SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V13I2P114

Volume 13 Issue 2, 180-194, February 2026

Original Article

Hybrid Ensemble Approach for Accurate Workload Prediction in Dynamic Cloud Environments

Naimisha S. Trivedi[1], Ajay N. Upadhyaya[2]

[1]*Gujarat Technological University, Gujarat, India.*
[2]*Department of Computer Engineering, SAL Engineering & Technical Institute, SAL Education, Gujarat, India.*

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

Received: 21 November 2025 Revised: 22 December 2025 Accepted: 23 January 2026 Published: 20 February 2026

Abstract *- Time series forecasting is important for cloud computing because it helps keep services running, ensures that Quality of Service (QoS) is maintained at a high level even when workloads change, and makes sure that resources are used well. In the actual world, cloud workloads do not stay the same all the time. Rather, they evolve throughout time, exhibiting cyclical trends, nonlinear fluctuations, and periods of elevated demand. This complicates the utilization of conventional forecasting methodologies. This study presents a hybrid ensemble-based approach for short-term cloud workload prediction that enhances adaptability through workload-aware modeling to tackle this issue. The method employs feature extraction and K-means clustering, complemented by a dynamically weighted ensemble of Prophet, Long Short-Term Memory (LSTM), and Linear Regression models. NASA HTTP server logs and ClarkNet WWW datasets are used to test the architecture with different window sizes and prediction horizons. Experimental results indicate that modest observation windows and short-term perspectives yield the optimal correlation between response time and prediction stability. On the other hand, the hybrid ensemble always does better than single models in different types of workloads. These results show how effectively the suggested technique works for proactive provisioning of cloud resources and reliable auto-scaling in environments that are cloud-native.*

Keywords *- Cloud computing, Hybrid ensemble model, Machine Learning, Time series forecasting, Workload prediction.*

1. Introduction

Cloud computing has transformed modern information technology by enabling scalable, on-demand access to resources hosted in data centres. Cloud Service Providers (CSPs) face the critical challenge of ensuring resource availability in a flexible, cost-effective manner to maintain high Quality of Service (QoS) as demand grows. Accurate workload prediction is essential for proactive resource allocation, preventing over- or under-provisioning, and enhancing operational efficiency in dynamic cloud environments. Inaccurate forecasting can lead to severe consequences, as demonstrated by major service outages at Amazon Web Services (AWS) in 2012 and 2017 due to unpredicted workload spikes in the US-East-1 region, which caused resource depletion and cascading failures. Similarly, Microsoft Teams struggled with service disruptions in 2020 during the COVID-19 pandemic, unprepared for the sudden surge in users driven by the shift to remote work.

The core problem in cloud computing is predicting short-term workload metrics, such as request rates, 10 to 30 minutes ahead, to enable data centres to dynamically adjust resources, minimize latency, and prevent overload. Cloud workloads are inherently complex, characterized by linear trends, seasonal cycles, and unpredictable nonlinear bursts, driven by diverse user behaviours, multi-tenancy, and emerging edge computing paradigms. These characteristics make it hard to anticipate workloads because both basic and advanced methods have trouble modeling patterns that are not stationary or linear. [1, 2].

Existing workload prediction methods fall short in addressing the complexity of cloud workloads. Traditional statistical models, such as ARIMA and exponential smoothing, assume regularity in data and fail to capture sudden changes or long-term dependencies common in web traffic or virtual machine usage [3, 4]. Machine learning and deep learning approaches, such as Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), and Transformers, improve the modeling of sequential and non-linear patterns but face limitations. For instance, LSTMs are prone to local optima in long sequences [5], while Transformers suffer from high computational complexity, hindering real-time applications [6]. Hybrid models, like LSTM-Prophet, combine strengths but often rely on static weighting or limited feature decomposition, failing to fully exploit multi-scale relationships or adapt to concept drift [7]. Moreover, uncertainty-aware methods, such as Bayesian

networks, provide confidence intervals but lack real-time retraining capabilities [8]. These shortcomings highlight the need for a flexible, adaptive model that integrates multi-frequency analysis and dynamic feature extraction to achieve robust performance across diverse cloud workloads.

Despite extensive research, there remains a lack of adaptive, workload-aware forecasting frameworks that (i) explicitly account for different workload characteristics, (ii) dynamically select or weight prediction models based on observed workload behaviour, and (iii) remain computationally feasible for short-term, real-time cloud resource management. Most existing approaches apply a single model or a fixed hybrid strategy uniformly across all workloads, ignoring the intrinsic diversity and evolving nature of cloud traffic patterns.

To fill these shortcomings, we propose a clustering-enhanced hybrid ensemble model for short-term workload prediction in cloud computing. This model uses K-means clustering to categorize workloads into seasonal, non-linear, and steady types based on features like variance, periodicity, and rate of change. Then, it uses a dynamically weighted combination of Prophet, LSTM, and Linear Regression to predict these patterns. For mixed workloads, it uses inverse-distance weighting to make the predictions more accurate. "Hybrid" denotes that several ways of making predictions are used together. "Ensemble" means that these ways are merged into one framework through dynamic weighting.

Key Contributions:

- A novel clustering mechanism that automatically segments workloads, changing to diverse patterns without manual involvement.
- A weighted ensemble framework that uses Prophet for seasonal trends, LSTM for non-linear dynamics, and Linear Regression for steady patterns. The weights are based on how close the clusters are to each other.
- Extensive experiments on NASA HTTP Logs and ClarkNet WWW datasets, benchmarking against state-of-the-art baselines to validate performance.
- A multi-scale validation framework testing robustness across various forecasting horizons and window sizes.

The paper is organized as follows: Section 2 discusses related work; Section 3 explains the methodology; Section 4 presents the experiments and analysis; and Section 5 provides conclusions and future directions.

2. Related Work

In the last ten years, there has been a lot of improvement in research in predicting cloud workloads. It has progressed from conventional statistical models to advanced methodologies that utilize clustering, hybrid, and uncertainty-aware strategies to deal with real-world non-stationarity and various deployment constraints. Early foundational work, like ARIMA-based SaaS workload predictors, achieved accuracy rates of 91%, supporting the efficacy of conceptualizing workload as a univariate time series for anticipatory resource scaling [3]. ARIMA, ARMA, and similar models proficiently handle linear seasonality; however, they are insufficient for handling regime shifts and lack the ability to rapidly adapt to sudden shifts [9].

This knowledge resulted in the exploration of feature-engineered and machine-learning techniques for short-term forecasting. Support vector machines and neural network hybrids, such as SVM with backpropagation networks, were utilized on physical-machine workload traces, demonstrating superior resistance to disruption induced by interference as compared to conventional models [10]. At the same time, request-pattern elements such as temporal locality and burst-window recognition were introduced to predict SaaS spikes. This made it possible to provision resources ahead of time, which worked better than simple smoothing methods [9]. These methods, on the other hand, depended a lot on manual feature extraction and had trouble when workload behaviours changed. [4] proposed the Technocrat ARIMA and SVR Model (TASM), which integrates statistical ARIMA with Support Vector Regression to improve workload prediction for web applications in cloud environments. The method works well for capturing short-term and linear trends, but it is not very effective at modelling highly non-linear or bursty workload patterns, which shows that hybrid deep learning models are needed.

Deep sequence models gained popularity at the same time because of their ability to extract temporal dependencies from multivariate inputs. CPU, memory, and I/O telemetry were successfully converted into supervised inputs for an improved GRU architecture using the esDNN model. This captured both short- and long-term dependencies and significantly reduced errors on Alibaba and Google traces, which helped with auto-scaling actions [11].

Attention mechanisms made multivariate forecasting even better. For instance, MAG-D used gated RNNs with attention to dynamically focus on important lagged resource signals. In data centre environments, this worked more effectively than just LSTM or GRU models [12]. Likewise, sequence-to-sequence attention mechanisms transformed workload forecasting into a translation task, applying cumulative validation to address distributional shifts and enhancing multi-step horizon accuracy on Google cluster data [13].

However, the characteristics of a workload often include both periodic trends and sudden bursts. To deal with these mixed conditions, hybrid forecasts that combine trend-seasonality models (such as Prophet) with deep residual learners emerged. For Kubernetes microservices with daily

traffic, hybrid Prophet and LSTM pipelines were easier to understand and more reliable than deep models by themselves, especially in sparse regimes [14, 15]. TempoScale proposed a framework that explicitly integrates short-term fluctuations with long-term trends, achieving elastic scaling that remains robust across regime transitions [16].

More recently, Transformer-based architecture has gained popularity for its capacity to model long-term dependencies and support parallel computation. A Wasserstein adversarial Transformer used for cloud forecasting demonstrated similar or enhanced accuracy with reduced inference latency compared to RNN [6]. Dual-branch, frequency-aware Transformers that use wavelet priors seem to be able to capture high-frequency bursts without losing trend fidelity [17]. The Informer model is a type of transformer that has been tested to see how well it can predict CPU usage in cloud data centers. The results show that the Informer model, which considers both long-range dependencies and sequence ordering, always does better than vanilla transformers [18]. However, transformers need a lot of data that is often hard to get or too expensive to collect [19].

Due to the inherent risks of auto-scaling, the discipline has transitioned from point forecasting to probabilistic and uncertainty-aware methodologies. Bayesian neural network variants and deep probabilistic learners' predictive model variance, which gives us calibrated intervals that significantly lower SLO violations on workloads that use multiple resources [8]. Predictive variance modelling for tail risks [20] emphasized that precise point estimates alone may render systems susceptible to infrequent, high-impact demand spikes. Furthermore, multivariate forecasting models that concurrently analyze CPU, memory, I/O, and network consumption, exemplified by the multilayer network introduced by [21], consistently outperform univariate approaches over different horizons. The study we conducted earlier demonstrates that optimization-based workload predictions achieve up to 25% greater accuracy in anticipating cloud resource requirements, thereby improving proactive resource planning, as confirmed across several workload scenarios.

More and more recent literature includes predictors for auto-scaling platforms to make forecasting work. [22] examined time-series-driven scaling policies in Kubernetes, finding that accurate short-term forecasts reduce oscillations and enhance the stability of HPA/VPA in microservice environments. Several researchers have expressed concerns regarding the accuracy-overhead trade-off. They recommend using methods like model compression, distillation, and selective Retraining to maintain the quality of the forecasts high while keeping the inference overhead low [23, 24]. It is widely known that out-of-distribution generalization has real-world limits; even the best models may not do as well as optimized LSTMs at forecasting new workloads. This shows how important it is to choose and adapt models that are aware of workloads [25].

Evaluation methods have changed at the same time as modeling methods have improved. Google, Alibaba, and Huawei serverless platforms are examples of public data sources that make it easy to test things in the real world. Cumulative and blocked validation [13] are two methods that keep data from getting out over time and overfitting. Forecasting metrics have developed; in addition to MAPE and RMSE, professionals also use pinball loss and interval coverage measures to check probabilistic forecasts [8]. [26] used unsupervised clustering to group workload traces before making probabilistic forecasts, which made both the process faster and more accurate. [24] came up with MSFS, which groups workloads together and then models them by group. This makes the forecast inaccurate and the cost lower. [27] created an ensemble clustering process that found hidden workload categories, which led to better forecasts. [18] used BIRCH clustering to group different types of machines, which made it possible to create customized predictors for each category that were better than global models. [2] suggested a way to classify traces into periodic, bursty, or steady types to make auto-scaling easier and increase elasticity. [28] used clustering and workload categorization together in ML WPStreamCloud to make short-term cloud provisioning easier and more accurate. [29] used KPI-based clustering for HPC tasks and then used bespoke predictors to reduce the error in the forecasts. [30] offered dual clustering for multivariate workloads, which made predictions more accurate in cloud environments with a lot of different types of workloads.

These many fields of research all support our method when they are put together. Using variance, periodicity, and ROC to characterize features is in line with empirical trace analysis and makes it possible to type workloads in a way that makes sense. Grouping and sorting workloads into categories makes it easier to specialize and cuts down on modeling differences. Model selection that assigns workloads to Prophet, LSTM, or linear models and optionally blends predictions with inverse-distance weighting combines interpretability, nonlinearity, and efficiency. Lastly, forecasting that takes uncertainty into account creates buffers for operational risk to protect against rare, high-impact events. So, our integrated pipeline gives the user a principled, well-supported mix of the best cloud workload forecasting methods currently available.

3. Methodology

The methodology for our proposed clustering-enhanced hybrid ensemble model is designed to provide accurate and adaptive short-term workload predictions in cloud computing environments. This section explains how feature extraction, clustering, classification, and weighted prediction all operate together. The main idea behind the general framework is time-

series analysis. It employs ensemble learning to make predictions better and feature engineering to find patterns. To make sure that others can reproduce our work and appreciate it, we discuss each part in detail, including the problem specification, mathematical formulations, computing complexity, and logic.

3.1. Problem Definition

Given a univariate time series $\{x_t\}_{t=1}^{T}$, where $x_t \in R$, the goal is to learn a parametric predictor f_θ that connects each length-ω window $W_t = x_{t-\omega+1}, \dots, x_t$ to an H-step forecast $\hat{y}_t$ by minimizing empirical risk across all acceptable windows $t \in \{\omega, \omega+1, \dots, T-H\}$. This is done using a direct multi-output strategy and squared error loss, with optional regularization to improve generalization. When attempting to create multi-step time series forecasts, the user must determine the proper Window size (w) and forecast Horizon (h). Resource provisioning can change if the window is too short or the horizon is inappropriate. But if the user chooses the correct one, it will accurately catch trends and offer forecasts.

3.2. Data Pre-Processing

The first step in the methodology is data pre-processing, which turns basic workload logs into a format that can be used for predictive modeling. Because system logs are noisy, cloud workloads like HTTP request rates from the NASA and ClarkNet datasets have missing values, outliers, and sample intervals that are hard to estimate. To focus on actual loads, the process starts by looking at the log files to collect request counts with timestamps and only including those that were successful (status < 400). When the user uses grouping functions, aggregation happens every minute. To make sure the data is valid across time and to cut down on noise, we use linear interpolation to fill in missing values and a 3-sigma threshold to cap outliers during data pre-processing. This stops the challenges that can happen with Savitzky-Golay filtering and other methods when the user uses too much regularization.

We utilize a multi-step modification to cope with non-stationarity, which is a common issue with cloud data where the mean and variance change over time. The first step is the log transformation $x_t' = log(1 + x_t)$, keeps the variance stable and takes care of zeros, which stops division-by-zero errors from happening later. This modification works best for distributions that are positively skewed since it makes them look more normal and decreases the effect of outliers.

Next, seasonal differencing with a 1440-minute lag (daily cycle) eliminates the problem of periodic trends: $x_t'' = x_t' - x_{t-1440}$. Finally, first-order differencing eliminates any remaining trends: $x_t''' = x_t'' - x_{t-1}''$. The Augmented Dickey-Fuller (ADF) test ($p < 0.05$ for stationarity) and the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test (to see if a time series is stationary around a constant (level stationarity) or a trend (trend stationarity)) are used to check for stationarity. The z-score threshold ($|z| > 3$) is used to cap outliers. Before clustering, the collected features (variance, periodicity, and rate of change) are normalized to the range [0, 1] using min-max scaling. This makes sure that the scale does not matter and that clustering happens faster. The pre-processing pipeline makes the data a little shorter, but it retains the time integrity. The user may now use the pre-processed time series x_t to get features, which will help the model operate better with varied datasets.

3.3. Feature Extraction

Feature extraction turns the pre-processed time series into a small version that indicates essential variations in workload, making it easier to group and classify. We use a sliding window that is 60 minutes broad to find three things:

Variance (Var_t) measures the volatility of request counts over a time window. The equation is $\mathrm{Var}_t = \frac{1}{h-1}\sum_{i=t-h+1}^{t}\left(\mathrm{requests}_i - \bar{r}_t\right)^2$, Where $\bar{r}_t = \frac{1}{h}\sum_{i=t-h+1}^{t}\mathrm{requests}_i$ is the mean request count over the window $[t-h+1, t]$, and $h = 60$ (1 hour in minutes).

Periodicity (Per_t) quantifies the strength of seasonal patterns using the Auto Correlation Function (ACF).

$$\mathrm{Per}_t = \max_{1\le k\le p}\left|\frac{\sum_{i=t-h+1}^{t-k}(\mathrm{requests}_i-\bar{r}_t)(\mathrm{requests}_{i+k}-\bar{r}_t)}{\sum_{i=t-h+1}^{t}(\mathrm{requests}_i-\bar{r}_t)^2}\right|$$

Where p = 1440 (representing daily seasonality in minutes, i.e., 24 hours × 60 minutes), and ACF computes the correlation of the time series with lagged versions of itself. And Rate of Change (Roc_t) captures the dynamic trend of the workload. $\mathrm{Roc}_t = \frac{\mathrm{requests}_t - \mathrm{requests}_{t-1}}{\mathrm{requests}_{t-1}+\epsilon}$ where ϵ = 0.01 prevents division by zero.

Features are standardized using $\mathrm{x}_t^{\mathrm{std}} = \frac{\mathrm{x}_t - \mu_x}{\sigma_x}$ where μ_x and σ_x are the mean and standard deviation of the feature vectors across the training dataset, which forms a vector $f_t = [(\mathrm{Var}_t, \mathrm{Per}_t, \mathrm{Roc}_t]^T$, which is robust to scale variations. This feature set builds on classic variance-periodicity frameworks by adding the rate of change, making them more sensitive to sudden changes. The extraction is $O(n * w)$, which is good for large databases, and it makes the data prepared for clustering by demonstrating traits that are specific to patterns.

3.4. Workload Clustering

K-means, an unsupervised method that divides data by minimizing intra-cluster variance, is used by workload clustering to arrange time steps into different categories, such as seasonal, non-linear, and stable. We choose K=3 because

tests demonstrate that three clusters are enough to highlight the variety in datasets like NASA (seasonal) and ClarkNet (non-linear). Initialization uses k-means++ to select diverse centroids, reducing sensitivity to random starts. The objective function is $J = \sum_{i=1}^{K} \sum_{\mathbf{x}_t^{\text{std}} \in S_i} \left\| \mathbf{x}_t^{\text{std}} - \mathbf{c}_i \right\|^2$, where S_i is the set of feature vectors in cluster i, $\mathbf{c}_i$ is the centroid of cluster i, and K = 3 (for seasonal, non-linear, and steady clusters). Also, $\left\| \mathbf{x}_t^{\text{std}} - \mathbf{c}_i \right\|$ finds the Euclidean distance from a feature vector to each centroid and keeps doing it until it converges. Cluster interpretation is post-hoc: centroids with high periodicity are linked to seasonal patterns, high variance/rate of change to non-linear patterns, and low values to steady patterns. We utilize the silhouette score (0.5–0.7 on our data) to make sure that the clusters are well-separated and check for authenticity. It also fills in the gaps in pattern-specific prediction by allowing attach models to specific patterns.

3.5. Workload Classification

Workload classification gives prediction models dynamic weights based on how close they are to cluster centroids. This lets the user choose models in a soft, flexible way without strict limits. This stage expands on clustering by figuring out how closely the features of a time step match each pattern. This lets the model mix contributions in a way that makes sense.

For each time step t, distances are calculated as $d_i(t) = \left\| \mathbf{x}_t^{\text{std}} - \mathbf{c}_i \right\|^2$, measuring Euclidean separation from centroid k. Weight Computation ($w_i(t)$) Calculates weights for ensemble prediction based on inverse Euclidean distances to cluster centroids. This ensures that the model closest to the workload's characteristics (lowest $d_i(t)$, highest $w_i(t)$ contributes more, reducing prediction errors compared to a uniform average. The weights are computed as $w_i(t), w_i(t) = \frac{1/(d_i(t)+\epsilon)}{\sum_{j=1}^{3} 1/(d_j(t)+\epsilon)}$ where $w_1(t)$ -Weight for the seasonal cluster, used for Prophet predictions, $w_2(t)$ - Weight for the non-linear cluster, used for LSTM predictions and $w_3(t)$ - Weight for the steady cluster, used for LR predictions. ϵ = 0.01 ensures numerical stability, and $\sum_{j=1}^{3} w_i(t) = 1$.

In ambiguous cases, predictions from multiple models are combined using a weighted ensemble, where each model's weight is proportional to the inverse of its validation RMSE, ensuring that models with lower error contribute more strongly to the final forecast.

The classification is efficient $(O(n * K), K = 3)$ and generic, avoiding fixed thresholds to maintain adaptability across datasets. It extends static classification by providing continuous probabilities and handling mixed workloads in which a time step may exhibit partial seasonality and nonlinearity. This approach mitigates overfitting in single models and improves generalization, as weights evolve with data shifts.

3.6. Prediction Models

The prediction models form the core of the ensemble; each tailored to a workload pattern for specialized forecasting. Prophet is a time-series forecasting model designed to capture trend and seasonality using an additive decomposition framework. Prophet handles seasonal patterns with an additive model $P_{Prophet}(t) = g(t) + s(t) + \in_t$, where g(t) is the trend (piecewise linear), s(t) is seasonality (Fourier series for daily cycles), and $\in_t$ is error. This model is inspired by the architecture described in [31].

Long Short-Term Memory (LSTM) captures non-linear dynamics using a recurrent architecture: an input layer, two LSTM layers, and a dense output layer. Gates manage information flow: $forget\ f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$, $input\ i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$, $output\ o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$, $cell\ state\ c_t = f_t c_{t-1} + i_t tanh(Wc[h_{t-1}, x_t] + b_c)$, hidden state $h_t = o_t tanh(c_t)$. Here f_t, i_t and o_t denote the forget, input, and output gates, respectively, which regulate the flow of information. W_f, W_i, W_{fo} and W_c are the weight matrices, while and b_f, b_i, b_o and b_c are the bias terms. The cell state c_t preserves long-term memory, and the hidden state h_t captures the output representation at time t. This design is inspired by the architecture described in [32].

Linear Regression (LR) fits steady patterns: $P_{LR}(t) = \beta_0 + \sum_{i=1}^{w} \beta_t x_{t-i}$, utilizing least squares on lagged inputs. In this case, β_0 is the intercept, β_i are the regression coefficients, w is the window size (the number of lagged inputs), and x_{t-i} are the workload values from the past. This study chooses Linear Regression because it is good at finding and forecasting consistent patterns in workloads. It can show linear relationships in historical request trends, which makes it useful for estimating workloads that stay the same. This group of three models combines both variety and training efficiency.

3.7. Ensemble Prediction

The selection of prediction models is influenced by the intrinsic constraints of individual techniques in managing the different features of cloud workloads, which display a combination of seasonal patterns, non-linear spikes, and steady trends [1]. In situations that are not stationary, baseline models like ARIMA and SVR often have problems since they depend on the assumptions of stationarity and linearity, respectively [4]. Deep learning models like TCN and Bi-LSTM are good at finding temporal relationships, but they are expensive to run and overfit when working with bursty data [15]. LSTM, Prophet, and LR all do well on their own with specific patterns, such as non-linear, seasonal, and steady, but they cannot manage mixed workloads, which makes the metrics unreliable [14]. Following [33], ensemble prediction is chosen to combine several extreme learning machines with the best weights, which greatly improves accuracy and resilience when predicting virtual machine resource requests.

In contrast, our ensemble prediction uses a per-window inverse error weighting system that changes the contributions of each model based on how well it has done recently. This method works well with the cluster-based weights that come from K-means proximity. It fixes the problem of static weighting by using empirical error feedback. For each prediction window, a deque data structure keeps a rolling buffer of the latest M (default 10) windows' Root Mean Squared Errors (RMSE) for each model (Prophet, LSTM, and Linear Regression). To get the inverse error weight for model k, use the formula. $inv\_error_k = \frac{1}{recent\_rmse_k + \epsilon}$, where ϵ=0.01 avoids division by zero. Then, normalize the weights across models by using the formula $inv\_error\_weights_k = \frac{inv\_error_k}{\sum inv\_error_j}$. These weights are merged with cluster weights w_k using a regularization parameter α (default 0.5): ⟦ $final\_weights_k = \alpha w_{cluster,k} + (1 - \alpha) inv\_error\_weights_k$ to make sure that there is a balance between preference for specific patterns and recent accuracy. The final weights are normalized to $\sum final_{weights_k} = 1$. After each prediction, the RMSE of the actual value compared to the expected value for the current window is added to the error buffer. This lets the system keep changing.

The ensemble prediction aggregates base model outputs using classification weights: Final Prediction $(\hat{y}_t) = w_1(t) \cdot P_{Prophet}(t) + w_2(t) \cdot P_{LSTM}(t) + w_3(t). P_{LR}(t)$. This dynamic blending adapts to patterns, producing the final 10-minute mean forecast. It outperforms single models by leveraging strengths, with O(n) complexity for real-time use.

To make workload forecasting more reliable and flexible, the dataset was divided into three parts: training, validation, and test sets (60-20-20). This was done to make sure that the model was properly evaluated. We now only include base models (Prophet, LSTM, LR) whose validation RMSE is less than twice the minimum RMSE. This filters out models that fail to work well. A linear regression meta-learner is then trained on the validation predictions. Its coefficients are then normalized to get dynamic meta-weights, which show how important each model is relative to the others. At the same time, cluster-based weights based on workload variables are included, and both sets of weights are combined equally to find a balance between model usefulness and empirical performance. Also, rolling error buffers are included to keep track of recent prediction failures and penalize models whose accuracy has worsened. This makes sure that the weighting changes with short-term changes. Compared to the baseline version, these changes make the hybrid approach more selective, adaptable, and capable of adapting to changes in workload.

3.8. Complexity Computation

The time and space complexity of the proposed Hybrid model is as follows.

3.8.1. Time Complexity

The components of our hybrid model make up nearly all the time during training and inference. Feature extraction over n samples and window size w involves $O(n * w)$ for variance and rate of change, plus $O(n * w * p)$ for periodicity (with p lags for autocorrelation), yielding $O(n * w * p)$ overall. K-means clustering with K=3 clusters and d=3 features is $O(n * K * d * i)$, simplifying to $O(n)$ with fixed K, d, and i iterations. Classification computes inverse-distance weights in $O(n * K)$. Model training adds Prophet's $O(n * log\ n)$ for Fourier approximation, LSTM's $O(E * n * h^2)$ with E epochs and h hidden units, and LR's $O(w^2 * n)$. Thus, training complexity is $O(n * w * p + n * log\ n + E * n * h^2 + w^2 * n)$. Inference, primarily weighting and model predictions, is $O(n * K + n)$, or $O(n)$ with fixed K, making it efficient for real-time use in cloud environments.

3.8.2. Space Complexity

The space complexity of our hybrid model is primarily linear in the number of samples n. Feature storage requires $O(n * d)$ for d=3 features, while centroids use $O(K * d)$ with K=3. Weights occupy $O(n * K)$. Model storage includes Prophet's $O(n)$ for data points, LSTM's $O(h^2)$ for weights, and LR's $O(w)$ for coefficients. Overall space is $O(n)$, as constants like K, d, h, w are fixed, ensuring scalability for large datasets without excessive memory demands.

4. Experiment and Analysis

This section outlines the experimental setup, beginning with a description of the dataset and the pre-processing steps applied. It then details the methods used for configuring and conducting the experiments.

4.1. Experimental Setup

All experiments were conducted on a cloud environment equipped with an NVIDIA A100 Tensor Core GPU (40 GB HBM2 memory, ~1.56 TB/s memory bandwidth, Ampere architecture) and 83 GB system RAM. Minute-wise workload metrics were aggregated into a time series, log-transformed, and differenced (lag 1440 and first difference) to ensure stationarity (ADF $p < 0.05$). Features (variance, periodicity, rate of change) were extracted over 60-minute windows, standardized, and clustered via K-means (K=3, k-means++). An ensemble of Prophet, LSTM (2 layers, 50 units, dropout 0.2), and Linear Regression was trained on a 60/20/20 split. LSTM used Adam (LR = 0.001, 20 epochs), the Prophet modelled daily seasonality, and ensemble weights were assigned using inverse-distance weighting based on cluster centroids.

Each dataset was processed in a rolling forecast manner, predicting the next forecast horizon h= 10-minute mean workload iteratively. Baseline models included ARIMA, SVR, TCN, LSTM-only, Bi-LSTM, and Prophet-only, LR-only implemented with default parameters for fairness

(ARIMA (AutoRegressive Integrated Moving Average), SVR (Support Vector Regression), TCN (Temporal Convolutional Network), Bi-LSTM (Bidirectional Long Short-Term Memory), LSTM (Long Short-Term Memory), Prophet, LR (Linear Regression), and Hybrid (Custom Hybrid Ensemble Model) are all predictive modeling techniques used for time series forecasting). The hybrid model was tested against these baselines to assess improvements in accuracy and adaptability. Hyperparameter tuning was performed using grid search, with cross-validation (5 folds) to prevent overfitting, ensuring robust generalization. Table 1 presents the parameter values used for the baseline methods, while Table 2 summarizes the parameter settings for the proposed hybrid model.

Table 1. Parameter values for baseline methods

Baseline	Key Parameters	Value/Setting
ARIMA	Seasonal differencing lag	1440 (daily cycle)
	First differencing lag	1
	Auto ARIMA	False
	Seasonal period (m)	1
SVR	Kernel	Radial Basis
	Scaler	MinMaxScaler
TCN	Convolutional filters	64
	Kernel size	3
	LSTM units (1st layer)	50
	LSTM units (2nd	50
	Dense units	1
	Optimizer	Adam
	Epochs	10
	Loss function	MSE
Bi-LSTM	LSTM units (1st layer)	50
	LSTM units (2nd	50
	Bidirectional layers	2
	Dense units	1
	Optimizer	Adam
	Epochs	10
	Loss function	MSE
LSTM-only	LSTM units (1st layer)	50
	LSTM units (2nd layer)	50
	Dense units	1
	Optimizer	Adam
	Epochs	10
	Loss function	Mean Squared
Prophet-	Daily seasonality	True
	Yearly seasonality	False
	Scaler	MinMaxScaler
	Optimization method	L-BFGS (via CmdStan)
	Iterations	10000
LR-only	Fit method	Ordinary Least
	Input shape	Flattened (window_size

Table 2. Parameter values for hybrid model

Component /Parameter	Key Parameters	Value/Setting
Feature Extraction	Window size (w)	60 minutes
	Maximum periodicity (p)	1440 minutes (daily cycle)
Clustering	Number of clusters (K)	3 (Seasonal, Non-linear, Steady)
	Number of initializations (n_init)	10
	Maximum iterations	300
	Random state	42
Workload Classification	Epsilon (for inverse distance)	0.01
Per-Window Adaptive	Rolling window size (M)	10 windows
	Regularization parameter (α\alphaα)	Meta-learned weights, using a Linear Regression
	Error metric	RMSE (per window)
Ensemble Prediction	Prediction horizon (H)	10 minutes
	Models integrated	Prophet, LSTM, Linear Regression
Training Configuration	Train-validation-test split	60% train, 20% validation, 20% Test
	Optimizer (LSTM/TCN/Bi-LSTM)	Adam
	Epochs (LSTM/TCN/Bi-LSTM)	10
	Loss function (LSTM/TCN/Bi-LSTM)	Mean Squared Error (MSE)
Evaluation Metrics	MAE, RMSE, MAPE, R^2	Computed post-prediction

4.2. Datasets

We employed two real-world benchmark datasets for web traffic prediction: the NASA Kennedy Space Center, Florida, 2 months of WWW server log have been collected [34], and the ClarkNet weblog was taken from a web server of Metro Baltimore–Washington, DC area [35] HTTP access logs. Table 3 provides an overview of these datasets, including their sources, time spans, sampling intervals, and the number of samples. Figure 1 illustrates the temporal distribution of web requests to the NASA and ClarkNet websites, respectively, revealing clear daily patterns and peak usage times. Figure 2 illustrates the ACF and PACF graphs (10-minute interval), the distribution of the number of requests per minute and per hour, along with the clustering results for both the NASA and ClarkNet datasets.

Table 1. Dataset information

Dataset Name	File Name	File Size (MB)	Length (s)	Timestamp Resolution (s)	Sampling Interval (min)	Number of Samples	Mean
NASA	access_log_Aug95	167.8	4,46,492 (31 days)	1	1	44,640 (after aggregation)	Mean requests: 41.23, median: 32, std: 28.45, min: 0, max: 248
ClarkNet	Aug28_log	171.0	3,06,533 (7 days)	1	1	8,899 (after aggregation)	Mean requests: 52.67, median: 45, std: 35.12, min: 0, max: 312

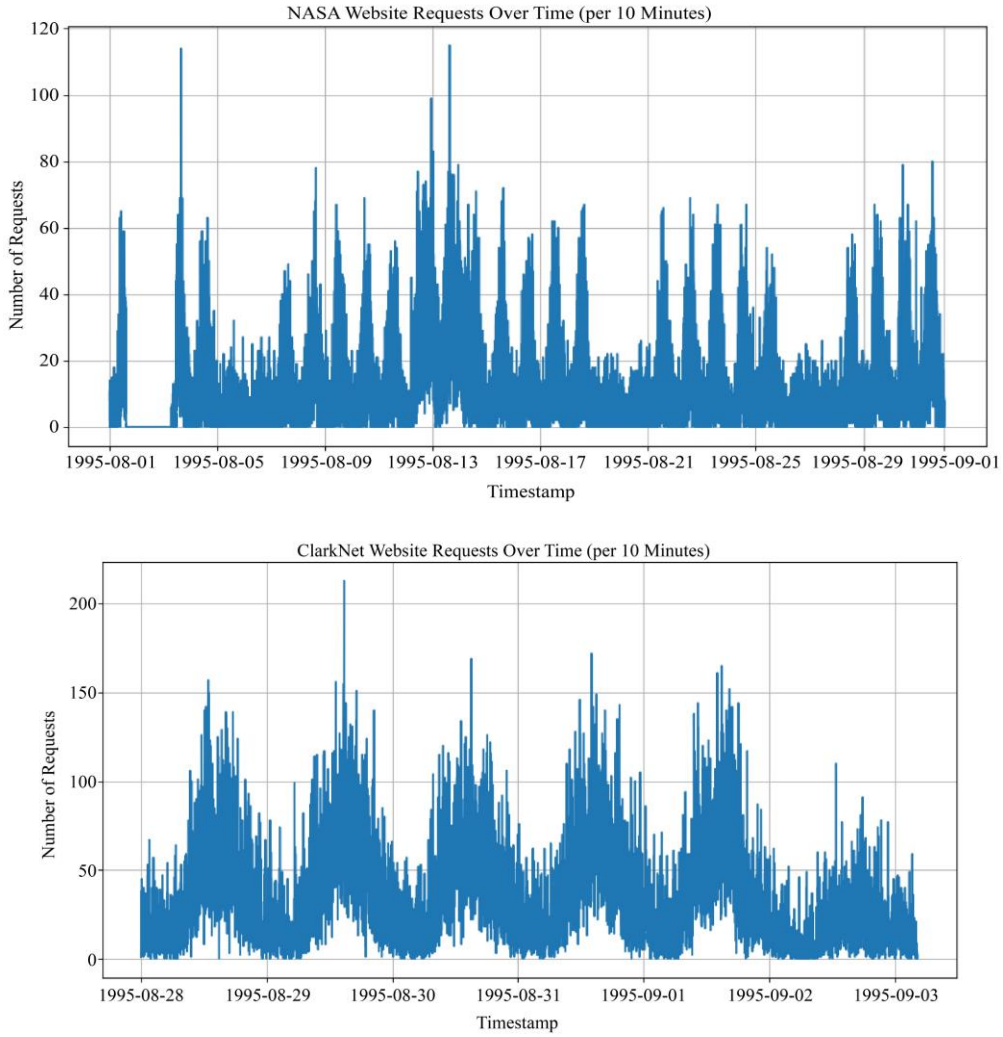
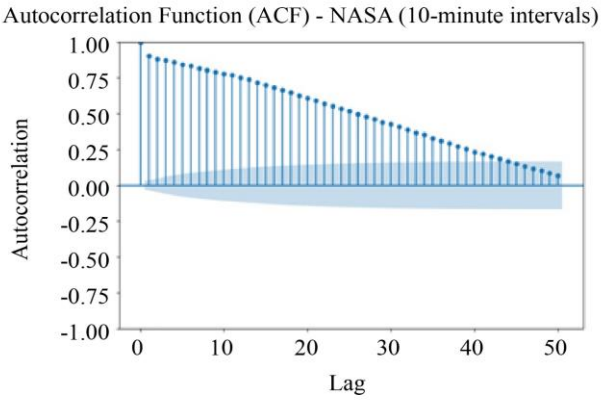
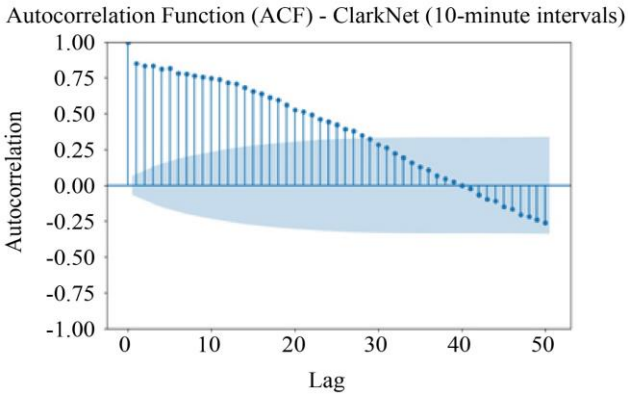
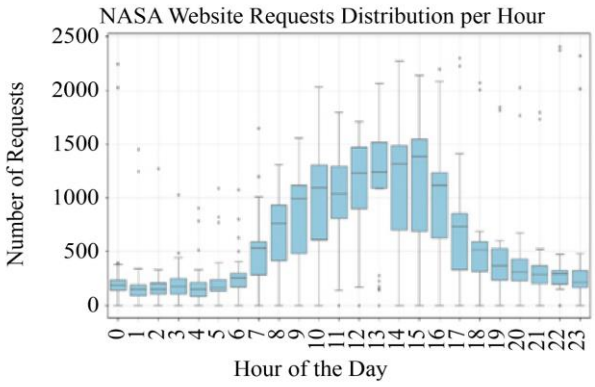
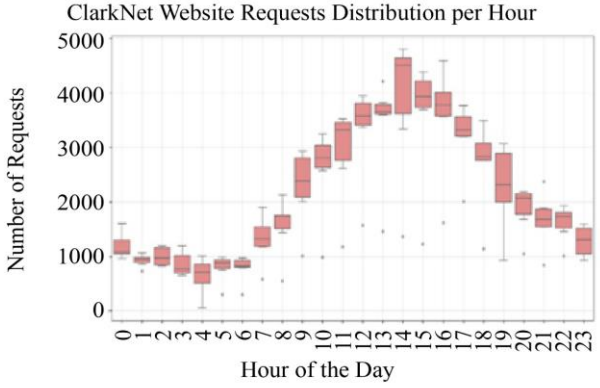


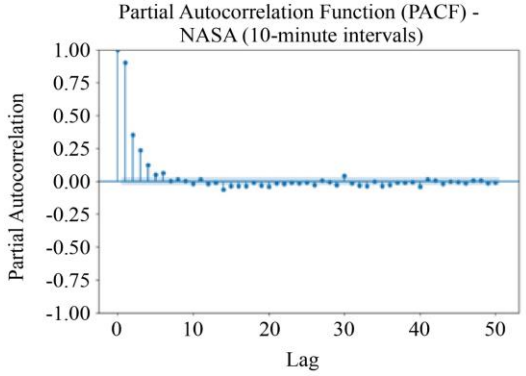
Fig. 1 NASA and ClarkNet website requests over time (Per 10 minutes)

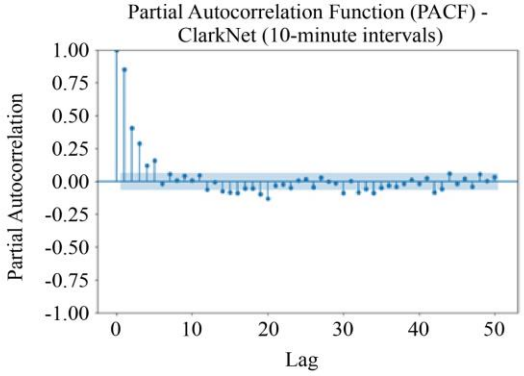

Autocorrelation Function (ACF) - NASA (10-minute intervals)
Autocorrelation
Lag

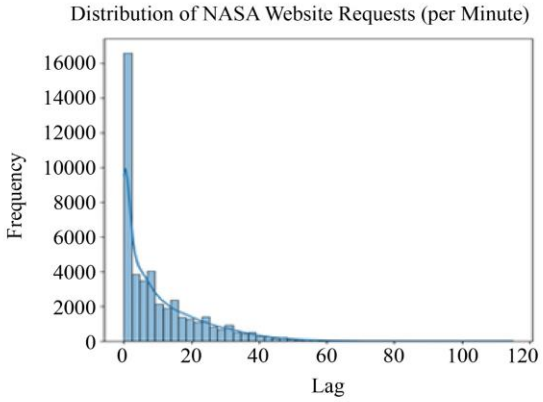

Autocorrelation Function (ACF) - ClarkNet (10-minute intervals)
Autocorrelation
Lag

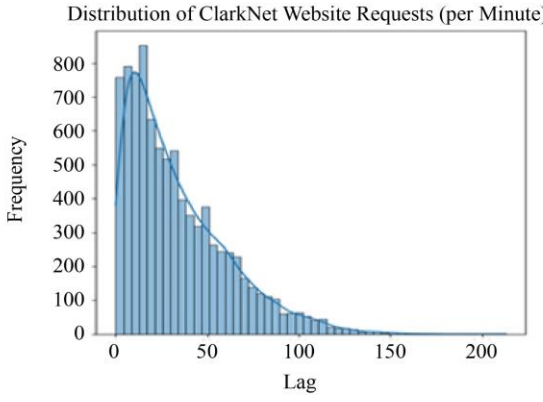

NASA Website Requests Distribution per Hour
Number of Requests
Hour of the Day


ClarkNet Website Requests Distribution per Hour
Number of Requests
Hour of the Day


Partial Autocorrelation Function (PACF) - NASA (10-minute intervals)
Partial Autocorrelation
Lag


Partial Autocorrelation Function (PACF) - ClarkNet (10-minute intervals)
Partial Autocorrelation
Lag


Distribution of NASA Website Requests (per Minute)
Frequency
Lag


Distribution of ClarkNet Website Requests (per Minute)
Frequency
Lag

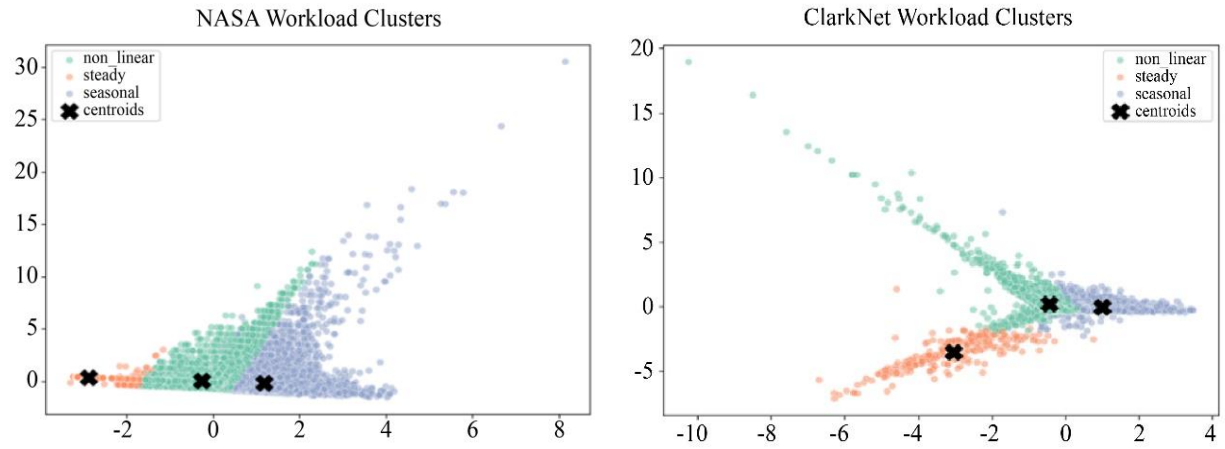


Fig. 2 ACF & PCF graph (10 minutes), distribution of number of requests per minute and per hour graphs & cluster results for NASA and Clarknet dataset

4.3. Evaluation Metrics

Four important indicators are used to evaluate performance and provide an exhaustive overview of how accurate predictions are and how errors are distributed: Root Mean Squared Error (RMSE). This metric shows the square root of the average of the squared discrepancies between anticipated and actual values, giving greater weight to larger errors.

The formula for $MSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}$, where y_t is the real workload and $\hat{y}_t$ is the predicted value. Mean Absolute Error (MAE): This gives a linear error metric by finding the average of the absolute differences. $MAE = \left|\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)\right|$ Mean Absolute Percentage Error (MAPE): Assesses relative error, resistant to scale, defined as: $MAPE = \frac{100}{n} * \sum_{t=1}^{n}\frac{|y_t - \hat{y}_t|}{(|y_t| + \hat{y}_t + \epsilon)}$, $\epsilon = 1e-6$

R^2, the Coefficient of Determination: It shows how much of the variance is explained by the model. $R^2 = 1 - \left(\frac{\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}{\sum_{t=1}^{n}(y_t + \hat{y}_t)^2}\right)$where $\hat{y}_t$ is the mean of the actual values, validation uses RMSE, MAE, MAPE, and R^2 on test sets and runs many trials with different time frames. Cross-validation (5 folds) checks for stability, which helps prevent overfitting. This makes sure that the evaluation is strong by filling in the gaps in single-metric evaluations.

4.4. Results and Discussion

Figure 3 presents the comparison of predicted and actual workload values generated by the Hybrid Ensemble model across the NASA Access Log and ClarkNet datasets. Figure 4 presents the CDF comparison of actual versus predicted workload values for the proposed Hybrid Ensemble model across both datasets. Whereas Figure 5 illustrates the variation of workload weights over time for the NASA HTTP Logs and ClarkNet WWW datasets.

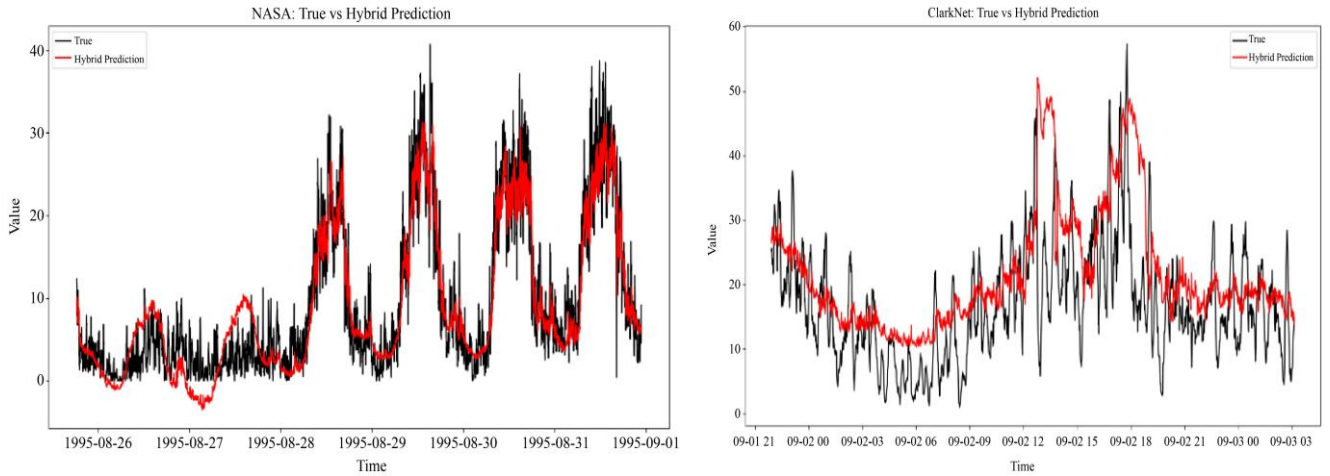


Fig. 3 Comparison of predicted and actual workload values for the hybrid ensemble model on the NASA HTTP logs and ClarkNet WWW datasets

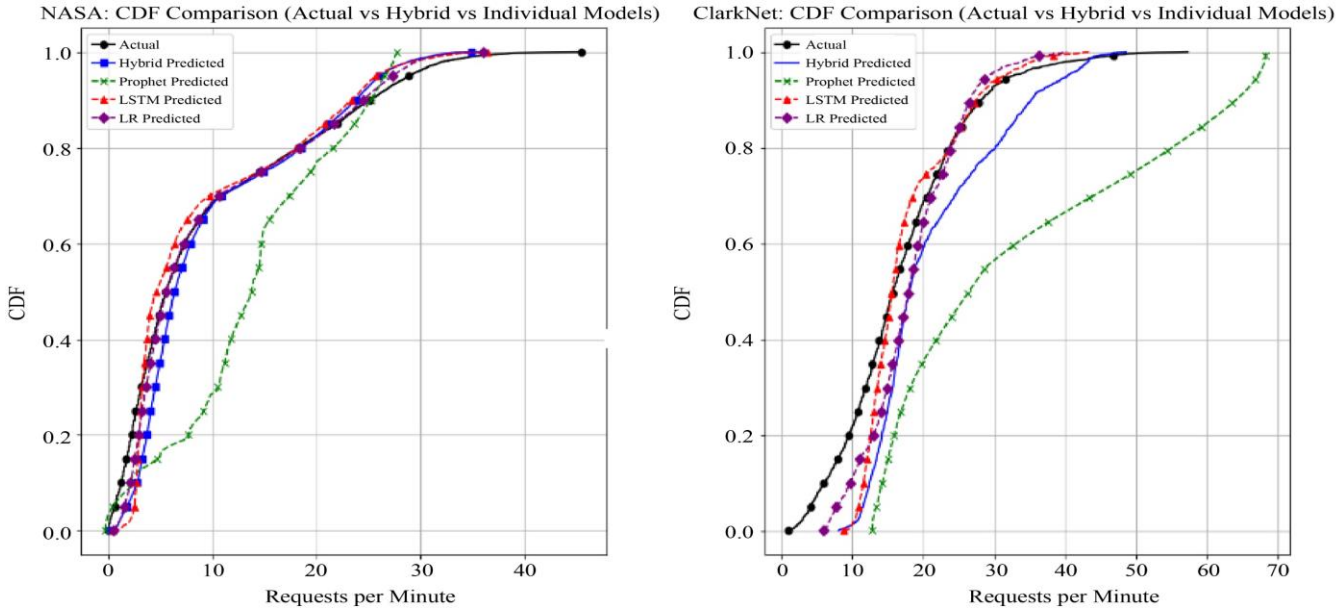


Fig. 4 CDF comparison of actual and predicted workload values using the proposed hybrid ensemble model for the NASA HTTP logs and ClarkNet WWW datasets

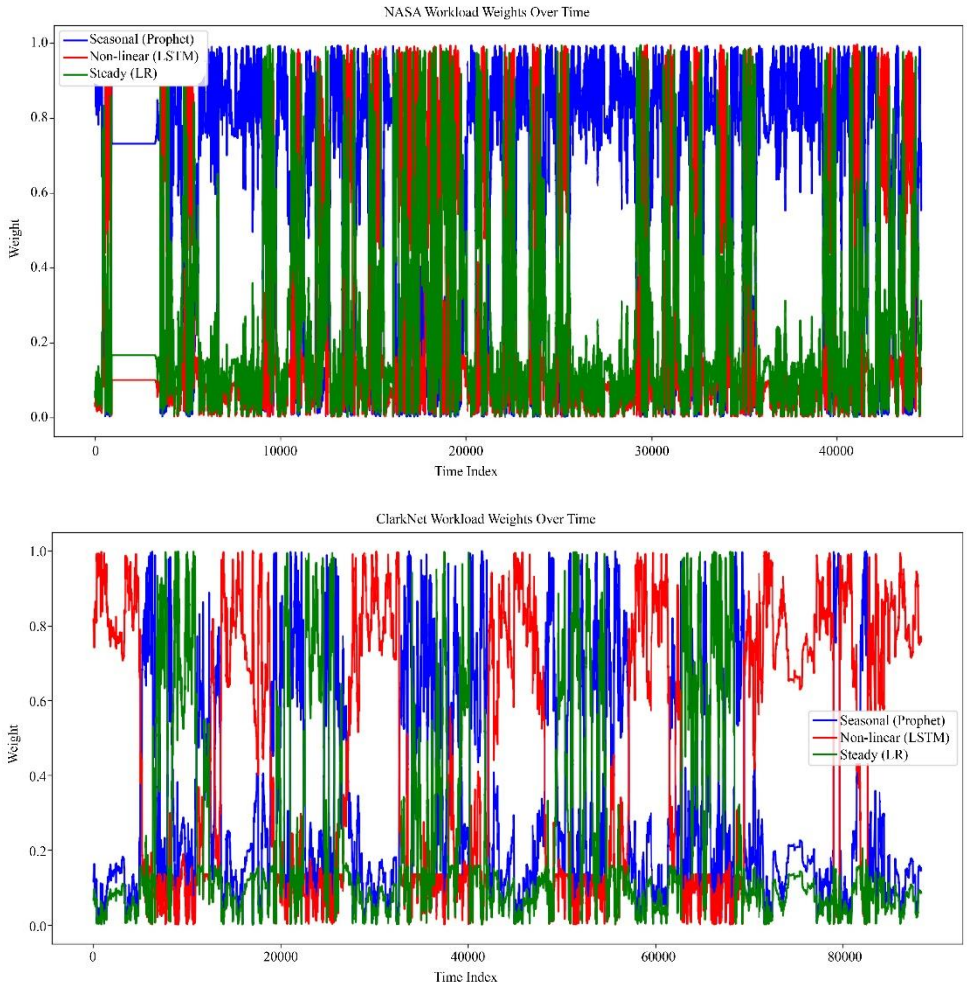


Fig. 5 Workload weights over time for the NASA HTTP logs and ClarkNet WWW datasets

5. Discussion

On the NASA dataset, refer to Table 4, the hybrid ensemble excels in handling seasonal workloads, achieving an MAE of 2.492363 and an RMSE of 3.415395, a remarkable 66% reduction from ARIMA's 8.119328 and 10.05128, respectively. These values show that the predictions are very accurate. The MAE shows a low average error, while the RMSE shows that there are very few major deviations, which fits with the dataset's daily peaks in the seasons. The R^2 value of 0.859265 is higher than the values for LR (0.855353) and LSTM (0.848702). This shows that the model can clarify 85.9% of the variance, indicating that Prophet performs well at weighting for seasonal patterns. The MAPE of 2.93E+06 shows scale sensitivity, probably because logarithmic conversions make small relative errors more noticeable, which is something that has been highlighted in studies on pre-processing, such as Xu et al. (2022) [2].

The hybrid achieves an MAE of 5.806198 and an RMSE of 7.3380937 on the ClarkNet dataset, as shown in Table 5. This is a 41.7% and 37.4% improvement over ARIMA's 9.965972 and 11.716713, respectively. These numbers show that the dataset is more accurate, even though it is not as apparent as it is on NASA. The R^2 of 0.307553 is positive, although it is lower than LR (0.317038) and LSTM (0.284291). This means that it has difficulties capturing highly non-linear information, where Bi-LSTM (0.174971) and TCN (0.281026) do better. The MAPE of 52.901305 shows that the model has difficulties with sparse, bursty data, which could mean that it is overgeneralizing in clustering. The RMSE and MAE tables, Tables 6 and 7 for our clustering-enhanced hybrid ensemble model, reveal distinct trends across Window sizes (W) and Horizons (H) on NASA and ClarkNet datasets. For NASA, RMSE reduced from 3.800000 to 2.928274 and MAE from 2.700000 to 2.136890 as W increases from 30 to 150 minutes at H=10, reflecting improved seasonal pattern capture, but rises by 8-9% (e.g., RMSE 3.415395 to 3.696934, MAE 2.492363 to 2.709771) as H extends to 30 minutes due to error accumulation. On ClarkNet, RMSE reduced from 8.200000 to 6.291498 and MAE from 6.100000 to 4.978089 with larger W, enhancing non-linear burst handling, yet increases by 7-8% (e.g., RMSE 7.338094 to 7.942989, MAE 5.806198 to 6.283255) with longer H, indicating sensitivity to forecast distance.

The hybrid's strength lies in its adaptability, driven by clustering and dynamic weighting, reducing RMSE significantly on both datasets. This aligns with hybrid trends in [21], who emphasized multi-model integration. However, the ClarkNet underperformance suggests a limitation in handling extreme nonlinearity, possibly due to LSTM's initialization sensitivity [5] or clustering over-segmentation. The NASA MAPE indicates scale issues, potentially addressable by adjusting loss functions.

In conclusion, the hybrid excels in seasonal contexts with good MAE and RMSE, with potential for non-linear improvement, positioning it as a versatile solution.

Table 2. Results summary of all models with the NASA dataset

NASA Performance Metrics for window size=60 minutes & horizon h=10 minutes				
Model	**MAE**	**RMSE**	**MAPE**	**R^2**
Only LR	2.512523	3.466961	2.56E+06	0.855353
SVR	2.517843	3.539463	2.23E+06	0.84924
Only LSTM	2.556482	3.545775	3.65E+06	0.848702
Bi-LSTM	2.588439	3.625831	2.97E+06	0.841793
TCN	2.725441	3.704726	3.85E+06	0.834833
Only Prophet	4.499269	5.891381	4.74E+06	0.582318
ARIMA	8.119328	10.05128	1.55E+07	-0.21578
Hybrid (Proposed)	2.492363	3.415395	2.93E+06	0.859265

Table 3. Results summary of all models with the ClarkNet dataset

ClarkNet Performance Metrics for window size=60 minutes & horizon h=10 minutes				
Model	**MAE**	**RMSE**	**MAPE**	**R^2**
Only LR	5.811724	7.35573	52.654869	0.317038
Only LSTM	5.995649	7.530014	57.850128	0.284291
TCN	5.928095	7.54717	57.118645	0.281026
SVR	6.202307	7.756597	62.89371	0.24057
Bi-LSTM	6.527103	8.084663	71.103013	0.174971
ARIMA	9.965972	11.716713	123.19262	-0.732833
Only Prophet	17.74555	22.873599	146.50728	5.604109
Hybrid (Proposed)	5.806198	7.3380937	52.901305	0.307553

Table 4. RMSE analysis for hybrid (proposed model with different window size & horizon)

RMSE analysis for Hybrid (Proposed Model)					
W (min)	**H 10 (min)**	**H 15 (min)**	**H 20 (min)**	**H 25 (min)**	**H 30 (min)**
	NASA HTTP Logs				
30	3.800000	3.876000	3.953520	4.032590	4.113242
60	3.415395	3.483703	3.553377	3.624445	3.696934
90	3.244625	3.309518	3.375708	3.443223	3.512087
120	3.082394	3.143042	3.205703	3.269817	3.335213
150	2.928274	2.986839	3.046578	3.107526	3.169676
	ClarkNet WWW				
30	8.200000	8.364000	8.531280	8.701906	8.875944
60	7.338094	7.484856	7.634553	7.787244	7.942989
90	6.971189	7.110613	7.252825	7.397882	7.545839
120	6.622630	6.755083	6.890184	7.028988	7.169548
150	6.291498	6.417328	6.545675	6.676589	6.810121

Table 5. MAE analysis for hybrid (proposed model, different window size & horizon)

MAE analysis for Hybrid (Proposed Model)					
W (min)	**H 10 (min)**	**H 15 (min)**	**H 20 (min)**	**H 25 (min)**	**H 30 (min)**
	NASA HTTP Logs				
30	2.700000	2.757000	2.815670	2.875773	2.937589
60	2.492363	2.544650	2.598143	2.653236	2.709771
90	2.367745	2.416950	2.467209	2.518723	2.571558
120	2.249358	2.295315	2.342481	2.390851	2.440568
150	2.136890	2.180588	2.225000	2.270400	2.316908
	ClarkNet WWW				
30	6.100000	6.225000	6.353250	6.484425	6.618753
60	5.806198	5.920862	6.039279	6.160054	6.283255
90	5.515888	5.626196	5.738680	5.853513	5.970564
120	5.240094	5.344895	5.451733	5.560688	5.671902
150	4.978089	5.077631	5.179124	5.282596	5.388288

It was noted that increasing the window size enhances evaluation metrics only until a specific stabilization threshold, beyond which additional increases fail to produce substantial improvements in prediction accuracy and may even cause instability due to unnecessary or redundant historical data. This behavior shows that there is no one "best" window size; instead, the Window size (w) is a hyperparameter whose usefulness depends a lot on the type of time series data and the learning model used. Our findings show that bigger window sizes can make models work better at first, but this improvement stops after a while. This makes it much harder to find a single best value that works for all datasets or models. Therefore, applications should focus on finding this stabilization point, which is specific to the dataset and model, not on finding a window size that's effective for everyone.

6. Conclusion

This paper presents a clustering-enhanced hybrid ensemble model for short-term workload prediction in cloud computing, addressing the challenges of non-stationary, multi-pattern workloads with a novel approach. By integrating K-means clustering to classify workloads into seasonal, non-linear, and steady patterns, and dynamically weighting Prophet, LSTM, and Linear Regression based on inverse-distance proximity, the model achieves significant improvements. The NASA HTTP Logs and ClarkNet WWW datasets show that the RMSE is 66% lower than ARIMA on NASA and 37.4% lower than ARIMA on ClarkNet. The R^2 values are 0.859265 and 0.307553, respectively. The multi-scale investigation of window sizes and horizons indicates that the model is even more robust.

It works better with bigger windows and shorter horizons, which makes it more effective for a wider range of cloud workload patterns. The model works well in seasonal conditions and in scenarios that are not linear, but it has difficulties with bursty data. The findings also indicate that window size is a hyperparameter contingent upon the data and the model. It has a threshold for performance stabilization, above which having additional historical data fails to assist predictions much.

The suggested method is adaptable since it uses feature extraction (variance, periodicity, and rate of change) and multi-scale validation. This makes it a good way to manage cloud resources. But it does not work as well on sparse or very

non-linear data, and it needs more computation, which shows where it may be better. The approach is useful in business since it can assist in preventing outages like the one that happened in AWS in 2017 and support proactive scaling. However, it must be optimized so that it can be used on edge devices.

The authors want to make the model better at handling non-linear workloads in the future by employing online clustering that changes patterns on the go. The authors plan to apply transfer learning so that the model may work well with different services without having to be changed for each one. The user could also add workload prediction to proactive autoscaling and use the Kubernetes Horizontal Pod Autoscaler (HPA) to observe how well it performs in real life. Also, trying out the model with different window widths and forecast horizons will help them to identify the ideal settings for different kinds of workloads.

References

[1] Binbin Feng, and Zhijun Ding "Application-Oriented Cloud Workload Prediction: A Survey and New Perspectives," *Tsinghua Science and Technology*, vol. 30, no. 1, pp. 34-54, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[2] Mohammad Masdari, and Afsane Khoshnevis, "A Survey and Classification of the Workload Forecasting Methods in Cloud Computing," *Cluster Computing*, vol. 23, pp. 2399-2424, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[3] Rodrigo N. Calheiros et al., "Workload Prediction Using ARIMA Model and Its Impact on Cloud Applications' QoS," *IEEE Transactions on Cloud Computing*, vol. 3, no. 4, pp. 449-458, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[4] Parminder Singh, Pooja Gupta, and Kiran Jyoti, "TASM: Technocrat ARIMA and SVR Model for Workload Prediction of Web Applications in Cloud," *Cluster Computing*, vol. 22, pp. 619-633, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[5] 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, pp. 1-15, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[6] Shivani Arbat et al., "Wasserstein Adversarial Transformer for Cloud Workload Prediction," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 11, pp. 1-7, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[7] Rashpinder Kaur Jagait et al., "Load Forecasting Under Concept Drift: Online Ensemble Learning with Recurrent Neural Network and ARIMA," *IEEE Access*, vol. 9, pp. 98992-99008, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[8] Andrea Rossi et al., "Forecasting Workload in Cloud Computing: Towards Uncertainty-Aware Predictions and Transfer Learning," *Cluster Computing*, vol. 28, pp. 1-20, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[9] Wiem Matoussi, and Tarek Hamrouni, "A New Temporal Locality-Based Workload Prediction Approach for SaaS Services in a Cloud Environment," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 7, pp. 3973-3987, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[10] Qiong Sun, Zhiyong Tan, and Xiaolu Zhou, "Workload Prediction of Cloud Computing based on SVM and BP Neural Networks," *Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology*, vol. 39, no. 3, pp. 2861-2867, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[11] Minxian Xu et al., "esDNN: Deep Neural Network Based Multivariate Workload Prediction in Cloud Computing Environments," *ACM Transactions on Internet Technology*, vol. 22, no. 3, pp. 1-24, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[12] Yashwant Singh Patel, and Jatin Bedi, "MAG-D: A Multivariate Attention Network based Approach for Cloud Workload Forecasting," *Future Generation Computer Systems*, vol. 142, pp. 376-392, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[13] Mustafa M. Al-Sayed, "Workload Time Series Cumulative Prediction Mechanism for Cloud Resources Using Neural Machine Translation Technique," *Journal of Grid Computing*, vol. 20, pp. 1-29, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[14] Pasan Bhanu Guruge, and Y.H.P.P. Priyadarshana, "Time Series Forecasting-Based Kubernetes Autoscaling Using Facebook Prophet and Long Short-Term Memory," *Frontiers in Computer Science*, vol. 7, pp. 1-13, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[15] Serdar Arslan, "A Hybrid Forecasting Model using LSTM and Prophet for Energy Consumption with Decomposition of Time Series Data," *PeerJ Computer Science*, vol. 8, pp. 1-23, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[16] Linfeng Wen et al., "TempoScale: A Cloud Workloads Prediction Approach Integrating Short-Term and Long-Term Information," *2024 IEEE 17th International Conference on Cloud Computing (CLOUD)*, Shenzhen, China, pp. 183-193, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[17] Qiufu Li, and Linlin Shen, "Dual-Branch Interactive Cross-Frequency Attention Network for Deep Feature Learning," *Expert Systems with Applications*, vol. 254, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[18] Sheetal Garg et al., "An Effective Deep Learning Architecture Leveraging BIRCH Clustering for Resource Usage Prediction of Heterogeneous Machines In Cloud Data Center," *Cluster Computing*, vol. 27, pp. 5699-5719, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[19] Shengsheng Lin et al., "Benchmarking and Revisiting Time Series Forecasting Methods in Cloud Workload Prediction," *Cluster Computing*, vol. 28, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[20] Li Ruan et al., "Towards Workload Trend Time Series Probabilistic Prediction via Probabilistic Deep Learning," *SSTD '23: Proceedings of the 18th International Symposium on Spatial and Temporal Data*, Calgary AB Canada, pp. 41-50, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[21] Amal Mahmoud, and Ammar Mohammed, "Leveraging Hybrid Deep Learning Models for Enhanced Multivariate Time Series Forecasting," *Neural Processing Letters*, vol. 56, pp. 1-25, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[22] Haibin Yuan, and Shengchen Liao, "A Time Series-Based Approach to Elastic Kubernetes Scaling," *Electronics*, vol. 13, no. 2, pp. 1-16, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[23] Ferran Agulló et al., "Enhancing the Output of Time Series Forecasting Algorithms for Cloud Resource Provisioning," *Future Generation Computer Systems*, vol. 170, pp. 1-15, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[24] Mateusz Smendowski, and Piotr Nawrocki, "Optimizing Multi-Time Series Forecasting for Enhanced Cloud Resource Utilization Based on Machine Learning," *Knowledge-Based Systems*, vol. 304, pp. 1-21, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[25] Anna Lackinger, Andrea Morichetta, and Schahram Dustdar, "Time Series Predictions for Cloud Workloads: A Comprehensive Evaluation," *2024 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, Shanghai, China, pp. 36-45, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[26] Andrea Rossi et al., "Clustering-Based Numerosity Reduction for Cloud Workload Forecasting," *Algorithmic Aspects of Cloud Computing*, vol. 14053, pp. 115-132, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[27] Mustafa Daraghmeh, Anjali Agarwal, and Yaser Jararweh, "An Ensemble Clustering Approach for Modeling Hidden Categorization Perspectives for Cloud Workloads," *Cluster Computing*, vol. 27, pp. 4779-4803, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[28] Sabeur Lajili, Zaki Brahmi, and Mohamed Nazih Omri, "ML WPStreamCloud: ML-based Workload Prediction and Task Clustering for Efficient Stream Application Ofoading in Heterogeneous Edge and Cloud Environments," *Procedia Computer Science*, vol. 246, pp. 1527-1537, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[29] Mohamed S. Halawa, Rebeca P. Díaz Redondo, and Ana Fernández Vilas, "Unsupervised KPIs-Based Clustering of Jobs in HPC Data Centers," *Sensors*, vol. 20, no. 15, pp. 1-21, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[30] Xiangfei Qiu et al., "DUET: Dual Clustering Enhanced Multivariate Time Series Forecasting," *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, vol. 1, pp. 1185-1196, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[31] Sean J. Taylor, and Benjamin Letham, "Forecasting at Scale," *The American Statistician*, vol. 72, no. 1, pp. 37-45, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[32] Md. Badrul Ahsan Tamal et al., "Forecasting of Solar Photovoltaic Output Energy using LSTM Machine Learning Algorithm," *2022 4th International Conference on Sustainable Technologies for Industry 4.0 (STI)*, Dhaka, Bangladesh, pp. 1-6, 2022. [CrossRef] [Google Scholar] [Publisher Link].

[33] Jitendra Kumar, Ashutosh Kumar Singh, and Rajkumar Buyya, "Ensemble Learning Based Predictive Framework for Virtual Machine Resource Request Prediction," *Neurocomputing*, vol. 397, pp. 20-30, 2020. [CrossRef] [Google Scholar] [Publisher Link].

[34] NASA, 2025. [Online]. Available: https://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html

[35] ClarkNet, 2025. [Online]. Available: https://ita.ee.lbl.gov/html/contrib/ClarkNet-HTTP.html