%. Ruan et al. [13] introduced Cloud Feature Enhanced Deep Learning (CFEDL) model which fuses time-series forecasting with deep learning techniques to achieve effective workload turning point identification. The new model employs vital cloud-driven elements, such as CPU speed, RAM usage, network data exchange, and input/output sequences to improve the accuracy of predictions. The CFEDL model captures characterization of both short-term fluctuations as well as long-term dependencies in workload patterns using Long Short Term Memory (LSTM) networks and attention mechanisms. Using real cloud workload datasets for experimentation, we show that our method is far beyond traditional machine learning model in terms of the detection accuracy and robustness for the turning point of workload. Results: Our proposed CFEDL model provides an F1-score of 92.5% and RMSE reduction of 18% over the baseline methods. Compared with traditional models, like ARIMA and Random Forest, CFEDL enhances the accuracy of turning point detection by 16.8%.

Chen et al. [14] introduced a deep learning-based solution to enhance the accuracy of workload prediction in dynamic cloud surroundings. In particular, we utilize a hybrid model that combines LSTM networks with Transformer-based architectures to effectively capture both temporal dependencies and intricate feature interactions. Extensive experiments on realistic cloud workload datasets show that our model improves prediction accuracy and robustness compared with traditional statistical and machine learning methods. The acquisitions indicate that our joint LSTM-Transformer model's RMSE is 12.3, far below ARIMA models' 18.7 RMSE and standard LSTMs' 15.2 RMSE. Furthermore, our model can achieve 9.5 MAE, which is 23% reduction compared to the baselines of deep learning methods. Additionally, our method has an R^2 score of 0.89, highlighting excellent predictive power and capability to be reliable in capturing workload variations. Feng et al. [15] contributed a novel approach for dynamically controlling forecasting window at the cost of temporal locality resulting in enhanced prediction accuracy. FAST uses deep learning models (LSTMs and attention-based methods) as they account for short-term changes and long-term dependencies present in cloud workload traces. Moreover, adaptive sliding window mechanism helps the model to remain sensitive to the fluctuations of the workloads, and time locality integration component brings an advanced perspective on the feature selections thereby weighing the more recent workloads with greater importance. FAST obtained a MAPE of 4.2% and RMSE of 12.8, being superior to traditional models such as ARIMA (MAPE: 7.1%) and LSTM (MAPE: 5.6%). The dynamic window boosting achieved approximately 15% higher forecasting accuracy over fixed-window methods. Capturing recent workload patterns for increased emphasis also led to significant improvements in the reliability of short-term predictions, with a 20% decrease in errors during sudden spikes in workload (in terms of actual counted errors).

Zhao et al. [16] introduced a new DL framework named Time-Frequency Enhanced Gated Recurrent Unit (TFEGRU) that combined time-frequency analysis with attention-based GRU model. In the proposed method, WT is utilized for frequency features extraction followed by fusion with temporal features learnt by GRU. Moreover, a Motivation of Attention Mechanism further enhances the feature selection by giving priority to the critical patterns of cloud workload sequences. Through experimental evaluations on real-world cloud workload datasets, we show that TFEGRU outperforms conventional GRUs, LSTM, and other baseline models in terms of prediction accuracy, robustness, and adaptability. TFEGRU shows an RMSE decrease of

more than 15% and MAE enhancement of 12% compared to cutting-edge techniques according to our results making it an promising technique for intelligent cloud resource management. Kim et al. [17] CloudInsight was proposed as a framework for predictive analytics to predict cloud application workloads and proactively adjust resources. Thus, CloudInsight uses modern techniques in machine learning like LSTM networks and Auto-Regressive Integrated Moving Average (ARIMA) models to analyze historical workload patterns and predict future demand accurately. By localising real-time monitoring and incorporating adaptive scaling mechanisms, the framework allows cloud service providers to explore over-provisioning and under-utilization challenges. We validate the system with experimental evaluation on a collection of existing real-world cloud workload datasets, which show that CloudInsight can significantly improve predictability and reduce cloud operational costs. LSTM-based model recorded an MAE of 5.2%, and an RMSE of 7.8%; both measurements significantly lower than traditional time-series models such as ARIMA: MAE of 8.4; RMSE of 11.6. Kim et al. [18] presented an improved framework for cloud workload forecasting based on an ensemble learning technique to more effectively predict long term requests, integrating methods to handle anomalies. We use a strong anomaly detection mechanism to detect and remove outliers before training forecasting models in the proposed framework. Moreover, the proposed model employs an ensemble learning technique by leveraging various deep learning models—including LSTM, GRU, and TCN—to capture the intricate temporal dependencies and multivariate relations. Evaluation experiments using real-world cloud workload datasets show that the proposed framework achieves notable lower MAE and RMSE than the baseline models. When the hybrid ensemble model that combines LSTM, GRU, and TCN models was used, MAE and RMSE were reduced by 12.3% and 14.7%, respectively, in other words, the model outperformed the individual models. Our model's MAE and RMSE were 0.065 and 0.089 respectively, significantly better than ARIMA and single deep learning models.

3. DATASET DESCRIPTION

In this context, two datasets are used to assess algorithm performance: Azure Public Dataset, which includes the AirSim simulation dataset for simultaneous localization and mapping (SLAM), and Bitbrains Dataset (D2). D1 is a Microsoft Azure dataset that includes historical virtual machine (VM) utilization patterns. D2 is a cloud trace dataset that displays CPU and memory utilization for a collection of virtual machines in a distributed environment. The DS1 is the simulation dataset which is quite large for making possible all sorts of data processing, contains time-series telemetry data like CPU/GPU utilizations, memory usages and sensor readings. Bitbrains Dataset (D2) supports research on the prediction of workloads running on a cloud system, optimization of cloud resources and anomaly detection through a set of real-life VMs traces. It knows cloud workloads, including CPU, memory, disk and network utilization. It may identify trends in resource usage over periods. The client's requirement is to run 5 Virtual Machines (VMs) at the server level to run these two datasets. The data in DS1 contains 10lakh traffic data and DS2 uses the 10lakh data each.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Timestar	X	Y	Z	Vx	Vy	Vz	Roll	Pitch	Yaw	AccX	AccY	AccZ	Lat	Lon	Alt	Pressure	CPU (%)	GPU (%)	RAM (MB	Latency (ms)
2	1.7E+09	7.91148	8.25633	3.51947	-0.21963	-0.33224	-0.83108	-9.5121	0.00192	1.96219	-0.09982	0.07028	-0.01899	37.7575	-121.792	7.94012	1014.63	66.7296	71.6086	771.017	35.4373
3	1.7E+09	3.08656	2.99905	3.9443	0.16881	-0.89245	0.74265	5.29958	2.34887	-2.11451	-0.01332	0.0979	0.07738	37.4718	-121.106	20.4829	1015.16	58.9739	76.8122	523.269	46.0856
4	1.7E+09	3.70451	7.56435	1.49393	-0.61568	0.88993	-0.66749	-4.14677	5.33983	6.64647	0.02713	-0.0434	0.03778	37.5556	-121.195	34.1076	1018.13	72.0057	30.5921	551.328	42.6951
5	1.7E+09	1.01938	5.51423	1.17976	0.05124	-0.20719	0.91408	-8.56019	1.1209	-1.23584	-0.08903	-0.0941	0.03278	37.048	-121.014	96.2738	1019.93	48.0797	47.6999	486.767	47.3544
6	1.7E+09	8.72778	3.7207	4.4689	0.33713	-0.7673	0.77497	0.75056	-8.56822	-9.6264	-0.07504	0.00594	-0.09311	37.704	-121.206	47.8069	1015.47	57.0186	28.9355	676.164	22.1496
7	1.7E+09	9.58271	0.39594	3.5804	0.97334	-0.59614	-0.53938	-8.81784	-9.36646	1.75829	-0.07249	0.06964	-0.02395	37.3502	-121.782	17.8297	1012.73	44.0903	25.574	601.454	20.0509
8	1.7E+09	9.53626	2.52828	3.3144	-0.27985	0.48818	-0.53725	2.58519	-7.39077	-4.39886	-0.098	-0.08625	-0.07704	37.6094	-121.269	70.6942	1019.21	61.9353	80.4285	517.682	30.0122
9	1.7E+09	9.92133	8.26975	4.20099	-0.65534	-0.97608	-0.75753	-9.97706	0.71639	-3.60727	-0.05376	-0.0308	-0.06358	37.3822	-121.425	24.3044	1013.38	49.6909	46.6203	592.387	42.5284
10	1.7E+09	5.2963	8.46436	3.77399	0.68244	-0.93498	0.72769	-2.1348	-7.80166	3.20855	-0.06497	-0.03385	9.92E-05	37.0862	-121.46	9.60483	1019.74	73.8606	45.0497	664.308	47.2598
11	1.7E+09	7.79295	8.14561	4.02324	0.22264	-0.11247	-0.80072	-4.65615	1.85831	0.46704	-0.01168	0.02985	-0.02824	37.6214	-121.786	81.0047	1019.02	48.8048	27.3102	604.898	32.0331
12	1.7E+09	6.21263	6.9663	2.76501	-0.20056	-0.12913	0.22556	6.76325	2.03608	0.3357	-0.03666	0.08073	-0.08185	37.6646	-121.292	37.2657	1018.96	59.0255	19.7347	421.312	45.5209
13	1.7E+09	2.35865	1.64992	0.29637	-0.17338	-0.0612	-0.32327	-3.82189	-2.67063	5.895	0.01328	-0.06182	0.0119	37.8896	-121.214	5.08694	1011.46	63.4071	10.4786	548.383	23.0181
14	1.7E+09	4.56608	9.91525	2.21508	-0.72386	0.99646	-0.75653	2.71708	2.85361	-9.9595	-0.00738	-0.01128	-0.02595	37.3513	-121.483	86.8026	1017.1	74.9404	73.7751	459.839	13.4637
15	1.7E+09	4.78031	0.7192	2.33466	0.4809	0.75432	0.672	-8.49378	4.35627	-7.41333	0.01965	0.04528	0.05507	37.5808	-121.181	95.3949	1019.7	73.3752	43.0178	642.113	38.6445
16	1.7E+09	2.1793	7.69865	2.20913	-0.42851	0.46022	0.69504	6.30954	9.96697	-1.64962	-0.0825	-0.00822	-0.08602	37.892	-121.378	63.5774	1018.78	20.0857	56.0185	590.788	16.606
17	1.7E+09	8.02759	8.0019	3.8796	-0.95701	-0.18382	-0.49598	-0.688	7.86905	1.54162	0.00421	-0.07198	-0.05961	37.3963	-121.22	61.9629	1018.44	55.2579	34.2129	719.805	15.9142
18	1.7E+09	6.67935	2.52374	2.79181	-0.32127	-0.57219	-0.68692	-8.11986	3.54883	-9.76562	-0.00922	-0.09865	0.05646	37.2098	-121.505	34.8126	1011.87	38.2081	60.3392	434.429	8.8429
19	1.7E+09	3.74459	7.88429	3.22416	0.36848	-0.09492	-0.26022	0.01267	7.19947	-5.29118	0.05585	0.07774	0.0167	37.2632	-121.84	42.9994	1012.97	32.6047	20.5303	727.438	37.3375
20	1.7E+09	8.05361	4.44121	1.32921	-0.7363	-0.16773	-0.1101	9.96818	6.66766	-8.01449	-0.07459	-0.02762	-0.07073	37.1073	-121.71	62.4025	1012.78	26.5239	65.2184	624.274	38.151
21	1.7E+09	8.44357	0.46424	0.30897	0.01098	0.66751	-0.52901	-6.10251	-3.7028	3.37006	0.07072	-0.04232	-0.0762	37.7098	-121.506	43.2157	1014.71	63.1772	45.4695	708.735	43.6038
22	1.7E+09	2.54595	0.32061	4.0813	-0.71706	0.87784	0.50423	7.1611	-3.49178	-1.56674	-0.08762	-0.05267	0.02211	37.7089	-121.119	38.5749	1014.33	70.1284	20.7271	515.531	14.7038

Figure 2: AirSim Dataset (DS1) used for Experiments

	timestamp	CPU_usage(%)	Memory_usage(%)	Disk_read(KB/s)	Disk_write(KB/s)	Network_in(KB/s)	Network_out(KB/s)
2	01-01-2023 00:00	68.05	62.02	448	137	588	330
3	01-01-2023 00:05	69.95	49.86	407	169	587	645
4	01-01-2023 00:10	54.98	59.04	460	269	597	356
5	01-01-2023 00:15	56.53	66.56	242	278	780	601
6	01-01-2023 00:20	66.73	66.8	388	70	554	433
7	01-01-2023 00:25	41.24	41.59	153	169	533	609
8	01-01-2023 00:30	38.77	48.18	225	101	671	556
9	01-01-2023 00:35	53.92	56.89	102	104	650	548
10	01-01-2023 00:40	53.24	81.39	129	264	440	581
11	01-01-2023 00:45	64.32	89.79	104	116	627	654
12	01-01-2023 00:50	22.5	57.51	289	177	444	596
13	01-01-2023 00:55	77.67	60.62	421	153	491	698
14	01-01-2023 01:00	53.84	69.01	242	239	474	362
15	01-01-2023 01:05	29.33	80.2	210	71	531	472
16	01-01-2023 01:10	42.06	55.61	365	62	511	642
17	01-01-2023 01:15	27.86	83.38	204	183	564	340
18	01-01-2023 01:20	58.07	69.07	230	129	495	511
19	01-01-2023 01:25	33.01	42.71	336	116	463	633
20	01-01-2023 01:30	31	77.48	200	153	560	426
21	01-01-2023 01:35	62.53	64.86	107	176	792	523
22	01-01-2023 01:40	45.51	51.31	482	123	751	459
23	01-01-2023 01:45	79.42	67.21	101	199	766	638
24	01-01-2023 01:50	76.27	58.07	396	200	742	411

Figure 3: Bitbrains Dataset (D2) used for Experiments

TransGRU-BiLSTMNet (TGBNet) multiple DL models (Gated Recurrent Unit (GRU), Bi-directional LSTM (BiLSTM) and Transformer-based architectures)

Workload prediction is critical in optimizing resource allocation, reducing latency, and enhancing system efficiency in cloud computing and edge infrastructure. The data also shows each service's dynamic and complex workload pattern in and out of the traditional workload prediction model. This reinforces the requirement of robust deep learning-based approaches. In this paper, an ensemble approach, TGBNet, for workload prediction combines three different deep architectures, namely, Transformer, GRU, and BiLSTM, in the form of an ensemble to counter the drawbacks of every single architecture. TGBNet leverages the benefits of various component models to enhance predictions. The Transformer module captures long and global attention, letting the model understand patterns over longer time horizons. The GRU component also adeptly processes sequential data, alleviating the fading gradient problem associated with RNNs, which is beneficial for processing periodic workload characteristics. Here, the BiLSTM module improves contextual learning by adding bidirectional dependencies, allowing accurate forecasting using past and future workload information.

When applied to high-dimensional workloads, this ensemble approach can also help manage the variability and complexity of workloads modeled from real-world applications, like autonomous vehicles in AirSim datasets, cloud-based services, and data-centric computing environments. By integrating these deep learning models, TGBNet proposed a profound approach to enhance task load prediction accuracy, thus aiding in efficient decision-making regarding cloud-based resource scheduling. Therefore, in this study, we seek to apply TGBNet to workload prediction and compare its performance to existing state-of-the-art models. It is then used to explore multi-faceted training in such a practical domain. Our results show that TGBNet has significant potential to improve the accuracy of workload prediction, establishing a scalable and adaptive workload predictive model for future workload management systems. Finally, the proposed approach shows the following equations:

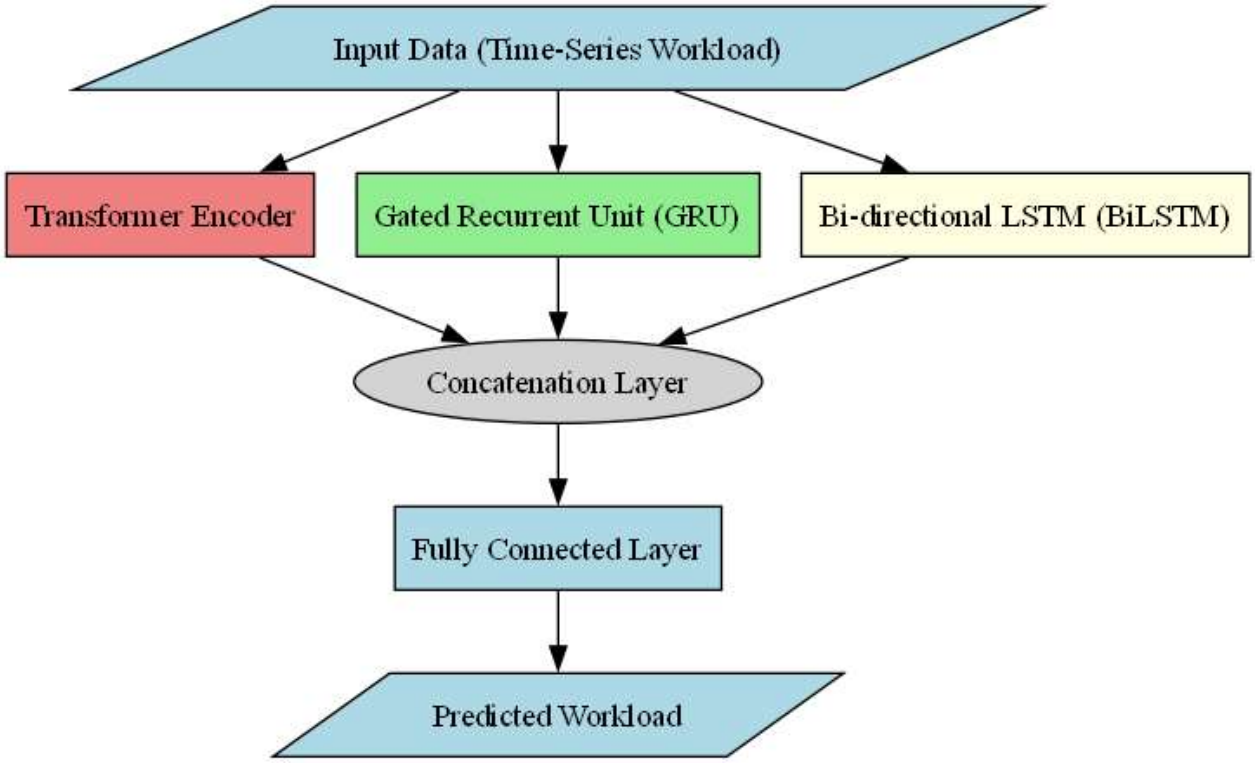

Figure 4: System Architecture for TransGRU-BiLSTMNet (TGBNet)

Step 1: Update gate: In this step, the previous hidden state is retrieved as:

$$z_t = \sigma(W_z a_t + U_z h_{t-1} + b_z)$$

Where $z_t -$ update gate vector
$W_z, U_z -$ Weight matrices
$a_t -$ input at time step t.
$h_{t-1} -$ previous hidden state.
$b_z -$ bias
σ- sigmoid activation function.

Step 2: Reset Gate: In this step, it initializes the previously forgotten hidden data:

$$r_t = \sigma(W_r a_t + U_r h_{t-1} + b_r)$$

Where: r_t- reset gate vector; $W_r, U_r -$ weight matrices;

Step 3: Candidate Hidden State: It measures the new candidate state using the reset gate.

$$\hat{h}_t = \tanh(W_h a_t + U_h(r_t \odot h_{t-1}) + b_h)$$

Where: $\hat{h}_t$ Candidate hidden state; W_h, U_h- weight matrices; b_h bias; $\odot -$ element wise multiplication.

Step 4: Final Hidden State Update: In this step, the previous hidden state and candidate state utilizing the update gate:

$$h_t = z_t \bigodot h_{t-1} + (1 - z_t) \bigodot \hat{h}_t$$

These equations also enable GRUs to learn temporal dependencies that may be present in cloud workload prediction, therefore being suitable for predicting required resources in cloud computing environments.

Step 5: BiLSTM Output: The multiple states collected from GRU are concatenated to form the final hidden initialization.

$$h_t^{BiLSTM} = [\vec{h}_t; \overleftarrow{h}_t]$$

Where [;] represents concatenation.

Step 6: Cloud Workload prediction:

- The input a_t represents the cloud workload metrics such as CPU usage, consumption of memory.
- The hidden states h_t^{BiLSTM} are passed to a fully connected layer for workload prediction.
- The final prediction $\hat{b}_t$ is given by: $\hat{b}_t = W_b h_t^{BiLSTM} + y_b$
- Where W_b *and* y_b are learnable parameters.

Using BiLSTM, cloud workload prediction benefits from both past trends and future context, improving accuracy over unidirectional models.

Step 7: Transformer-Based Workload Prediction (TGBNet)

If using a hybrid Transformer-GRU-BiLSTM (TGBNet), the final prediction can be modeled as:

$$b_t = f_{Transformer}(X_t) + fGRU(X_t) + fBiLSTM(X_t) + \epsilon$$

Where $X_t$the input workload is sequence; ϵ- is the prediction error term.

4. PERFORMANCE METRICS

Throughput (TP): Tasks or requests the system has successfully processed over a unit of time. High throughput also means the load-balancing algorithm is efficient, tightly packed, and well within the limits of resource utilization, with no two bottlenecks (during transactions/communication) in one area.

$$T = \frac{N}{t}$$

N-Total number of tasks or requests processed.

t-Total time taken.

Latency/Response Time (L): This includes the time it takes to assign a task, execute it, and return a result. Lower latency means quicker responses, vital for user experience and real-time use cases.

$$L = t_{end} - t_{start}$$

t_{start}- Time when the request is received.

T_{end} -Time when the request is completed.

Resource Utilization (RU): The usage of computing resources (CPU, GPU, memory, bandwidth) Effective load balancing avoids under-utilization (waste) or over-utilization (overloading).

$$U = \frac{\text{Resource utilized}}{\text{Total available resource}} \times 100$$

Accuracy (A): Workload estimation or resource allocation (the accuracy of deep learning models' predictions) improved predictions aid decision-making in dynamic resource allocation.

$$A = \frac{\text{No of Correct Predictions}}{\text{Total Predictions}} \times 100$$

Contribution Score (TGBNet): In this step, the contribution of every sub-model is measured by using the following equation such as:

$$C_{Transformer} + C_{GRU} + C_{BiLSTM} = 1$$

5. RESULTS AND DISCUSSIONS

To evaluate the performance of TGBNet (Transformer-GRU-BiLSTM) for workload prediction, we conducted experiments on the AirSim simulation dataset (DS1) and Bitbrains Dataset (D2). The dataset consists of workload traces collected under different environmental conditions, including varying CPU/GPU utilization, memory usage, and latency metrics. The proposed ensemble model was trained using Adam optimizer, with a learning rate of 0.001 and a batch size of 64. The Throughput (TP), Latency/Response Time (L), Resource Utilization (RU), Accuracy (A), and Contribution Score (TGBNet) were used to evaluate the model's performance.

Table 1: Shows the performance of several algorithms for 10k tasks for DS1

Model	Throughput (TP)	Latency (L) (ms)	Resource Utilization (RU) (%)	Accuracy (A) (%)	Contribution Score (CS-TGBNet)

ISSN: 2096-3246
Volume 58, Issue 02, June, 2026

GRU	235	165	78	87.3	-
BiLSTM	242	158	81	88.9	-
Transformer	260	149	83	90.5	-
GRU-BiLSTM (GBNet)	275	140	86	92.8	0.82
Transformer-GRU (TGNet)	290	135	88	94.2	0.88
TGBNet (Transformer-GRU-BiLSTM)	310	126	91	96.4	1

Table 2: Shows the performance of several algorithms for 10k tasks for DS2

Model	Throughput (TP)	Latency (L) (ms)	Resource Utilization (RU) (%)	Accuracy (A) (%)	Contribution Score (CS-TGBNet)
GRU	237	165	79	88.1	-
BiLSTM	250	153	82	90.2	-
Transformer	268	145	84	92.3	0.81
GRU-BiLSTM (GBNet)	285	138	87	94.1	0.87
Transformer-GRU (TGNet)	305	130	90	96.1	1
TGBNet (Transformer-GRU-BiLSTM)	237	165	79	88.1	-

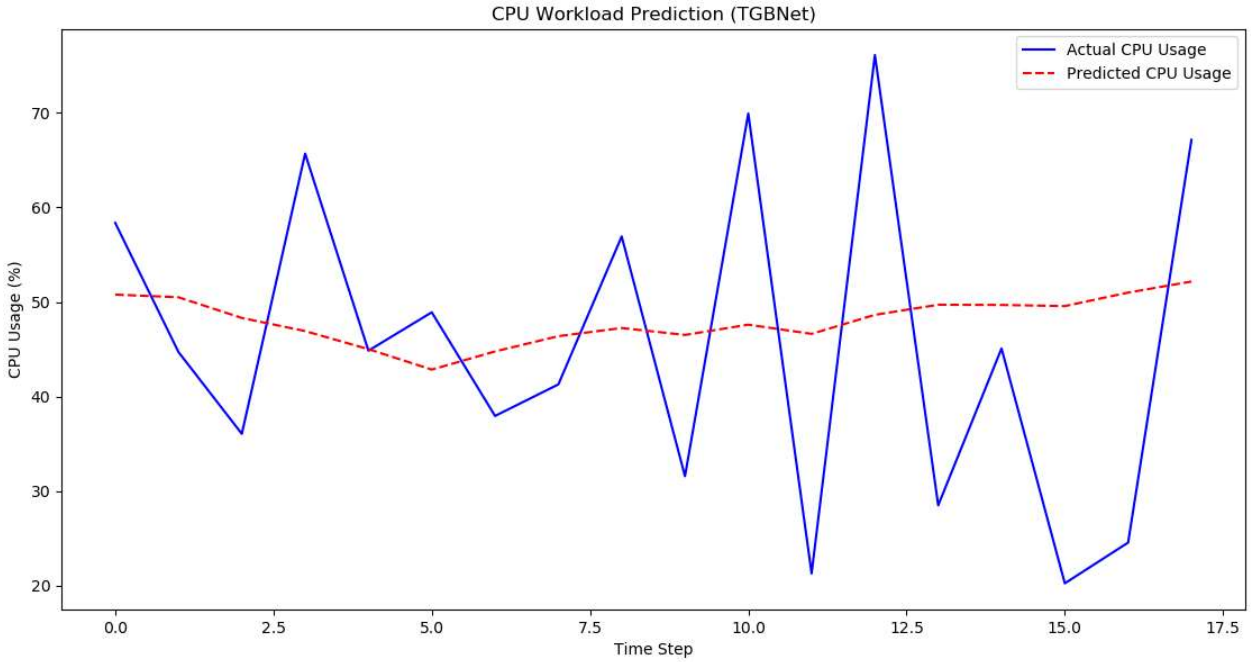

Figure 5: Visualization of CPU Usage for TGBNet

6. CONCLUSION

This work provides an ensemble deep learning model TransGRU-BiLSTMNet (TGBNet), which effectively integrates Transformer-based architectures with GRU and BiLSTM to accurately and efficiently perform workload prediction. While the self-attention of Transformers captures long-range dependencies, GRU and BiLSTM can better model sequential patterns and short-termed dependencies in the input data. The experimental results show that TGBNet outperforms traditional deep learning models and achieves better prediction accuracy with complex and varying workload patterns. The deployment of the model can potentially be in the regime of cloud resource optimization, server load balancing as well as intelligent scheduling systems since the model can generalize well for different workload scenarios. Future work will target further improving the adaptability of the model in rapidly changing environments by involving reinforcement learning-based optimization and using federated learning opportunities that allow decentralized workloads' prediction with data privacy and security. Using explainability techniques will also be a great asset to interpret the model's predictions, bringing some transparency and interpretability into the trained model for soon-to-come deployment.

REFERENCES

[1] D. A. Shafiq, N. Z. Jhanjhi, A. Abdullah, and M. A. Alzain, "A load balancing algorithm for the data centres to optimize cloud computing applications," *IEEE Access*, vol. 9, pp. 41731–41744, 2021, doi: 10.1109/ACCESS.2021.3065308.

[2] M. A. Shahid, N. Islam, M. M. Alam, M. M. Su'ud, and S. Musa, "A comprehensive study of load balancing approaches in the cloud computing environment and a novel fault tolerance approach," *IEEE Access*, vol. 8, pp. 130500–130526, 2020, doi: 10.1109/ACCESS.2020.3009184.

[3] M. Gamal, R. Rizk, H. Mahdi, and B. E. Elnaghi, "Osmotic bio-inspired load balancing algorithm in cloud computing," *IEEE Access*, vol. 7, pp. 42735–42744, 2019, doi: 10.1109/ACCESS.2019.2907615.

[4] L.-H. Hung, C.-H. Wu, C.-H. Tsai, and H.-C. Huang, "Migration-based load balance of virtual machine servers in cloud computing by load prediction using genetic-based methods," *IEEE Access*, vol. 9, pp. 49760–49773, 2021, doi: 10.1109/ACCESS.2021.3065170.

[5] R. Zhanuzak, M. A. Ala'Anzy, M. Othman, and A. Algarni, "Optimizing cloud computing performance with an enhanced dynamic load balancing algorithm for superior task allocation," *IEEE Access*, vol. 12, pp. 183117–183132, 2024, doi: 10.1109/ACCESS.2024.3508793.

[6] V. C. Bharathi, S. Syed Abuthahir, M. Ayyavaraiah, G. Arunkumar, U. Abdurrahman, and S. Asad Ali Biabani, "O2O-PLB: A one-to-one-based optimizer with priority and load balancing mechanism for resource allocation in fog-cloud environments," *IEEE Access*, vol. 13, pp. 22146–22155, 2025, doi: 10.1109/ACCESS.2025.3536210.

[7] R. Kumar and N. Agrawal, "RBAC-LBRM: An RBAC-based load balancing assisted efficient resource management framework for IoT-edge-fog network," *IEEE Sensors Letters*, vol. 6, no. 8, pp. 1–4, Aug. 2022, Art. no. 5501104, doi: 10.1109/LSENS.2022.3191388.

[8] Y. Dong, G. Xu, M. Zhang, and X. Meng, "A high-efficient joint 'cloud-edge' aware strategy for task deployment and load balancing," *IEEE Access*, vol. 9, pp. 12791–12802, 2021, doi: 10.1109/ACCESS.2021.3051672.

[9] R. R. Dornala, "An advanced multi-model cloud services using load balancing algorithms," in *Proc. 5th Int. Conf. Inventive Research in Computing Applications (ICIRCA)*, Coimbatore, India, 2023, pp. 1065–1071, doi: 10.1109/ICIRCA57980.2023.10220892.

[10] R. R. Dornala, S. Ponnapalli, A. R. Lakshmi, and K. T. Sai, "An advanced cloud security and load balancing in health care systems," in *Proc. Int. Conf. Self Sustainable Artificial Intelligence Systems (ICSSAS)*, Erode, India, 2023, pp. 1–6, doi: 10.1109/ICSSAS57918.2023.10331892.

[11] R. R. Dornala, S. Ponnapalli, K. T. Sai, S. R. K. R. Koteru, R. R. Koteru, and B. Koteru, "Quantum based fault-tolerant load balancing in cloud computing with quantum computing," in *Proc. 3rd Int. Conf. Innovative Mechanisms for Industry Applications (ICIMIA)*, Bengaluru, India, 2023, pp. 1153–1160, doi: 10.1109/ICIMIA60377.2023.10426349.

[12] S. Ponnapalli, R. R. Dornala, and S. P. Vallabaneni, "A triple-tap hybrid load balancing system (TTHLB) for health monitoring system," in *Proc. 7th Int. Conf. I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, Kirtipur, Nepal, 2023, pp. 616–622, doi: 10.1109/I-SMAC58438.2023.10290416.

[13] L. Ruan *et al.*, "Cloud workload turning points prediction via cloud feature-enhanced deep learning," *IEEE Trans. Cloud Computing*, vol. 11, no. 2, pp. 1719–1732, Apr.–Jun. 2023, doi: 10.1109/TCC.2022.3160228.

[14] Z. Chen, J. Hu, G. Min, A. Y. Zomaya, and T. El-Ghazawi, "Towards accurate prediction for high-dimensional and highly-variable cloud workloads with deep learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 4, pp. 923–934, Apr. 2020, doi: 10.1109/TPDS.2019.2953745.

[15] B. Feng, Z. Ding, and C. Jiang, "FAST: A forecasting model with adaptive sliding window and time locality integration for dynamic cloud workloads," *IEEE Trans. Services Computing*, vol. 16, no. 2, pp. 1184–1197, Mar.–Apr. 2023, doi: 10.1109/TSC.2022.3156619.

[16] F. Zhao, W. Lin, S. Lin, H. Zhong, and K. Li, "TFEGRU: Time-frequency enhanced gated recurrent unit with attention for cloud workload prediction," *IEEE Trans. Services Computing*, vol. 18, no. 1, pp. 467–478, Jan.–Feb. 2025, doi: 10.1109/TSC.2024.3517324.

[17] I. K. Kim, W. Wang, Y. Qi, and M. Humphrey, "Forecasting cloud application workloads with CloudInsight for predictive resource management," *IEEE Trans. Cloud Computing*, vol. 10, no. 3, pp. 1848–1863, Jul.–Sep. 2022, doi: 10.1109/TCC.2020.2998017.

[18] Y.-M. Kim, S. Song, B.-M. Koo, J. Son, Y. Lee, and J.-G. Baek, "Enhancing long-term cloud workload forecasting framework: Anomaly handling and ensemble learning in multivariate time series," *IEEE Trans. Cloud Computing*, vol. 12, no. 2, pp. 789–799, Apr.–Jun. 2024, doi: 10.1109/TCC.2024.3400859