

HybridTimeNet: A Novel Approach to Proactive Load Forecasting in a Cloud Environment

Siddarth D Pai[†], Pranav P Vatsa[†], Nishtha Panchratna^{*†}, Saksham Alok[†], Dr. Jeny Jijo^{*}

^{*}Department of CSE, PES University, Bangalore, India

[†]Equal Contributions

siddarthdpai@gmail.com, pranav.ppv23@gmail.com, nishthap981@gmail.com, sakshamalok13@gmail.com, jenyjijo@pes.edu

Abstract—Load forecasting is the prediction of trends based on historical data. For cloud providers, these forecasts are essential for both long-term planning and immediate operational adjustments, allowing them to anticipate the resources required. The primary focus of this research is to develop an ensemble time series model specifically designed to improve the forecast accuracy in a cloud environment and ultimately improve the efficiency of the resources allocated. By combining multiple models that are commonly used for time series prediction, the strengths of each model are leveraged to capture the demand patterns as accurately as possible. HybridTimeNet is an amalgamation of Neural Prophet, LSTM and XGBoost. The predictions of the first layer are fed into a Multilayer Perceptron which gives the final prediction. To enhance resource allocation efficiency, the M/M/c queueing model was integrated, which accounts for the optimal number of servers required to handle variable loads for a given time period. HybridTimeNet is evaluated within the cloud domain using key metrics such as Service Level Agreement violations, number of unserved requests and wait time of each request as well as linear and quadratic error metrics namely mean squared error and root mean squared error. The outcome of this approach is to establish a scalable and reliable infrastructure model that balances demand fluctuations along with optimised resource utilisation.

Index Terms—Cloud Computing, Load Forecasting, Predictive Model, Time-Series Analysis, NeuralProphet-LSTM-XGBoost model

I. INTRODUCTION

Load forecasting is essential in cloud computing environments, as it directly has an impact on the resource allocation of the server center. Cloud infrastructures host a diverse array of applications and services, and each of them is unique and often has highly variable workloads, making accurate forecasting of resource demand crucial to ensure reliable workflow. As organisations increasingly rely on these systems for mission-critical operations, maintaining Service Level Agreements (SLAs) and delivering high Quality of Service (QoS) are necessary[1]. Commercially, a reactive scheduling method is used which allocates resources based on observing demand spikes and auto-scaling that can lead to significant challenges such as delays, high energy costs as well as unserved requests. This approach often results in delays in resource allocation causing underperformance during peak usage and risks SLA breaches. For example, if a sudden increase in traffic occurs and resources are not provisioned for use, users may experience slow response times or service outages. There is a need

for proactive forecasting strategies that allow the anticipation of workload fluctuations[2].

In this research, these challenges are addressed by proposing HybridTimeNet, an ensemble model of NeuralProphet, Long Short-Term Memory (LSTM) networks, and XGBoost for load forecasting. By integrating these models, we aim to create a comprehensive solution that can effectively capture both linear and non-linear patterns, seasonal variations, and long-term dependencies in cloud workloads. In HybridTimeNet, the output from each model is combined using a stacking mechanism, where the final prediction is derived based on the contributions of each model. To calculate the required virtual machines for a particular hour from the forecasted values the M/M/c queueing model is employed. The M/M/c model is particularly relevant for cloud environments as it accommodates varying numbers of servers needed to handle fluctuating load levels effectively. This model considers the arrival rates of incoming requests and the service rates of servers, facilitating an efficient distribution of resources. Coupled with scheduling algorithms this approach enhances workload management. By combining these methodologies, this proactive approach allows cloud providers to dynamically scale resources, ensuring infrastructure can meet evolving demand without incurring unnecessary costs or wasting energy, as well as mitigate SLA violations, optimize resource allocation, and maintain a higher level of QoS.

II. FORECASTING IN CLOUD

Forecasting, in the context of cloud computing, involves predicting future workloads or demand based on historical data over a period of time. The trends are monitored using statistical inference and machine learning techniques. By forecasting future resource usage, cloud providers can proactively allocate resources, and in turn, reduce the likelihood of performance degradation, SLA violations, or even system outages. Accurate load forecasting enables cloud providers to maintain optimal QoS by ensuring that resources are dynamically adjusted to meet real-time demands while minimizing waste and over-provisioning [3].

Traditional load management techniques, which rely on static or threshold-based rules, often fail to deal with the highly dynamic nature of cloud environments. Predictive models, which leverage machine learning and statistical approaches, provide more sophisticated methods for capturing and an-

anticipating fluctuations in workload, leading to more efficient resource allocation.

III. RELATED WORK

This section presents some of the current research in the domain of reactive and proactive scaling as well as the predictive models used for load forecasting in a cloud environment.

Shahid et. al compared reactive and proactive approaches particularly in the area of fault tolerance. A contrast was made between proactive methods that aim to address the problem before it manifests and are better suited for real-time applications, and reactive methods which focus on repetitive tasks making them more appropriate for long-term tasks, but potentially have a slower response [4]. Lorido-Botran et al. provided a simpler meaning to the scaling systems where a scale-up or a scale-down is defined to be a reactive technique when it does so in reaction to a change in the system. Proactive methods are implemented when there is an attempt to anticipate the future workload change and perform the vital scaling actions before the change occurs [5].

Siarni-Namini et.al. [6] compared LSTM with the traditional ARIMA time series forecasting model on an economical and financial time series dataset. The results of the paper concluded that LSTM was able to outperform conventional forecasting models like ARIMA in terms of predicting future data based on previous data. Vinay Kumar et.al [7] used the LSTM predictive model to forecast future COVID-19 cases as well as the ending date of the transmission in Canada. The paper stated that the LSTM networks was able to adapt to non-linearities of a dataset and also tackle the vanishing gradient problem by memorizing network patterns for long durations. A major advantage of the LSTM predictive model is its ability to selectively learn and remember or forget the required historical data. S K Lakshminarayanan et al. [8] observed this to be an advantage of LSTM over SVM in their work which compared four variations of both models over a stock price data set.

NeuralProphet is an advanced forecasting tool built on Facebook's Prophet framework, enhanced with neural network capabilities to achieve greater accuracy and flexibility in time series forecasting. It is specifically designed to handle complex data patterns and seasonality, making it suitable for a variety of applications, including energy consumption forecasting, stock price predictions, and demand forecasting [9]. A key strength of NeuralProphet lies in its ability to integrate traditional time series components, such as trend and seasonality, with sophisticated neural network structures, resulting in improved predictive performance [10]. However, NeuralProphet does have some limitations, such as high computational requirements and a risk of overfitting, especially when working with limited or noisy datasets. Despite these challenges, numerous studies have validated its effectiveness in capturing intricate temporal relationships and delivering accurate forecasts, highlighting its value in the field of time series forecasting [11].

XGBoost has demonstrated noteworthy improvements in predicting time series data across different application areas. Chen and Guestrin [12] first proposed the XGBoost, a scalable

machine-learning system making use of tree boosting, which later gained popularity in the forecasting of workloads in the cloud computing environment, as it could effectively accommodate nonlinear interactions among features. Xia et al. [13] emphasised XGBoost's architecture, a new algorithm for the learning of tree structures that implemented parallel and distributed computations to enhance learning and exploration of models, which made it appropriate for our scenario of predicting cloud workloads. In a study targeting only cloud environments, Nawrocki and Smendowski [14] reported that XGBoost models have improved the quality of forecasting 16% better than the statistical methods, and it was found to be better in forecasting sudden spikes in resource utilisation, which is vital to our research on forecasting the cloud workloads. Cai et al. [15] demonstrated that XGBoost is indeed effective by contrasting it with Random Forest (RF) algorithms, noting that it is more effective in the short-term devising load forecasting scenarios. Supporting our choice further Aguilar Madrid and Antonio [16] showed that the use of the XGBoost algorithm has outperformed several other algorithms of machine learning including the artificial neural networks and further provided useful insights about feature importance. These findings played an important role in our choice of including XGBoost in HybridTimeNet, especially for its competence in non-linear patterns and sudden variations observed in cloud workload characteristics.

Chen Li et al. [17] presented a combined model consisting of LSTM and XGBoost for power load forecasting. The paper constructed the combined model architecture using an error reciprocal method and by giving a larger weightage to the model with a smaller error. Duan et.al proposed a combination model which integrated LSTM-RF-XGBoost. The three individual algorithms were used as the base models in the first layer, with the multiple linear regression (MLR) algorithm serving as the meta-model in the second layer, merging the demand prediction model based on the hybrid model into the overall demand prediction model[18].

IV. TIME SERIES FORECASTING USING MACHINE LEARNING TECHNIQUES

A. Time Series Analysis in Cloud Computing Environments

Time Series data represents observational data recorded at specific time intervals. It has a timestamp with each data point. Such ordering of data points in time allows for identifying patterns, trends and seasonality over the course of time. In regard to our study, the time series data primarily consists of hourly timestamps paired with corresponding hit rates, representing the number of requests received by cloud servers over the period of an hour

Cloud computing environments present unique challenges for time series analysis due to the dynamic nature of resource utilization patterns. These patterns are characterized by periodic fluctuations (daily or weekly or even monthly cycles), sudden spikes in demand, and seasonal variations, requiring sophisticated forecasting approaches to ensure optimal resource allocation. Their fluctuations are a The accurate predic-

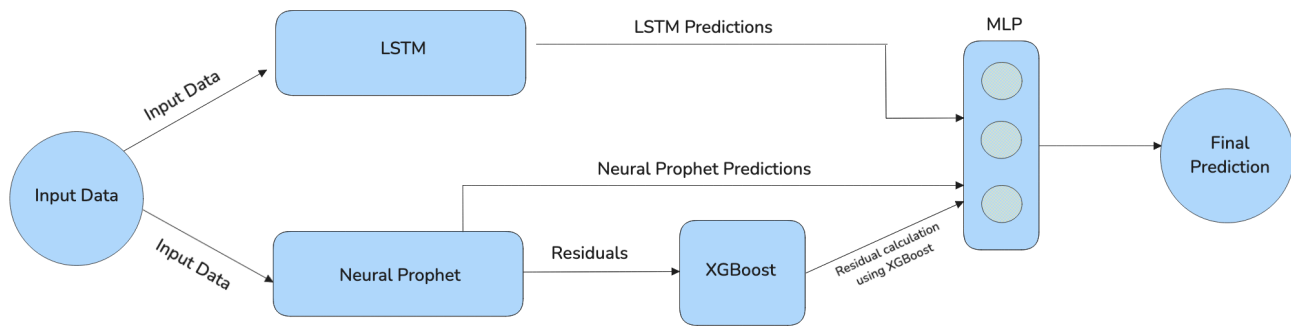


Figure 1. Proposed HybridTimeNet architecture integrating LSTM, Neural Prophet, and XGBoost with MLP for time series forecasting.

tion of cloud workload patterns is crucial for maintaining QoS and preventing Service Level Agreement (SLA) violations, and optimizing resource utilization. Traditional forecasting methods often struggle with the complexity of these workload patterns, necessitating advanced machine learning approaches that can capture both long-term dependencies and short-term variations in the data.

B. Foundational Models

The proposed architecture leverages three complementary machine learning models, each chosen for their specific strengths in handling different aspects of time series forecasting:

Long Short-Term Memory (LSTM) networks excel in capturing long-term temporal dependencies due to their specialized gated architecture. Such structure allows a selective approach to memory management which is useful in recognizing repeating features in cloud workload history[19].

Neural Prophet, is an advanced version of the Facebook Prophet model and it offers good handling of multiple seasonal and trend patterns. It can do so due to its neural network-based approach towards the time series data which has both global and local context trends[20].

XGBoost contributes to the ensemble as it can learn highly complex relationships as well as learn interactions among different features of the data. Building on its established gradient-boosting architecture, it further strengthens robust prediction accuracy with efficiency of operations[21].

C. HybridTimeNet Architecture: Integration and Performance Benefits

The proposed architecture presents a novel approach to time series forecasting by leveraging the complementary strengths of multiple models. This innovative framework combines Long Short-Term Memory (LSTM) networks, Neural Prophet, and XGBoost, integrated through a Multi-Layer Perceptron (MLP) for final prediction synthesis. The following models were shortlisted based on experimental data. We ran the individual models on the time series data: SVM, LSTM, Neural Prophet, Holts, XGBoost individually as well as various combinations of the them on the datasets mentioned in the paper. The

combination of Holts and LSTM was able to capture the trends but kept repeating the set and hence unable to capture the seasonality. LSTM and Neural Prophet were combined but had quite a high MAE and RMSE hence were not used as the final model.

As depicted in Fig. ??, there are two separate streams into which input data undergoes processing. Within the first stream, an LSTM network is employed. The structure of the LSTM affords it the capability to remember and apply concepts throughout extensive sequences, allowing it to be particularly effective in locating time patterns and other repeating features within the data set therefore it is inputted into the final MLP without other interferences. This helps mitigates the risk of short-term dependencies becoming redundant while ensuring that long-term structural patterns remain coherent and informationally significant. It also allows for the volatile short term fluctuations to be captured.

Concurrently, in the second stream, Neural Prophet, a neural network-based evolution of Facebook Prophet also takes in the input. It provides the complementary perspective to the LSTM's black box analysis by giving a transparent trend decomposition and explicit modelling of holidays and external regressors.

The third stream of input to the Multilayer Perceptron is fed by taking the residuals of Neural Prophet prediction which are passed to XGBoost. The residuals of the Neural Prophet predictions are calculated by finding the difference between the actual values and the predicted values. This creates a meta learning process which attempts to refine the recurring systematic prediction errors.

The architecture's differentiating feature is its novel integration layer, implemented as an MLP, which combines three distinct inputs:

- Pure LSTM's predictions for temporal pattern recognition
- The baseline prediction from Neural Prophet
- Residual calculations from XGBoost for non-linear pattern capture

Such an integration method gives the model a means of adaptively allocating weight to each of its constituents based on its historical performance. By figuring out the optimal

weights for their combinations during training, MLP creates an ensemble that outperforms each model alone.

The HybridTimeNet architecture alleviates the common drawbacks that occur in single-model architectures. It is true that LSTM networks are suitable for learning temporal relations, but their response to sudden changes may be insufficient. Neural Prophet addresses seasonality and trend changes very well; however, it may not address the complicated sequential patterns. XGBoost fills this gap by capturing non-linear relationships in the residuals. A more reliable and accurate prediction system is finally produced by the MLP's adaptive combination of these complimentary strengths.

V. SERVER PROVISIONING CALCULATION MODEL

The **Markovian Model of Queuing (M/M/c)** is used to determine the optimum number of servers necessary in attending to arriving requests with efficiency and meeting SLAs with minimum wastage of resources using the forecasted hit rate. The model is important in the cloud environment since resource allocation directly relates to service quality [22].

The MMC model is derived from queuing theory, particularly the **M/M/c** system, which refers to:

- **M**: Markovian arrival process (Poisson distribution of arrivals),
- **M**: Markovian service time (exponentially distributed service times),
- **c**: The count of servers present in the system.

A. How the MMC Model Works

- 1) **Input: Forecasted Hitrate (λ)**: The forecasted hitrate is derived from the time series forecasting model, serves as the arrival rate (λ) in the MMC model. This value expected number of requests to hit the system within a given time period.
- 2) **Service Rate (μ)**: The service rate (μ) represents the average rate at which a single server can process requests.
- 3) **Output: Required Number of Servers (c)**: The output of the optimal number of servers (c) required to maintain efficient operations based on the forecasted hitrate[23].

B. Key Metrics and Considerations

The MMC model also helps to evaluate several key performance metrics that affect decision-making in server allocation:

- 1) **Utilization Factor (ρ)**: This represents the ratio of the workload to the available system capacity, which indicates how efficiently the system resources are being utilized

$$\rho = \frac{\lambda}{c\mu} \quad (1)$$

- 2) **Response Time (W)**: The average total time a users request remains within the system, which includes both the waiting period in the queue and the actual service duration, is represented by W . Shorter response times are critical for satisfying SLAs and ensuring user satisfaction.

$$W_q = \frac{c^c \rho^{c+1} P_0}{\lambda c! (1 - \rho)^2} \quad (2)$$

The average time that a user's request waits in a queue before it is attended to is referred to as W_q processed.

$$W = W_q + \frac{1}{\mu} \quad (3)$$

- 3) **Probability of Idle Servers (P_0)**: This is the probability that no requests are in the system, and all servers are idle. While it might indicate efficient resource use, too high a value can signal under utilization of resources.

$$P_0 = \left(\frac{c^c \rho^c}{c! (1 - \rho)} + \sum_{n=1}^{c-1} \frac{(c\rho)^n}{n!} \right)^{-1} \quad (4)$$

In particular, the parameter ρ has to be less than 1 for stability. The objective of the performance model, therefore, is to determine the minimum number of servers (c) at each time period so that the system remains stable, i.e., $\rho < 1$, while still meeting the response time (W) commitments of the SLA.

The system makes provisions for automatic scaling on an hourly basis. This auto-scaling process uses the M/M/c queuing model to predict the average server load for the coming hour and results in service provision accordingly.

An hourly scaling interval selection can be explained by two main reasons. First, adjusting resources too frequently risks the stability of the system, similar to the problems encountered with reactive scaling methods. Secondly, due to a number of reasons, such as some suppliers of cloud infrastructure charging at least one hour, modifying the allocation on a scale of hours would not make any economic sense. While this particular study applies an hour period for automatic scaling, the time frame does not have to be fixed, as such mechanical installation can be restructured to accommodate other time frames as the nature of the system requires.

VI. EXPERIMENTAL EVALUATION AND RESULTS ANALYSIS

A. Datasets

Two distinct datasets were employed to test the resilience and flexibility of the proposed architecture. The first set contains hourly load time series of the cloud system from the Complutense University of Madrid across 28 consecutive days, with twenty-four hourly observations in each day making it a sum of 672 observations. This 28-day timeframe was used to train the models, and on the 29th day, their predictions were made[24].

The capacity of the model to deal with downtimes in the system and make responses appropriately has been validated with the help of the second NASA dataset collected from the Kennedy Space Center. Server requests and responses over 30 days with the 31st used for testing were provided (683 observations). Noteworthy, this dataset contained a downtime on day 2, which provided an opportunity to examine the degree to which the model withstands disruption of systems - a vital aspect in real-world cloud applications.

TABLE I
COMPARISON OF MAE AND RMSE FOR DIFFERENT MODELS (MADRID DATASET)

Model	MAE	RMSE
Neural Prophet	780.742	981.355
SVM	1133.00	1364.00
Holt's	853.932	1041.115
LSTM	1309.186	1609.084
HybridTimeNet	784.29	1066.46

B. Evaluation Metrics

The models' performance was evaluated using two standard metrics:

- **Mean Absolute Error (MAE):** An essential measure that assesses the accuracy of a prediction through the computation of the simple average of the absolute prediction mistakes. This metric gives equal weight to each error and gives a simple measure of how much the model is away from the true value.

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (5)$$

- **Root Mean Squared Error (RMSE):** An error that gives more weight to larger deviations by squaring the individual errors which are then averaged. Since these individual squared errors are first averaged, and then a square root operation is conducted, RMSE is a measurement that pays attention to large errors in prediction and is useful to applications where larger errors are especially undesirable.

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n [y_i - \hat{y}_i]^2} \quad (6)$$

C. Results and Analysis

1) *Forecasting Analysis:* Initial evaluations of individual models revealed that Neural Prophet consistently outperformed other standalone approaches in terms of RMSE and MAE for the Madrid dataset, as shown in Table I. However, it would tend to underpredict the number of servers required, which could potentially lead to resource constraints and Service Level Agreement (SLA) Violations in practical applications.

The proposed HybridTimeNet architecture shows commendable performance measures as well as solves the underprediction limitation of Neural Prophet. Although HybridTimeNet would not be able to exceed Neural Prophet's raw numerical metrics, it was able to outdo the former in trend capture capabilities and has a bias toward slight overprediction which is more desirable in cloud resource allocation applications, as it reduces Service Level Agreement (SLA) breaches and improves Quality of Service (QoS).

Table II presents the results of the NASA dataset which indicates the resilience of HybridTimeNet towards the system's downtimes and other such anomalies. The model maintained

TABLE II
COMPARISON OF MAE AND RMSE FOR DIFFERENT MODELS (NASA DATASET)

Model	MAE	RMSE
Neural Prophet	885.217	1232.504
SVM	1199.106	1674.663
Holt's	874.366	958.366
LSTM	605.902	822.644
HybridTimeNet	792.5928	986.5781

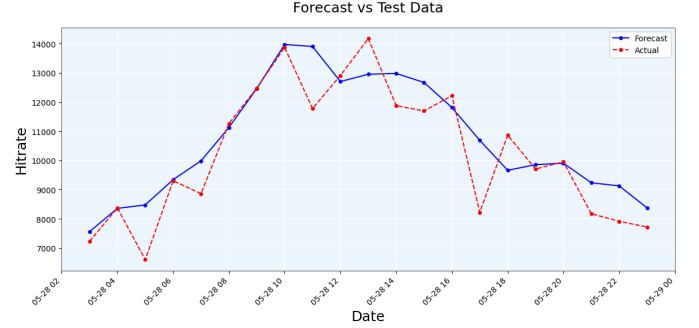


Figure 2. Prediction comparison for Madrid dataset showing actual vs. predicted values across different models.

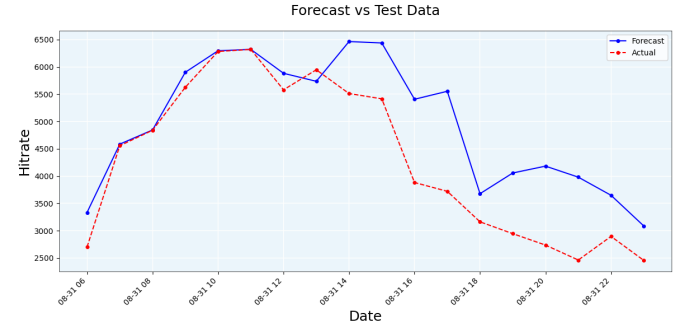


Figure 3. Prediction comparison for NASA dataset demonstrating model performance during system downtime periods.

consistent performance despite the presence of service interruptions, demonstrating its suitability for real-world applications.

2) *Resource Allocation Analysis:* In order to compute the necessary number of backend resources, an M/M/c queueing performance model was used. This information was integrated into the previously mentioned forecasting models of server load so that the models they produced did not overestimate the number of resources from the perspective of SLA-compliant response time. We then performed a 24 hour time span analysis of the two prediction resource management models: one derived from the actual observed load and the other from the predicted model. This comparison, for each forecasting model and hourly period, identified whether the server was over-provisioned, under-provisioned, or correctly provisioned.

M/M/c Model Parameters

- **Arrival Rate (α):** Average server load, which can be derived from actual and forecasted data with requests per

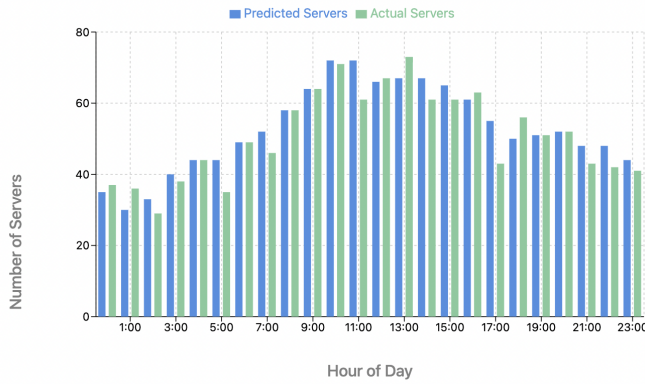


Figure 4. Comparison of actual server deployment versus M/M/C queueing model predictions over a 24-hour period.

second over the 24-hour period.

- Service Rate (μ): The processing capability of one backend server is taken to be fixed at 200 requests per second, which is common for middle-sized servers like Amazon EC2 instances which serve content that is dynamic.
- System Utilization (ρ): Must be < 1 for system stability; thus, appropriate resource provisioning must be done.
- Average Queue Time (W_q): It must not exceed 2.5 ms, which is half of the minimum response time as taken as the bench mark.
- Average Response Time (W): It is given by

$$W = W_q + \frac{1}{\mu} \quad (7)$$

is limited by to 7.5 ms in this research.

Using the actual and forecasted load values (λ), the number of backend servers to be provisioned for guaranteed stability of the system ($\rho < 1$) while meeting the SLA-imposed response time constraints ($W \leq 7.5$ ms) was computed by using the M/M/c queueing model.

Evaluation Metrics

Forecasting models were compared with respect to the following metrics:

- Number of Provisioned Resources: It gives the efficiency of resource allocation.
- SLA Violations: Response times higher than 7.5 ms.
- Unserved Requests: Those taking longer than 1 second.

The ideal resource allocation, from a user perspective, focuses on both, minimizing the cost of the infrastructure while keeping the service level at, at least, the minimum acceptable level as defined by the Service Level Agreement and the number of successfully handled requests. On the one hand, under-allocation results in reduced infrastructure costs but contributes to higher network service disruptions and number of failed requests. Over-allocation on the other hand, assists in providing a more reliable service but at a greater cost.

VII. CONCLUSION AND FUTURE WORK

The new approach discussed in this paper HybridTimeNet is a hybrid architecture combining LSTM, Neural Prophet and XGBoost using MLP integration layer for better prediction in cloud workloads. The results of the experimental evaluation were based on the datasets from the Complutense University of Madrid, as well as the NASA Kennedy Space Center and revealed the effectiveness of the model in real-world scenarios, especially when it was necessary to make resource decisions during the downtime of the system. Although the numerical metric scores were in favour of Neural Prophet, HybridTimeNet's slight overprediction tendencies turned out to be beneficial when deploying cloud resources, resulting in fewer violations of SLAs and better QoS. An interesting direction for future work may include the study of the possibilities of real-time adaptation, attention mechanisms within the integration layer, and such contextual features as scheduled maintenance windows so that the predictive model performs accurately even during anomalous periods.

REFERENCES

- [1] Masdari, M., Khoshnevis, A. (2019). A survey and classification of the workload forecasting methods in cloud computing. *Cluster Computing*. doi:10.1007/s10586-019-03010-3
- [2] Tokura, Tomohisa. (2023). Cognitive Cloud Computing: Proactive Fault Prediction and Resource Allocation with AI. 10.13140/RG.2.2.19107.67366
- [3] Mahbub, N.I., Sinthia, A.K., Jeon, M., Park, J., Huh, EN. (2025). ADAPT: Attention-Driven Domain Adaptation for Inter-cluster Workload Forecasting in Cloud Data Centers. In: Wang, Y., Zhang, L.J. (eds) CLOUD Computing – CLOUD 2024. CLOUD 2024. Lecture Notes in Computer Science, vol 15423. Springer, Cham. https://doi.org/10.1007/978-3-031-77153-8_6
- [4] Shahid, Muhammad Asim Islam, Noman Alam, Muhammad Mazliham, M. Musa, Shahrulniza. (2021). Towards Resilient Method: An exhaustive survey of fault tolerance methods in the cloud computing environment. *Computer Science Review*. 40. 100398. 10.1016/j.cosrev.2021.100398.
- [5] Lorido-Botran, T., Miguel-Alonso, J. Lozano, J.A. A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments. *J Grid Computing* 12, 559–592 (2014). <https://doi.org/10.1007/s10723-014-9314-7>
- [6] Siami-Namini, Sima, Neda Tavakoli, and Akbar Siami Namin. "A comparison of ARIMA and LSTM in forecasting time series." 2018 17th IEEE international conference on machine learning and applications (ICMLA). Ieee, 2018.
- [7] Chimmula, Vinay Kumar Reddy, and Lei Zhang. "Time series forecasting of COVID-19 transmission in Canada using LSTM networks." *Chaos, solitons fractals* 135 (2020): 109864.
- [8] Lakshminarayanan, Sai Krishna, and John P. McCrae. "A Comparative Study of SVM and LSTM Deep Learning Algorithms for Stock Market Prediction." *AICS*. 2019.
- [9] S. Taylor and B. Letham, "Forecasting at scale," 2017, doi: 10.7287/peerj.preprints.3190v2.
- [10] M. J. A. Shohan, M. O. Faruque, and S. Y. Foo, "Forecasting of electric load using a hybrid LSTM-Neural Prophet model," **Energies**, vol. 15, no. 6, p. 2158, 2022, doi: 10.3390/en15062158.
- [11] K. Albeladi, B. Zafar, and A. Mueen, "A Novel Deep-learning based Approach for Time Series Forecasting using SARIMA, Neural Prophet and Fb Prophet" **Preprints**, 2023, doi: 10.20944/preprints202311.0794.v1.
- [12] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. Association for Computing Machinery, New York, NY, USA, 785–794, 2016.
- [13] Z. Xia, S. Xue, L. Wu et al., "ForeXGBoost: passenger car sales prediction based on XGBoost," *Distrib Parallel Databases*, vol. 38, pp. 713–738, 2020.

- [14] P. Nawrocki and M. Smendowski, "Long-Term Prediction of Cloud Resource Usage in High-Performance Computing," in *Computational Science – ICCS 2023, Lecture Notes in Computer Science*, vol. 14074, Springer, Cham, 2023.
- [15] J. Cai, H. Cai, Y. Cai, L. Wu and Y. Shen, "Short-term Forecasting of User Power Load in China Based on XGBoost," in *2020 12th IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC)*, Nanjing, China, 2020, pp. 1-5.
- [16] E. Aguilar Madrid and N. Antonio, "Short-Term Electricity Load Forecasting with Machine Learning," *Information*, vol. 12, no. 2, pp. 50, 2021.
- [17] Li, Chen, et al. "Power load forecasting based on the combined model of LSTM and XGBoost." *Proceedings of the 2019 the international conference on pattern recognition and artificial intelligence*. 2019.
- [18] Duan, Ganglong Dong, Jiayi. (2024). Construction of Ensemble Learning Model for Home Appliance Demand Forecasting. *Applied Sciences*. 14. 7658. 10.3390/app14177658.
- [19] Shahzad Muzaffar, Afshin Afshari, Short-Term Load Forecasts Using LSTM Networks, *Energy Procedia*, Volume 158, 2019, Pages 2922-2927, ISSN 1876-6102, <https://doi.org/10.1016/j.egypro.2019.01.952>.
- [20] M. Daraghme, A. Agarwal, R. Manzano and M. Zaman, "Time Series Forecasting using Facebook Prophet for Cloud Resource Management," *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, Montreal, QC, Canada, 2021, pp. 1-6, doi: 10.1109/ICCWorkshops50388.2021.9473607.
- [21] Zhang, Lingyu Bian, Wenjie Qu, Wenyi Tuo, Liheng Wang, Yunhai. (2021). Time series forecast of sales volume based on XGBoost. *Journal of Physics: Conference Series*. 1873. 012067. 10.1088/1742-6596/1873/1/012067.
- [22] Zhanyou Ma, Shanshan Guo, Rong Wang, The virtual machines scheduling strategy based on M/M/c queueing model with vacation, *Future Generation Computer Systems*, Volume 138, 2023, Pages 43-51, ISSN 0167-739X, <https://doi.org/10.1016/j.future.2022.08.001>.
- [23] Li, Xinzichen. (2024). Performance Evaluation of M—M—1 Queueing Models in Cloud Computing Environments. *Highlights in Science, Engineering and Technology*. 85. 841-848. 10.54097/1xcw8y74.
- [24] R. S. Montero, E. Huedo, and I. M. Llorente, "Efficient resource provisioning for elastic cloud services based on machine learning techniques," *Journal of Cloud Computing**, vol. 8, no. 1, pp. 1-18, 2019.