BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

SpectraNet: Hybrid Time and Frequency-Domain Modeling for Sustainable Cloud CPU Prediction

Nahin F. Siddiqui
200041143
Zarif Safwan Hoque
200041147
Md. Ehsanul Haque
200041113

Department of Computer Science and Engineering
Islamic University of Technology
September, 2025

SpectraNet: Hybrid Time and Frequency-Domain Modeling for Sustainable Cloud CPU Prediction

Nahin F. Siddiqui
200041143
Zarif Safwan Hoque
200041147
Md. Ehsanul Haque
200041113

Department of Computer Science and Engineering
Islamic University of Technology
September, 2025

Declaration of Candidate

This is to certify that the work presented in this thesis is the outcome of the analysis and experiments carried out by **Nahin F. Siddiqui**, **Zarif Safwan Hoque**, and **Md. Ehsanul Haque** under the supervision of **Dr. Md. Azam Hossain**, Associate Professor, Department of Computer Science and Engineering and co-supervision of **Aashnan Rahman**, Junior Lecturer, Department of Computer Science and Engineering, Islamic University of Technology, Dhaka, Bangladesh. It is also declared that neither this thesis nor any part of it has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others have been acknowledged in the text and a list of references is given.

Dr. Md. Azam Hossain
Associate Professor
Department of Computer Science and Engineering
Islamic University of Technology (IUT)
Date: September 07, 2025

Aashnan Rahman
Junior Lecturer
Department of Computer Science and Engineering
Islamic University of Technology (IUT)
Date: September 07, 2025

Nahin F. Siddiqui
Student ID: 200041143
Date: September 07, 2025

Zarif Safwan Hoque
Student ID: 200041147
Date: September 07, 2025

Md. Ehsanul Haque
Student ID: 200041113
Date: September 07, 2025

Contents

List of Figures

List of Tables

Abstract

The proliferation of cloud computing has created an urgent need for sustainable resource management strategies to mitigate the growing energy consumption and carbon footprint of data centers. Accurate CPU workload forecasting is a cornerstone of this effort, enabling proactive resource allocation that minimizes energy waste from over-provisioning, and prevents performance degradation from under-provisioning. This study introduces **SpectraNet**, a lightweight hybrid model that advances sustainable cloud computing by delivering highly accurate CPU usage predictions with minimal computational overhead. By integrating time domain and frequency-domain analysis, **SpectraNet** effectively captures both transient and periodic workload patterns. Experimental results on the Azure VM CPU Readings and Alibaba Cloud Workload datasets demonstrate that this dual-domain approach significantly improves forecasting accuracy. **SpectraNet** demonstrates an competitive balance of performance and efficiency, achieving a Mean Absolute Error (MAE) of 0.0549 on long-range forecasts. Critically, the model achieves this accuracy while being up to **19 times smaller** (with only 89,805 parameters) and **5 times faster** at inference than larger, more complex architectures. The model's efficiency and small footprint make it a practical and scalable solution for real world, resource-constrained cloud environments. By enabling more precise and energy-efficient resource management, **SpectraNet** contributes a valuable tool for building more sustainable and cost-effective cloud infrastructures.

Keywords: Cloud Computing, Sustainable Resource Management, CPU Workload Forecasting, Time-Frequency Analysis, Hybrid Deep Learning Models, Lightweight AI Models.

Chapter 1

Introduction

1.1 Introductory Information

Cloud computing forms the backbone of modern digital infrastructure, but its rapid expansion has led to a significant increase in the energy consumption and carbon footprint of global data centers [129]. Consequently, **sustainable cloud management** has emerged as a critical imperative, driving research towards strategies that enhance operational efficiency while minimizing environmental impact [34], [82], [163]. A cornerstone of this effort is the accurate forecasting of resource usage, particularly CPU demand. Recent studies have demonstrated that leveraging predictive models for CPU scaling can significantly reduce energy consumption [14], [134]. Reliable forecasts enable dynamic resource allocation, preventing both **energy-wasting over-provisioning** and performance-degrading under-provisioning [22], [89]. This not only ensures compliance with Service Level Agreements (SLAs) but also directly contributes to more stable, cost-effective, and energy-efficient cloud ecosystems, a goal actively pursued by the latest generation of carbon-aware schedulers [133], [157]. However, the highly dynamic and non-linear nature of modern cloud workloads presents a formidable challenge to achieving such forecasts [119].

Traditional statistical methods like Autoregressive Integrated Moving Average (ARIMA)[113], while foundational, are ill-suited for these complex environments. Their reliance on linearity assumptions, sensitivity to noise, and inability to scale make them inadequate for today's high-dimensional cloud data [4], [130]. While deep learning models such as LSTMs and CNNs have advanced the field by capturing non-linear patterns, they often operate in a single domain, limiting their effectiveness [25]. Time-domain models frequently miss underlying periodicities, whereas frequency-domain models can overlook sudden, localized demand spikes. Hybrid architectures that integrate both domains are

essential for a holistic understanding but require careful design to avoid introducing noise and redundancy [167], [170]. A significant, unaddressed challenge remains: many advanced models, despite their accuracy, are too computationally demanding for practical, real-time deployment, thereby undermining the very goal of resource efficiency.

This research directly confronts this challenge by developing **SpectraNet**, a novel and lightweight hybrid model designed for **sustainable CPU usage prediction**[1], [127]. The scope of this work is to introduce a computationally efficient architecture that strategically integrates time- and frequency-domain information to maximize predictive accuracy while minimizing resource overhead. Our model is engineered to be practical for resource-constrained cloud environments, making it a viable tool for real-world energy conservation[32]. The key contributions of this work are threefold:

1. **A Novel Hybrid Time-Frequency Architecture:** We propose a dual-branch model that fuses spectral features (via Fourier Transform) with raw temporal data. This comprehensive representation captures both long-term periodic trends and short-term, localized fluctuations in cloud workloads.

2. **Adaptive Attention for Feature Refinement:** Our design incorporates a hierarchical Squeeze-and-Excitation Network (SENet) to dynamically weigh and select the most salient features from the fused data, enhancing signal clarity while suppressing noise and redundancy [63].

3. **A Focus on Lightweight, Sustainable Deployment:** We introduce an efficient framework that significantly reduces the computational overhead associated with complex deep learning models. By prioritizing a smaller footprint, SpectraNet offers a practical solution that improves resource management, reduces operational costs, and contributes to a more **sustainable and cost-effective cloud infrastructure**.

Existing research on cloud workload forecasting has explored various hybrid models combining time and frequency domain information. However, most existing methods transform data to the frequency domain only as an intermediate feature extraction step before returning to the time domain for final forecasting. In contrast, our approach directly integrates both domains in a parallel manner, enabling SpectraNet to jointly exploit temporal and spectral dependencies without reverting to a single-domain representation. This design redefines multidomain time series analysis by simplifying the modeling process while capturing richer long-term dependencies critical for cloud workload prediction.

The remainder of this paper is organized as follows: Section 2 reviews related works in cloud workload forecasting. Section 3 details the proposed SpectraNet methodology. Section 4 presents the experimental setup and results.Finally, Section 5 concludes the

paper by summarizing our findings and discussing their implications for sustainable cloud computing.

1.1.1 Motivations and Scope

The motivation for this research stems from the critical need to balance performance and cost in cloud computing. The dynamic and often unpredictable nature of cloud workloads makes it challenging to allocate resources efficiently. Over-provisioning leads to unnecessary expenses, while under-provisioning results in poor user experiences and potential SLA violations. This research is driven by the potential societal impact of enabling more efficient and cost-effective cloud services. Previous research has explored various methods for cloud workload prediction, including traditional time series forecasting and more advanced machine learning techniques. However, many existing solutions are computationally demanding, hindering their deployment in resource-constrained environments. This research aims to address the limitations of current approaches by developing a lightweight model that can accurately predict CPU usage.[129] The scope of this work includes investigating the benefits of incorporating frequency domain information alongside traditional time domain analysis to improve prediction accuracy while minimizing computational overhead.

1.1.2 Problem Statement

The core problem this research tackles is the need for a simple, lightweight time series forecasting model for CPU usage prediction in cloud computing platforms that effectively utilizes both time domain and frequency domain information to balance accuracy and computational efficiency. The specific research objectives are:

- To investigate and analyze existing time series forecasting models, including their strengths and limitations in the context of cloud CPU usage prediction. To explore the benefits of integrating frequency domain analysis with time domain methods for enhancing the accuracy and efficiency of CPU usage prediction.
- To develop a lightweight hybrid model that leverages both time and frequency domain information for accurate and efficient CPU usage forecasting in cloud environments.
- To conceptually evaluate the potential performance and efficiency of the proposed model in comparison to existing state-of-the-art methods.

Given a multivariate time series

$$X = x_t \in \mathbb{R}^d \mid t = 1, 2, \ldots, T, \tag{1.1}$$

where each vector

$$x_t = [\text{cpu\_util}_t, \text{mem\_util}_t, \text{net\_in}_t, \text{disk\_io}_t, \ldots] \tag{1.2}$$

represents resource usage metrics of cloud functions at time t, design a lightweight hybrid model

$$f_\theta : \mathbb{R}^{T \times d} \to \mathbb{R} \tag{1.3}$$

that minimizes the forecasting error

$$\mathcal{L}(y, \hat{y}) = \frac{1}{H} \sum_{h=1}^{H} (y_{t+h} - \hat{y}_{t+h})^2 [61] \tag{1.4}$$

where y_{t+h} is the true CPU usage and $\hat{y}_{t+h}$ is the predicted value. Our core objectives are to: (i) leverage both frequency-domain periodicity and time-domain locality; (ii) enhance feature relevance through adaptive attention; (iii) promote **sustainability** by ensuring a small model footprint; and (iv) maintain high efficiency for real-time forecasting.

1.1.3 Research Challenges

This research faces several challenges inherent to the field of cloud workload prediction. One significant challenge is the complex and highly variable nature of cloud workloads, which often exhibit non-linear patterns and are influenced by numerous external factors. Developing a model that can accurately capture these intricate dynamics while remaining lightweight is a key difficulty. Another challenge lies in balancing the trade-off between prediction accuracy and computational efficiency. Many advanced deep learning models achieve high accuracy but require significant computational resources, making them unsuitable for resource-constrained cloud environments. This research aims to confront this challenge by exploring simpler model architectures and efficient feature extraction techniques. Furthermore, the non-stationary nature of cloud workloads, with shifting trends and seasonality, demands innovative solutions that can adapt to these changes over time. Ensuring the model's robustness and generalizability across different cloud environments and workload types also presents a considerable obstacle.

1.1.4 Contribution

This research contributes a novel conceptual framework for a lightweight hybrid model designed for CPU usage prediction in cloud environments. This model distinguishes itself by strategically integrating information from both the time and frequency domains to enhance prediction accuracy while minimizing computational complexity. The key contributions include:

- A comprehensive analysis of the benefits of combining time and frequency domain information for cloud workload forecasting.
- A proposed architecture for a lightweight model that leverages both domains to capture short-term dynamics and long-term periodic patterns efficiently.
- A focus on reducing the computational overhead associated with complex deep learning models, making the solution more practical for resource-constrained cloud environments.
- The significance of these contributions lies in their potential to improve the efficiency of cloud resource management, reduce operational costs, and enhance the performance and reliability of cloud services. By providing a more accurate and lightweight prediction model, this research aims to advance the state of the art in cloud computing and contribute to more sustainable and cost-effective cloud infrastructures.

1.1.5 Organization

The remainder of this report is organized as follows: Section 2 presents a comprehensive review of related works, covering foundational concepts in time series forecasting for cloud workloads, CPU usage prediction techniques, the application of frequency domain analysis, and the development of lightweight models that combine time and frequency information. Section 3 details the proposed methodology, introducing the overall approach, model architecture, and a chronological breakdown of key components such as Fourier Transform, SENet, CNN, and LSTM layers. Section 4 discusses the experimental setup, including dataset descriptions, data preprocessing procedures, training strategies, evaluation metrics, and implementation specifics, followed by an analysis of the obtained results. Finally, Section 5 concludes the report by restating the research objectives, summarizing the key findings, and discussing the broader implications of the work.

Chapter 2

Related Works

2.1 Introduction

Cloud computing plays a pivotal role in providing scalable infrastructure and computing resources to organizations and companies [85]. Its popularity has surged in recent years and is expected to grow further due to advancements in AI and Big Data, which are resource-intensive technologies that often rely on cloud services for effective execution. One of the key advantages of cloud computing is its ability to replace fixed capital investments with pay-on-demand services, allowing firms to scale and reorganize their resources according to changing requirements [36].

Cloud platforms facilitate easy access to a wide pool of resources, including physical machines, servers, applications, storage, networking, and more. The pay-on-demand model necessitates dynamic resource allocation to match fluctuating demand. To provide optimal resource availability, cloud providers must continuously perform resource provisioning and de-provisioning. Under-provisioning can lead to violations of Service Level Agreements (SLAs) and degrade Quality of Service (QoS), resulting in customer dissatisfaction and tangible business losses. Conversely, over-provisioning leads to wasted energy and increased operational costs.

Given the critical nature of applications and services deployed on major cloud platforms such as Amazon Web Services (AWS), Microsoft Azure, IBM Cloud, Alibaba Cloud, and Google Cloud Platform (GCP), precise and timely resource management is essential. Many organizations identify managing cloud expenditure as one of their top challenges. Cloud computing's pay-for-use model enables users to pay solely for the resources they consume [109], making accurate resource management and usage prediction vital to ensuring high performance for end users [51], [108].

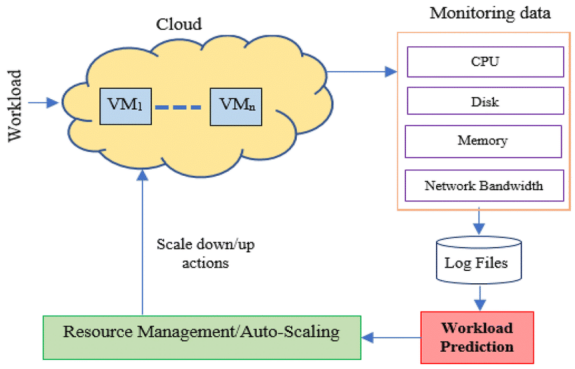


Figure 2.1: Workload Prediction using Machine Learning Scheme

Workload forecasting in cloud computing is challenging due to irregular, spiky, and heterogeneous usage patterns [150]. Traditional statistical models like ARMA, ARIMA, and SARIMA assume linearity and stationarity, making them effective in controlled settings but inadequate for large-scale, non-stationary cloud data. To overcome this, machine learning (ML) models such as Support Vector Machines (SVM), Linear Regression, and Random Forests have been explored. While better at capturing complex patterns, these face issues with high-dimensional data, feature engineering, and scaling non-linear dynamics.

Deep learning (DL) models—like Multi-Layer Perceptrons (MLP), Radial Basis Function (RBF) networks, and Long Short-Term Memory (LSTM) networks—have shown improved performance in modeling temporal dependencies and non-linear trends. However, their complexity introduces risks of overfitting and slow inference, especially in real-time or resource-limited settings. Most models also focus on point forecasts, ignoring prediction uncertainty. Uncertainty-aware forecasting, which models aleatoric (data) and epistemic (model) uncertainty, enables better decision-making and dynamic safety margins.

Despite progress, gaps remain in task-level resource prediction and hybrid approaches that combine statistical and ML models. These strategies could improve accuracy for key cloud tasks like scheduling, VM provisioning, and capacity planning. Given the temporal nature of workloads, recent research favors time series forecasting (TSF), which explicitly models sequential dependencies, making it more effective than traditional models in dynamic cloud environments.

This report presents a comprehensive overview of time series forecasting (TSF) advancements, highlighting the transition from classical to modern techniques. It focuses on:

- Tracing the evolution from statistical models to machine learning, deep learning, and generative approaches.
- Analyzing strengths, limitations, and performance of various forecasting methods.

- Summarizing current state-of-the-art practices in TSF for cloud workload prediction.

2.2 Fundamental Concepts

2.2.1 Time Series and Time Series Prediction

Time series prediction is the task of forecasting future values based on previously observed temporal data points. Classical statistical models such as ARIMA and its variants have been widely used for this purpose, primarily relying on assumptions of linearity and stationarity to capture trends and seasonality. While effective in simpler or controlled settings, these models often fall short in representing the complex, non-linear patterns observed in real-world data.

To address these limitations, the field has evolved toward machine learning (ML) and deep learning (DL) methodologies. ML models like Support Vector Machines (SVMs), Linear Regression, and Random Forests offer greater flexibility in handling non-linear and high-dimensional data but may require extensive feature engineering. DL models, including Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Transformer architectures, have shown superior capabilities in learning long-range temporal dependencies and capturing intricate dynamics directly from raw sequences. [26], [110]

Recent advances include hybrid models that integrate both time-domain and frequency-domain features to enhance prediction accuracy. Furthermore, generative approaches such as diffusion models have started gaining traction for their robustness and ability to model the uncertainty and variability inherent in time series data. [166] These developments reflect the growing need for precision, adaptability, and uncertainty-awareness in time series forecasting across dynamic and complex domains.

2.2.2 Workload Prediction

A cloud workload refers to the specific applications, services, or computing tasks that run within a cloud environment, using cloud resources like virtual machines, storage, and networking. Workload prediction refers to the process of forecasting future demand or usage patterns of cloud resources based on past or real-time data. This workload typically includes metrics related to compute, storage, memory, or network usage generated by running applications or services. A typical workload prediction model consists of the following stages:

- Collecting historical data on resource usage

- Performing data preprocessing and extracting relevant features
- Applying predictive models based on statistical or machine learning forecasting techniques to estimate future demand
- Incorporating the actual observed usage as new input for subsequent prediction cycles

The purpose of workload prediction is to perform Auto-scaling decisions, Load balancing, Energy efficiency, Cost optimization, SLA compliance and so on.

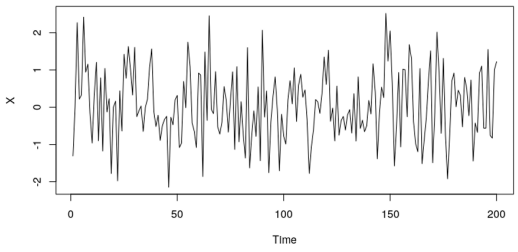
2.2.3 Workload Forecast Modeling

In the context of cloud resource forecasting, selecting an appropriate forecasting strategy is critical due to the dynamic and volatile nature of workload patterns. Two dominant paradigms are commonly adopted: *point forecast modeling* and *probabilistic forecast modeling*. Point forecasting aims to predict specific future values of resource metrics (e.g., CPU, memory, or bandwidth usage), and is typically implemented using either one-step or multi-step forecasting strategies. One-step models such as LSTNet[86], Deep Belief Networks (DBNs)[59], and Support Vector Regression (SVR)[40] are designed to predict the immediate next value in the time series, whereas multi-step models like N-BEATS[117], Stacked Autoencoders[155], DCRNN[95], and various decomposition- or clustering-based frameworks[11], [172] are used to forecast over longer horizons, often incorporating temporal dependencies and structural trends.[13]

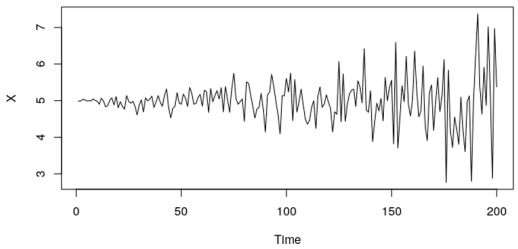
On the other hand, *probabilistic forecasting* seeks to estimate the entire probability distribution of future resource usage, allowing the model to capture both *aleatoric uncertainty* (stemming from data variability) and *epistemic uncertainty* (arising from model limitations). This is particularly useful in cloud environments where decisions about auto-scaling, provisioning, or SLA enforcement benefit from uncertainty-aware predictions. While still an emerging area, probabilistic models are increasingly recognized for their ability to support more robust and risk-aware resource management strategies in cloud systems[13], [132].

2.2.4 Stationarity

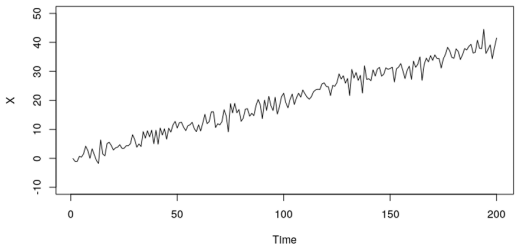
Stationarity is a key concept in time series analysis that refers to the consistency of a process's statistical properties over time. In the context of cloud workload forecasting, identifying whether the time series is stationary is essential, as many statistical models (e.g., ARIMA) assume this property for accurate and stable prediction. A stationary process is



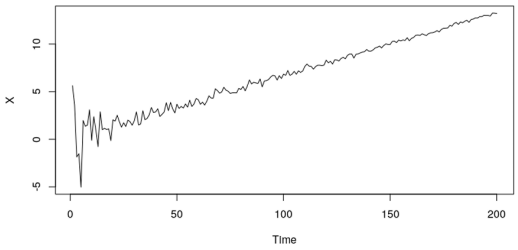
(a) Weak stationarity



(b) Mean stationarity and variance nonstationarity



(c) Mean nonstationarity and variance stationarity



(d) Mean and variance nonstationarities

Figure 2.2: Illustration of stationarity and nonstationarity on simulated time series.

one where characteristics such as mean, variance, and autocovariance do not change over time, making the series more predictable and easier to model.

Time series data exhibiting non-stationarity—such as trends, seasonal effects, or variance shifts—often require transformation techniques like differencing or detrending before modeling. Understanding the type of stationarity present helps in choosing appropriate preprocessing steps and forecasting models. In the following subsections, we formally distinguish between two major types of stationarity: *strong stationarity* and *weak stationarity*, each with different assumptions and practical implications.[35]

Strong Stationary

A stochastic process X is said to be **strongly stationary** if its joint probability distribution is invariant under time shifts. Formally, X is strongly stationary if for all $T \in \mathbb{N}$ and $\tau \in \mathbb{N}^*$, the following holds:

$$P(X_1, X_2, \dots, X_T) = P(X_1 + \tau, X_2 + \tau, \dots, X_T + \tau) \tag{2.1}$$

This means the full distribution of the process remains constant over time, making the statistical behavior completely time-invariant. A notable example is any independent and identically distributed (i.i.d.) process, which is inherently strongly stationary.

In the context of cloud resource forecasting, such as predicting CPU usage or memory

consumption, strong stationarity is often an overly strict assumption. Real-world cloud workloads are affected by time-dependent behaviors like diurnal cycles, user activity patterns, and periodic task execution, which violate the conditions of strong stationarity. Moreover, testing for strong stationarity in empirical data is practically infeasible, making it less useful in applied settings. As a result, most forecasting models focus on the more tractable and realistic notion of weak stationarity.

Weak Stationary

A stochastic process X is considered **weakly stationary** (or second-order stationary) if its first and second moments—mean, variance, and autocovariance—are time-invariant. Specifically, X is weakly stationary if the following conditions hold:

$$E[X_t] = \mu \quad \text{(constant mean)}$$

$$Var(X_t) = \gamma(0) < \infty \quad \text{(finite constant variance)}$$

$$Cov(X_t, X_{t+h}) = \gamma(h) \quad \text{(autocovariance depends only on lag } h\text{)}$$

Here, $\gamma(h)$ is a symmetric and bounded autocovariance function, meaning $\gamma(-h) = \gamma(h)$ and $\gamma(h) \leq \gamma(0)$.

Weak stationarity is widely adopted in time series modeling due to its practical applicability and reduced assumptions. In cloud environments, workload metrics like CPU usage or network traffic often demonstrate non-stationary behavior due to user-driven dynamics and workload bursts. However, these series can often be transformed into weakly stationary forms through techniques such as differencing, detrending, or seasonal adjustment. Establishing weak stationarity is a crucial step before applying models like ARIMA or state-space models, which require a stable mean and variance over time to produce reliable forecasts.

Testing Stationarity

Stationarity is a desirable property in time series modeling, as it implies that statistical characteristics such as mean, variance, and autocorrelation remain constant over time. This stability simplifies the modeling process and improves the reliability of forecast models, particularly those that rely on fixed underlying distributions, such as ARIMA. In cloud resource forecasting, where metrics like CPU utilization, memory consumption, and network throughput often fluctuate with user activity or system load, determining whether the data is stationary is essential for selecting appropriate forecasting techniques.

To achieve stationarity, various preprocessing techniques may be applied. Differencing, which involves subtracting the previous value from the current one, is commonly used to eliminate trends. Seasonal differencing addresses recurring patterns in periodic workloads, while transformations such as logarithmic scaling or Box-Cox transformation help stabilize variance across time.

Testing for stationarity can be performed using both graphical and statistical methods. Graphical approaches include time series plots, rolling mean and variance plots, correlograms, and covariograms to visually assess changes in distribution and autocorrelation. Statistical tests offer a more formal diagnosis and are broadly classified into time-domain and frequency-domain methods. Time-domain tests include unit root tests such as the Augmented Dickey-Fuller (ADF), Phillips-Perron, and KPSS tests, which evaluate the presence of non-stationary trends. Frequency-domain approaches, including those proposed by Priestley and Rao or Paparoditis, rely on spectral decomposition of the signal. Furthermore, structural break tests such as the Zivot-Andrews and CUSUM tests can detect sudden changes or regime shifts in the time series, which are common in cloud workloads due to software updates, user behavior changes, or scaling events.

These tests are typically applied as part of a preprocessing pipeline before fitting forecasting models to cloud resource data.

2.2.5 Forecasting Challenges

Time series forecasting (TSF) for cloud resource management presents several domain-specific challenges due to the dynamic and complex nature of cloud workloads. Metrics such as CPU utilization, memory usage, and network throughput often demonstrate irregular behavior, which can hinder the effectiveness of traditional forecasting models.

One of the primary challenges is **non-stationarity and volatility** in workload patterns. Cloud workloads frequently experience abrupt spikes due to user traffic surges, job scheduling, or application scaling events. Traditional statistical models like ARIMA or Exponential Smoothing (ETS) are based on assumptions of stationarity and linearity, making them less effective in such dynamic environments [152].

Cloud workloads also exhibit **multiscale patterns**, combining short-term bursts with long-term seasonal or trend components. Capturing such multi-resolution dependencies often requires hybrid or hierarchical models. Deep learning architectures like N-BEATS [117] and Autoformer [162] have been proposed to address this by integrating temporal decomposition and attention mechanisms.

Noise and anomalies, such as those caused by monitoring errors, system failures, or

network interruptions, introduce additional unpredictability. These outliers can mislead forecasting models, especially in real-time deployments. Robust models like Long Short-Term Memory (LSTM) networks or GARCH models have been explored to handle volatility and conditional variance effectively [19], [60].

The **cold start problem** remains a significant issue, particularly for new services or virtual machines that lack sufficient historical data. In such cases, transfer learning, meta-learning, or clustering-based similarity transfer can help leverage patterns from related workloads [154], [178].

Moreover, **real-time constraints** impose limitations on model complexity. Although Transformer-based models offer strong performance, their inference latency and memory footprint are high. Lightweight alternatives such as Temporal Convolutional Networks (TCNs) [9] and gradient-boosted models like LightGBM are often preferred in production environments where low-latency predictions are critical.

Forecasting cloud workloads also requires understanding **multivariate dependencies**, as resource usage is rarely isolated. For example, CPU, memory, and I/O consumption often correlate. Ignoring these relationships can reduce model accuracy. Graph Neural Networks (GNNs) and multivariate RNNs have been employed to learn such interdependencies [94].

Another challenge is **concept drift**, where the statistical properties of the data change over time due to evolving user behavior or workload scheduling policies. This necessitates adaptive models capable of online learning or drift detection to maintain long-term accuracy [45].

Finally, effective **uncertainty quantification** is essential for risk-aware resource management. Probabilistic forecasting approaches, such as DeepAR [132] and quantile regression, provide confidence intervals that can guide decisions like safe auto-scaling and provisioning.

Emerging solutions include hybrid models that combine statistical and deep learning methods, efficient Transformer variants like Informer [179], and automated model selection techniques using AutoML frameworks. These trends aim to balance accuracy, adaptability, and computational efficiency in the context of real-world cloud systems.

2.2.6 Evaluation Metrics

The performance of forecasting models was assessed using multiple error-based metrics, each capturing different aspects of predictive accuracy. Table 2.1 summarizes the definitions of the employed metrics.

Here, y_t denotes the actual value, $\hat{y}_t$ the predicted value, and n the number of observations. MAE provides a straightforward average deviation and is less sensitive to outliers compared to squared-error metrics[68]. Huber loss balances between MAE and MSE[67], being quadratic for small errors and linear for large ones, making it robust yet differentiable. MSE and RMSE emphasize larger errors due to squaring[69], with RMSE retaining interpretability by expressing errors in the same units as the data[69]. NMSE enables scale-free comparisons by normalizing against the variance of the actual series[160]. Percentage-based metrics such as MAPE and sMAPE facilitate intuitive interpretation, though MAPE can be unstable when $y_t = 0$ [69]. MASE addresses scale dependence by benchmarking errors against a naive forecast[69], while WAPE provides a relative measure of total deviation with respect to the magnitude of the data and is a less standardized metric and more industry-specific. Together, these metrics offer a comprehensive assessment of forecasting accuracy, balancing robustness, interpretability, and comparability across datasets.

2.3 State of the Art in Survey Literature

This section reviews existing surveys and comparative frameworks related to cloud workload forecasting and adjacent domains. We provide paper-by-paper overviews — summarizing each work's objectives, methodology coverage, and key insights — to clearly position our survey within this landscape. Table 2.3 cross-references the major contributions and gaps in descending chronological order.

Ding et al. (2024) – ***Application-Oriented Cloud Workload Prediction*** [44] Ding and Feng (Tsinghua Science and Technology, 2025) develop an "application-oriented" taxonomy that links workload prediction techniques to application characteristics and life-cycle management tasks. They analyze variability and heterogeneity in cloud workloads, categorize long- and short-term resource management approaches (e.g., proactive capacity planning, dynamic migration, elastic scaling), and highlight technical challenges including serverless environments, model interpretability, and unreliability in large-scale deployments :contentReference[oaicite:0]index=0.

Wang et al. (2025) – ***Efficient Model Selection via LLMs*** [158] Wang et al. propose using LLMs (e.g., LLaMA, GPT, Gemini) to automate time series forecasting model selection. Their approach replaces costly performance matrices with LLM reasoning capabilities, achieving better selection performance and significantly lower computational overhead in experiments — showcasing a promising direction for efficient meta-forecasting :contentReference[oaicite:1]index=1.

Zhang et al. (2024) – ***Large Language Models for Time Series*** [175] This IJCAI 2024 survey presents a taxonomy of methods for applying LLMs to time series: direct prompting, quantization, aligning techniques, leveraging vision modalities, and tool integration. It also catalogs multimodal datasets, discusses deployment challenges, and maintains an active GitHub repo for research continuity :contentReference[oaicite:2]index=2.

Ye et al. (2024) – ***Time Series Foundation Models*** [169] Ye et al. introduce a "3E" framework—Effectiveness, Efficiency, Explainability—to evaluate foundational LLM-based models tailored to time series tasks. They review both pre-training from scratch and adapting existing language models, offering domain taxonomies, resource repositories, and a maintained GitHub with related datasets :contentReference[oaicite:3]index=3.

Other notable surveys

Several earlier surveys provide important foundations. Lim & Zohren (2021) [96] articulate theoretical underpinnings of DL models (RNNs, LSTMs, Transformers) for sequential

data, enriching the technical depth of modern forecasting **index=4**. Torres et al. (2021) [152] expand on architectural alternatives (CNNs, attention-based DL) complementing Lim & Zohren's focus—though they don't address cloud-specific applications. Masdari & Khoshnevis (2020) [107] present a comprehensive method taxonomy across statistical, ML, DL, and hybrid models for cloud workload forecasting. Parmezan et al. (2019)[118] benchmark 11 forecasting methods across 95 datasets (incl. ARIMA, ETS, SVM, MLP, kNN), providing valuable empirical baselines. Earlier background context is provided by Amiri & Mohammad-Khanli (2017) [5], Lorido-Botran et al. (2014) [99], Manvi & Shyam (2014) [106], and Islam et al. (2012)[71], tracing forecasting's evolution from statistical to early neural approaches.

Table 2.3 provides a comparative view of existing survey papers against our own work. What becomes clear from this mapping is that prior surveys, while valuable, tend to emphasize only a subset of the dimensions relevant to modern workload forecasting. Some are strongly methodological, focusing on statistical or machine learning techniques; others explore deep learning models or autoscaling frameworks in isolation. A few recent efforts even move toward meta-level tools such as LLM-based model selection. Yet none of these works bring together the full range of perspectives that today's cloud environments demand.

Our survey distinguishes itself in several ways. First, it integrates **the entire methodological spectrum**—from classical statistical baselines to advanced deep learning, hybrid ensembles, Transformers, and even emerging meta-learning approaches. Second, it goes beyond purely algorithmic concerns by addressing **sustainability and FinOps**, recognizing that energy efficiency and cost optimization are now central to workload management in practice. Third, we emphasize **operational and deployment realities**, including cross-cloud benchmarking, dataset diversity, and the role of forecasts in autoscaling pipelines. These dimensions are either lightly treated or entirely absent in prior surveys.

In short, while earlier reviews have laid important groundwork, they remain partial in scope. By combining **methodological breadth, sustainability awareness, and deployment relevance within a single unified framework**, our survey provides a more complete and forward-looking reference point for CPU workload forecasting in cloud computing.

2.4 Datasets

Cloud computing platforms experience dynamic workloads arising from diverse applications such as microservices, batch processing, and serverless tasks. Predicting these workloads is essential for autoscaling, anomaly detection, and system design. Public

datasets are pivotal in enabling researchers to develop and evaluate predictive models under real or simulated conditions.

2.4.1 Dataset Descriptions

Google Cluster Traces

Google released several versions of cluster workload traces from their Borg system. The first version (v1) [128] covers a 7-hour trace on a single machine, primarily for testing purposes. The second version (v2) [128] contains 29 days of data from 12,500 machines, with detailed machine and task-level event records such as job submissions, scheduling, and resource usage. The latest version (v3) [159] extends this to one month, capturing workload data from 96,000 machines across 8 clusters with richer usage statistics including CPU and memory percentiles.

Alibaba Cluster Traces

Alibaba's trace datasets provide insights into workloads in a large-scale datacenter environment. Version 1 [2] offers 12 hours of data across 1,300 machines featuring co-located online and batch workloads. Version 2 [3] extends this to 8 days with expanded metadata covering containers and batch instances on 4,000 machines.

Azure Public Datasets

Microsoft's Azure workload traces consist of two major datasets. Version 1 [33] includes resource usage logs of about 2 million VMs over 30 days, featuring CPU utilization and VM configuration details. Version 2 [137] enhances this with more detailed resource buckets and covers 2.6 million VMs over 30 days, enabling fine-grained analysis of serverless and traditional workloads.

Alibaba Microservices Traces

The Alibaba microservices datasets focus on containerized microservices running on bare-metal nodes with complex inter-service dependencies. Version 1 [101] captures 12 hours of resource and communication metrics for over 1,300 bare-metal nodes. Version 2 [102] expands this to 13 days covering 40,000+ bare-metal nodes and 470,000+ containers, suitable for autoscaling research.

Additional Datasets

Other notable datasets include the Business Critical Workloads trace from 2015 (1 month, 1,250 machines), IBM Docker Registry Trace [7] capturing 75 days of registry access logs, the PlanetLab Dataset (2011), Scout OSR Dataset [62] focused on Amazon EC2 container metrics, Dionatrafk `https://github.com/dionatrafk/workload_prediction/tree/master` (univariate CPU traces), Intel Netbatch Logs, and OpenCloud Hadoop Workloads spanning multiple months.

2.4.2 Classification by Columns

Public datasets for cloud workload prediction exhibit significant diversity in scale, workload characteristics, available metrics, and system architecture. Understanding these dimensions is critical for selecting appropriate datasets for specific research goals and for benchmarking predictive models. We classify the datasets surveyed along five key dimensions: number of Machines, type of workload, resource metrics, time span and system architecture.

Number of Machines

The size of datasets varies widely, ranging from small academic clusters with hundreds of machines to hyperscale commercial clouds comprising millions of virtual machines (VMs). For example, Google Cluster Traces v3 include data from approximately 96,000 machines [159], while the Azure datasets encompass over 2 million VMs [33]. Dataset scale impacts both the complexity of modeling and the representativeness of workload diversity.

Workload Type

Datasets capture various workload types, which can be broadly categorized as:

- **Batch Jobs:** Offline, compute-intensive tasks typically with high resource demand but flexible timing (e.g., Intel Netbatch Logs).
- **Online Services / Microservices:** Latency-sensitive, continuously running services often containerized (e.g., Alibaba Microservices Traces [101], [102]).
- **Hybrid Workloads:** Mixed batch and online workloads colocated on shared infrastructure (e.g., Alibaba Cluster Traces [2], [3]).
- **Serverless / Function-as-a-Service (FaaS):** Short-lived, event-driven functions (e.g., Azure Dataset v2 [137]).

The workload type influences workload dynamics, predictability, and relevant resource metrics.

Time Span

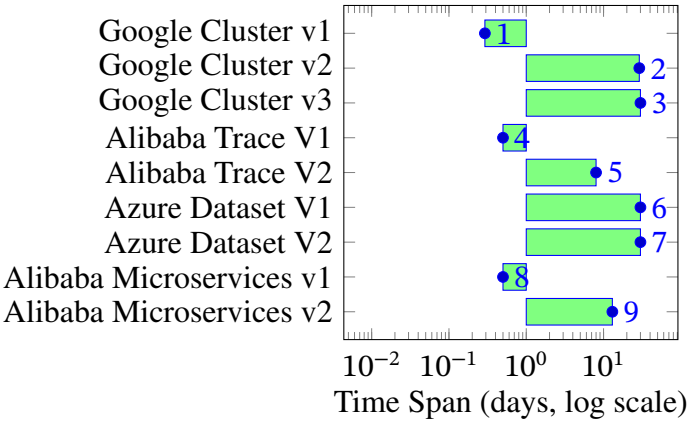


Figure 2.3: Time Span of Public Cloud Workload Datasets (log scale in days)

Datasets differ in the duration over which data was collected, ranging from several hours to multiple months. Longer traces allow models to learn from seasonal and trend patterns, while shorter traces may focus on specific workload bursts or scenarios. For example, Google Cluster v2 covers 29 days [128], while OpenCloud Hadoop workloads span over 20 months [120].

Architecture

Cloud and cluster management frameworks impact trace content and semantics. Key architectures include:

- **Cluster Managers:** Borg (Google), Yarn (Hadoop), and proprietary datacenter orchestrators.
- **Virtualization:** Virtual machines vs. containers.
- **Bare-metal vs. Virtualized Nodes:** Alibaba Microservices traces are collected from bare-metal nodes [101], while Azure datasets include VM-level metrics [33].

Understanding system architecture is necessary for correctly interpreting and using workload traces.

2.5 Taxonomy

Forecasting CPU workloads is not a one-size-fits-all problem. Over the years, researchers have explored a wide spectrum of approaches, each motivated by different assumptions, data availability, and performance goals. To make sense of this variety, it is helpful to organize them within a clear taxonomy.

As shown in Figure 2.4, the forecasting models can be grouped into four broad families. Statistical methods provide interpretable, mathematically grounded baselines. Machine learning approaches adaptively learn patterns from data. Deep learning architectures capture complex temporal dependencies at scale. Finally, hybrid models combine the strengths of multiple paradigms to achieve more robust forecasting.

This structure offers a big-picture roadmap: it not only shows where each technique belongs but also highlights the evolution of methods from traditional, theory-driven models to modern, data-centric and integrative solutions. In the following subsections, we will explore each category in detail, unpacking their principles, representative techniques, and application contexts.

1) Statistical Methods

Statistical approaches are foundational in time series forecasting and rely on assumptions such as linearity, stationarity, and decomposability. They are interpretable, data-efficient, and computationally light, making them suitable for quick baselines and scenarios with limited data. This section outlines the main categories and highlights notable works applying these models to workload forecasting.

Smoothing-Based Models: Methods such as Simple Exponential Smoothing (SES), Holt's Linear Trend, and Holt-Winters (HW) models apply exponentially decreasing weights to older observations:

$$\hat{y}_{t+1} = \alpha y_t + (1 - \alpha)\hat{y}_t, \tag{2.2}$$

where $\alpha \in [0, 1]$ controls the decay rate. Holt-Winters extends this by modeling both trend and seasonality.

Exponential smoothing methods, first introduced by Holt (2004)[49] and Gardner (1985)[48], provide foundational forecasting techniques that have been successfully applied in various contexts, including Baldán et al.'s (2016)[10] automated ETS modeling for Google Cluster traces, which demonstrated ease of use and reliable baseline performance. However, com-

parative evaluations by Parmezan et al. (2019)[118] of SES, additive, and multiplicative Holt-Winters models on synthetic and mixed datasets revealed performance degradation when handling complex, dynamic workloads. While these methods offer advantages in simplicity, interpretability, and computational efficiency, they are fundamentally limited by poor performance with non-linear patterns, multiple seasonalities, or abrupt changes in data behavior. Kumar and Mazumdar **kumar2016** benchmarked ARIMA and related ARMA-class models against singular spectrum analysis (SSA) on Wikimedia HPC traces, finding that plain ARIMA outperformed more complex ARMA variants for bursty network workloads, while SSA excelled for CPU and RAM series. Li et al. **li2013** proposed an ARIMA predictor with dynamic error compensation to drive a time-based multi-VM provisioning mechanism, reducing SLA violations and rental cost on real cloud traces.

Autoregressive Models: This class includes AR, MA, ARMA, and ARIMA, which model current values as combinations of past values and errors. The general ARIMA(p, d, q) model is:

$$y'_t = \sum_{i=1}^{p} \phi_i y'_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \epsilon_t, \tag{2.3}$$

where d is the order of differencing for stationarity.

ARIMA models, developed by Box and Jenkins (1970)[20], have been extensively applied to workload forecasting, with notable implementations including Calheiros et al.'s (2015)[21] proactive provisioning system for Wikimedia workload traces and Baldán et al.'s (2016)[10] automated baseline approach for Google Cluster data, though the latter exhibited residual autocorrelations. Parmezan et al. (2019)[118] conducted comprehensive benchmarks of ARIMA/SARIMA variants across mixed datasets, while Hu et al. (2014)[66] incorporated AR models into hybrid workload prediction systems. Kumar and Mazumdar **kumar2016** benchmarked ARIMA and related ARMA-class models against singular spectrum analysis (SSA) on Wikimedia HPC traces, finding that plain ARIMA outperformed more complex ARMA variants for bursty network workloads, while SSA excelled for CPU and RAM series. Li et al. **li2013** proposed an ARIMA predictor with dynamic error compensation to drive a time-based multi-VM provisioning mechanism, reducing SLA violations and rental cost on real cloud traces.

Despite their strengths in modeling autocorrelation and effectiveness for short-term univariate forecasting, ARIMA models are constrained by requirements for stationarity, sensitivity to parameter tuning, and limitations in capturing non-linear dynamics.

Decomposable & Structural Models: These models explicitly model trend, seasonality,

and irregular components. Prophet by Facebook is a notable implementation:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t, \tag{2.4}$$

where $g(t)$ is the trend, $s(t)$ is seasonality, $h(t)$ accounts for holidays, and ϵ_t is noise.

Hidden Markov Models, grounded in the classical theory established by Rabiner (1989)[125] including forward-backward and EM algorithms, have been applied to workload forecasting with notable success in Khan et al.'s (2012)[78] implementation for VM cluster usage prediction in private cloud settings, where they effectively captured distinct usage modes including idle, moderate, and high activity states. Di et al. **di2012** applied a Bayesian model to Google cluster traces and achieved 5.6–50% lower mean-squared error compared with moving-average and autoregressive baselines for host-load prediction, while Khan et al. **khan2012** used Hidden Markov Models on real cloud traces to capture VM group usage modes and accurately forecast workload pattern changes. While HMMs excel at capturing hidden regimes and group dynamics within complex systems, they present significant challenges in terms of complex training procedures, the critical need for appropriate state number tuning, and the frequent assumption of discrete states that may not adequately represent continuous workload variations.

2) Machine Learning Models

Machine Learning (ML) models approach CPU forecasting as a supervised regression task. Unlike traditional statistical models, they make fewer assumptions about data stationarity or linearity and are capable of learning complex, non-linear dependencies. However, they generally require larger datasets and thoughtful feature engineering. Below we describe the theoretical underpinnings of key ML model categories, along with major applied studies and their findings.

Workload–Absorber Based Forecasting

Tian et al. [149] designed a *workload-absorber* algorithm for cloud video-on-demand services that minimizes content reorganization and tolerates imperfect workload prediction using real VoD access traces, enabling cost-efficient and scalable dynamic server provisioning. Lyu et al. (2016) [103] developed a three-module forecast mechanism using generic machine-learning techniques on a simulated Eucalyptus cloud to balance availability and economic profit. Zhou et al. [180] proposed an AHP-based load evaluation combined with an HHGA-optimized RBFNN for dynamic load balancing in cloud storage, showing improved prediction for weighted round-robin scheduling.

Support Vector Regression (SVR): SVR learns a function $f(x)$ that approximates observed values within a tolerance ϵ, using kernel functions $K(x_i, x)$ to model non-linear relations:

$$f(x) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) K(x_i, x) + b \tag{2.5}$$

Support Vector Regression has demonstrated mixed effectiveness in workload prediction across various implementations, with Hu et al. (2014)[64] applying kernel-based SVR to public cloud traces and achieving good short-term CPU burst prediction despite noted sensitivity to kernel choice, while Gao et al. (2020)[47] conducted comparative evaluations against ARIMA, BRR, and LSTM on Google Cluster data, revealing SVR's strength on small datasets but weak performance on pathological workloads. Parmezan et al. (2019)[118] further demonstrated SVR's stability across mixed workloads when appropriate hyperparameter tuning is applied, highlighting both the method's potential effectiveness and its critical dependence on proper configuration and dataset characteristics. Dietrich et al. (2010) [38] proposed a Least-Mean-Squares linear predictor for dynamic voltage and frequency scaling in gaming, achieving power savings comparable to tuned PID controllers. Tarsa et al. [147] applied hierarchical sparse coding with SVM classifiers to predict instruction-level workloads for adaptive on-chip power scaling, showing superior accuracy over linear regression.

Hybrid SVR Models: Hybrid SVR approaches have emerged to address limitations of standard SVR methods, with Hu et al. (2014)[64] proposing Kalman-Smoothed SVR (KSwSVR) that demonstrated improved adaptation and denoising capabilities on realistic cloud traces, while Zhong et al. (2019)[176] introduced a Wavelet+SVM hybrid (WSVMLF) that proved effective in capturing multi-scale variations in synthetic workloads despite sensitivity to wavelet basis selection. Complementing these approaches, Bayesian Ridge Regression (BRR), a probabilistic linear regression technique that estimates model weights under Gaussian noise assumptions to promote regularization, was applied by Gao et al. (2020)[47] on Google Cluster traces, where it exhibited robust and interpretable behavior while remaining limited in its ability to capture non-linear dynamics. Nikravesh et al. (2015) [116] compared SVM and neural-network predictors on Amazon EC2 using the TPC-W benchmark and found SVM better for periodic or growing workloads while NN handled unpredictable ones.

Bayesian Ridge Regression (BRR): A probabilistic linear regression technique, BRR estimates model weights $w \sim \mathcal{N}(0, \alpha^{-1} I)$ under Gaussian noise, promoting regularization: Bayesian Ridge Regression, as a probabilistic linear regression technique that estimates model weights under Gaussian priors to promote regularization, has been specifically

evaluated in workload prediction contexts by Gao et al. (2020)[47] on Google Cluster traces, where it demonstrated robust and interpretable behavior while maintaining computational efficiency, though its linear nature fundamentally limits its capacity to capture the non-linear dynamics often present in complex workload patterns.

Neural Networks: Neural network approaches to workload prediction have evolved from basic implementations to more sophisticated evolutionary methods, with basic backpropagation neural networks (BPNNs) employed by Hu et al. and Parmezan et al. demonstrating simple non-linear modeling capabilities while struggling with generalization issues and local minima convergence problems. More advanced evolutionary neural networks, as exemplified by Mason et al. (2018)[111], utilized optimization algorithms including differential evolution (DE), particle swarm optimization (PSO), and covariance matrix adaptation evolution strategy (CMA-ES) to train recurrent neural networks on CPU logs, achieving superior generalization performance albeit at significantly higher computational cost. Wamba et al. [156] introduced neural network and constraint programming models for cloud workload prediction and synthetic trace generation, validated on real cloud traces. Kumar et al. [84] combined neural networks with self-adaptive differential evolution for workload prediction on NASA and Saskatchewan HTTP traces, significantly reducing prediction error.

Neuro-Fuzzy Systems: Neuro-fuzzy systems have been explored as hybrid approaches combining neural learning with fuzzy logic interpretability, with Zhong et al. (2019)[93] applying Adaptive Network-based Fuzzy Inference Systems (ANFIS) on synthetic traces to provide rule-based interpretability, though this approach remains susceptible to the curse of dimensionality in high-dimensional feature spaces. Additionally, Toopchinezhad and Ahmadi (2024)[151] proposed the aSNFAQM neuro-fuzzy controller for Active Queue Management simulations, which demonstrated adaptive capabilities but was specifically designed for network-oriented applications rather than CPU-specific workload prediction scenarios.

Clustering-Enhanced ML: Ensemble and clustering-based methodologies have emerged as strategies to improve forecasting accuracy through decomposition and specialized modeling, with Gao et al. (2020)[47] combining clustering techniques including K-Means and DBSCAN with per-cluster predictive models on Google Cluster traces, resulting in increased forecast accuracy while maintaining dependency on cluster quality and coherence. Similarly, Smendowski and Nawrocki (2024)[143] proposed Similarity-based Temporal Grouping (STG) to enable group-wise tailored modeling approaches that can adapt to different workload characteristics, though the generalization capability of this method to real-world data beyond controlled experimental conditions remains unclear. Zhang et al.

(2014) [173] combined pattern matching and collaborative filtering to identify service workload patterns for quality-driven dynamic cloud service configuration and auto-scaling.

Ensemble Learning: Advanced ensemble methods have demonstrated significant improvements in workload prediction through sophisticated model combinations, with Javeed et al. (2025) developing an ensemble approach combining Support Vector Machines, Random Forest, and XGBoost for request-based forecasting on real-world traces, achieving strong predictive performance while requiring complex feature engineering processes. Valarmathi and Kanaga (2021)[153] implemented Random Forest and LSTM ensembles on Alibaba Cluster v2018 data, demonstrating improved variance handling capabilities, though this approach operates at the boundary between traditional machine learning and deep learning methodologies due to LSTM's neural architecture. Duggan et al. (2013) [41] introduced a collaborative-filtering model to predict portable database workload performance across diverse IaaS platforms using TPC-H and TPC-DS benchmarks.

3) Deep Learning Models

Deep Learning (DL) models excel at capturing long-term dependencies, complex non-linearities, and interactions across multiple features without requiring explicit feature engineering.

Recurrent Neural Networks (RNNs): Traditional RNNs face vanishing gradient issues, limiting long-range dependency learning. They are mostly replaced by more advanced gated variants:

LSTM and GRU: Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) use gating mechanisms to regulate information flow through time steps. The GRU update equation is:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t, \tag{2.6}$$

where

- z_t is the update gate controlling retention of previous hidden states,
- $\tilde{h}_t$ is the candidate activation influenced by reset gate and current input,
- $\odot$ denotes element-wise multiplication.

GRUs simplify LSTM by combining forget and input gates, maintaining efficiency while modeling temporal dependencies.

[83] pioneered the use of LSTM-RNNs for workload forecasting on web server logs, achieving low MSE while encountering challenges with long sequence training. Prodan [136]

developed a pruned multivariate GRU with online adaptation capabilities, demonstrating improved efficiency and scalability for dynamic workloads. Song et al. [144] applied LSTM to multi-step-ahead cloud load prediction on Google and traditional distributed systems. Gupta and Dinesh [55] proposed multivariate and bidirectional LSTM models for cloud resource usage prediction, outperforming existing methods. Janardhanan and Barrett [73] forecasted CPU usage in data centers using LSTM and compared against ARIMA, showing superior nonlinear modeling. Nguyen et al. [114] combined RNNs with autoencoders in a stacked model for cloud/Grid workload prediction, achieving lower NRMSE than single models. Kumar and Singh [84] used a neural network with adaptive differential evolution for accurate workload forecasting on NASA HTTP traces. Kumar et al. [83] developed LSTM-RNNs for web server logs, reducing mean squared error. Sun et al. [145] proposed hybrid LSTM models to predict link quality confidence interval boundaries in smart grid wireless communications, validated on real-world traces.

Attention-Based Models: These models focus on important time steps—such as CPU spikes—improving interpretability and enhancing long-term predictions. Attention mechanisms can be integrated into LSTM architectures.

Zhu et al. [181] introduced an attention-based encoder-decoder LSTM for mixed workloads on Alibaba datasets, capturing complex temporal dependencies.

GAN-Based Models: Generative Adversarial Networks (GANs) like VTGAN and E2LG model workload distributions by having a generator predict future CPU load and a discriminator assess realism. These show promise for realistic workload simulation and anomaly detection.

Yazdanian and Sharifian [168] proposed E2LG, integrating Empirical Mode Decomposition with LSTM-GAN for multi-frequency modeling, and Maiyza et al. [104] presented VT-GAN, combining RNN generators with CNN discriminators for joint trend-value learning.

Advanced DL Architectures: Deep Belief Networks (DBNs) and **CPD-compressed DNNs** reduce network size for faster inference without sacrificing accuracy. Qiu et al. [123] employed a Deep Belief Network with stacked RBMs for unsupervised feature learning.Chen et al. [31] introduced L-PAW, which combines sparse autoencoders with GRU for high-dimensional data. (Note: L-PAW contains a GRU component, so it partially overlaps with RNN-like architectures.) Zhang et al. [174] developed a CPD-compressed deep learning model for virtual machine workload prediction on PlanetLab, achieving higher training efficiency and accuracy.

4) Hybrid Models

Hybrid models combine statistical preprocessing with learning-based prediction to leverage both interpretability and modeling flexibility.

Statistical + ML Hybrids: Statistical and machine learning hybrid models combine traditional time series preprocessing with learning-based prediction to leverage interpretability and modeling flexibility, encompassing approaches such as GRU+ES for input smoothing, Hu et al.'s[65] KSwSVR integrating Kalman filtering with SVR for noise reduction, Zhong et al.'s[177] WSVMLF utilizing wavelet decomposition with SVR for multi-resolution analysis, and Han et al.'s[57] comprehensive GARCH+CEEMDAN+GCN framework combining volatility modeling, signal decomposition, and graph networks for complex temporal dependency capture. Bi et al. (2018) combine wavelet decomposition and ARIMA to predict task arrivals in Google cloud data centers [18]. Shi et al. (2020) develop Block Hankel tensor ARIMA to forecast multiple short time series by exploiting low-rank correlations in traffic and electricity datasets [140]. Ramezani & Naderpour (2017) present a fuzzy VM workload prediction method using historical and current CPU data in real cloud environments [126]. Nguyen et al. (2019) combine opposition-based coral reefs optimization with neural networks (OCRO-MLNN) for efficient cloud workload forecasting on Google and real-world datasets [115]. Liu et al. (2017) propose an adaptive prediction approach that classifies workloads and assigns optimal prediction models using Google cluster data [97]. Mazumdar & Kumar (2016) evaluate hybrid Kalman-filtered ARIMA and wavelet-based methods for resource usage prediction on Wikimedia Grid traces [112]. Bi et al. (2024) integrate wavelet decomposition, Savitzky–Golay filtering, and ARIMA for multi-application workload prediction in Google traces [17]. Hai-hong et al. (2014) use classification-based methods for host load prediction to support VM migration and load balancing in real cloud systems [56]. Barati & Sharifian (2015) propose TSVR, a hybrid SVR model tuned via genetic and particle swarm optimization for load prediction in Google traces [12]. Cetinski and Juric [27] propose AME-WPC, which integrates statistical and learning methods and adapts to system-specific factors to improve prediction accuracy. Zhong et al. [176] present a weighted wavelet support vector machine optimized with particle swarm optimization to forecast host loads for energy-efficient resource scheduling. Singh et al. [142] introduce TASM, combining ARIMA, SVR, and linear regression with adaptive workload classification for web applications. Lu et al. [100] design RVLBPNN, a BPNN-based hybrid model for energy-efficient workload forecasting, demonstrating improved accuracy over HMM and Naïve Bayes classifiers. Sahni and Vidyarthi [131] propose a heterogeneity-aware auto-scaling heuristic that adapts to workload changes while maintaining QoS by employing online profiling and heuristic-based heterogeneous

resource provisioning. Jiang et al. [76] introduce a VM-level auto-scaling scheme for web applications, predicting request loads and making cost-latency optimal scaling decisions. Qazi et al. [122] leverage Chaos Theory and historical workload patterns to predict VM behavior and re-distribute workloads for efficient physical machine usage. Cao et al. [24] develop a dynamic ensemble model for CPU load prediction in cloud environments, combining multiple predictors with continuous optimization for improved performance. Shariffdeen et al. [138] propose an ensemble workload predictor for proactive auto-scaling in PaaS, integrating time series and machine learning models for accurate forecasts across varying patterns. Singh and Rao [141] present online ensemble learning methods to predict large-scale server workloads, achieving high prediction accuracy even under nonstationary conditions. Tang et al. [146] combine linear regression and wavelet neural networks in MLWNN for short-term workload prediction and energy-efficient job scheduling. Gandhi et al. [46] introduce a hybrid predictive and reactive provisioning strategy to minimize SLA violations and power consumption. Guo et al. [54] develop NUP, a type-aware hybrid predictor that dynamically adapts prediction algorithms based on workload type using linear regression, ARMA, and SVR.

Integrated Deep Learning: Integrated deep learning hybrid models represent sophisticated architectures that combine multiple neural network components for enhanced forecasting performance, including Karim et al.'s[77] BHyPreC system integrating CNN local feature extraction with LSTM sequence modeling and GRU refinement, Li et al.'s[92] BG-LSTM utilizing bidirectional and grid-LSTM architectures for multi-dimensional temporal dependency modeling, and Chen et al.'s[29] DeepGLO framework employing TCN-based encoders to decouple global and local trends in multivariate workload series. Bi et al. (2021) propose a bi-directional and grid LSTM-based deep learning method for accurate cloud workload and resource prediction using Google cluster traces [15]. Li et al. (2024) introduce EvoGWP, a spatio-temporal GNN framework that predicts long-term dynamic changes in workloads using shapelet extraction from Alibaba, Tencent, and Google datasets [90]. Bi et al. [15] propose an integrated method using bidirectional and grid LSTMs for high-quality prediction of cloud workloads. Dogani et al. [39] combine discrete wavelet transformation with bidirectional GRU networks to predict host load and improve resource utilization. Bi et al. [16] develop VAMBiG, a holistic model integrating variational mode decomposition, adaptive Savitzky-Golay filtering, multi-head attention, and bidirectional/grid LSTMs to capture nonlinear and multidimensional features for large-scale cloud traces. Tang et al. [146] use wavelet neural networks in conjunction with linear regression to forecast short-term workloads in cloud data centers, allowing energy-efficient scheduling. Similarly, Shariffdeen et al. [138] include machine learning-based ensemble models as part of their proactive auto-scaling mechanism to handle diverse workload

patterns.

Holistic Forecasting Frameworks: Holistic forecasting frameworks provide comprehensive end-to-end systems for scalable workload prediction, exemplified by Smendowski and Nawrocki's[143] Multi-Series Forecasting System (MSFS) that incorporates dynamic series selection, adaptive model choice, and multi-layer forecasters for cloud resource optimization, and Gao et al.'s clustering-enhanced framework that assigns optimized predictors per workload cluster to improve both scalability and interpretability while maintaining dependency on cluster quality and new task assignment complexity. . Li et al. [90] propose EvoGWP, a spatio-temporal graph-evolution model that identifies fine-grained workload usage patterns to predict long-term dynamic changes in Alibaba, Google, and Tencent traces. Kim et al. [79] introduce CloudInsight, an ensemble framework leveraging multiple predictors with runtime weighting to handle irregular and dynamically changing workloads. Yang et al. [165] combine autoencoders with echo state networks to improve multi-step-ahead host load prediction in Google cluster traces, while Shen et al. [139] highlight elastic resource scaling strategies for multi-tenant cloud systems to improve overall resource management. Sahni and Vidyarthi [131] address heterogeneity and cost-effectiveness in auto-scaling, while Guo et al. [54] dynamically classify workloads to self-adapt prediction methods for non-stationary cloud services. Qazi et al. [122] and Gandhi et al. [46] highlight frameworks that leverage historical traces and dynamic provisioning to optimize overall resource usage, SLA adherence, and energy efficiency in cloud systems.

This taxonomy offers a conceptual framework for organizing prior research and guiding model selection for real-world cloud CPU workload forecasting.

Table 2.1: Summary of evaluation metrics used for forecasting accuracy with comments.

Metric	Formula and Comments
Mean Absolute Error (MAE) [161] *Comments:* Simple and interpretable; less sensitive to outliers.	$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n}\lvert y_t - \hat{y}_t\rvert$
Huber Loss [67] *Comments:* Combines advantages of MAE and MSE; robust to outliers.	$L_\delta(y_t, \hat{y}_t) = \begin{cases} \frac{1}{2}(y_t - \hat{y}_t)^2, & \lvert y_t - \hat{y}_t\rvert \le \delta \\ \delta(\lvert y_t - \hat{y}_t\rvert - \frac{\delta}{2}), & \text{otherwise} \end{cases}$
Mean Squared Error (MSE) [28] *Comments:* Penalizes large errors more; widely used in regression.	$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2$
Root Mean Squared Error (RMSE) [28] *Comments:* Same scale as original data; emphasizes larger deviations.	$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}$
Normalized MSE (NMSE) [70] *Comments:* Scale-independent; useful for comparing across datasets.	$\text{NMSE} = \frac{\text{MSE}}{\sigma_y^2}$
Mean Absolute Percentage Error (MAPE) [105] *Comments:* Expressed in percentage; problematic when $y_t = 0$.	$\text{MAPE} = \frac{100}{n}\sum_{t=1}^{n}\left\lvert\frac{y_t - \hat{y}_t}{y_t}\right\rvert$
Symmetric MAPE (sMAPE) [105] *Comments:* Addresses division-by-zero issue in MAPE.	$\text{sMAPE} = \frac{100}{n}\sum_{t=1}^{n}\frac{\lvert y_t - \hat{y}_t\rvert}{(\lvert y_t\rvert + \lvert\hat{y}_t\rvert)/2}$
Mean Absolute Scaled Error (MASE) [70] *Comments:* Scale-free; useful for comparing different series.	$\text{MASE} = \frac{\frac{1}{n}\sum_{t=1}^{n}\lvert y_t - \hat{y}_t\rvert}{\frac{1}{n-1}\sum_{t=2}^{n}\lvert y_t - y_{t-1}\rvert}$
Weighted Absolute Percentage Error (WAPE) [52] *Comments:* Easy to interpret; less biased than MAPE for large datasets.	$\text{WAPE} = \frac{\sum_{t=1}^{n}\lvert y_t - \hat{y}_t\rvert}{\sum_{t=1}^{n}\lvert y_t\rvert} \times 100$

Table 2.2: Summary of Public Datasets for Cloud Workload Prediction.

Name	Year	Time Span	Size	Machines	Architecture	Workload Type
Google Cluster v1	2009	7 Hours	30MB	1	Google Borg	Real Workloads
Google Cluster v2	2011	29 Days	41GB	12,500	Google Borg	Real Workloads
Google Cluster v3	2019	1 Month	2.4TB	96,000	8 Google Clusters	Real Workloads
Alibaba Trace V1	2017	12 Hours	232MB	1,300	Datacenter	Online + Batch
Alibaba Trace V2	2018	8 Days	98GB	4,000	Datacenter	Online + Batch
Business Critical Workloads	2015	1 Month	284MB	1,250	—	—
Azure Dataset V1	2017	30 Days	117GB	2 Million	Azure Datacenter	Real Workloads
Azure Dataset V2	2019	30 Days	235GB	2.6 Million	Azure Datacenter	Real Workloads
PlanetLab Dataset	2011	10 Days	5.2MB	500+ Nodes	—	—
Scout OSR Dataset	2018	1 Day	587MB	Single	Amazon EC2 Containers	ML and Statistical Tests
IBM Docker Registry Trace	2017	75 Days	208GB	Multiple AZs	IBM Datacenter	Real Workloads
Dionatrafk	2018	1 Month	321KB	—	—	Univariate CPU
Intel Netbatch Logs	2012	1 Month	2.53GB	40,000+ Nodes	Distributed DCs	Mostly Serial Jobs
OpenCloud Hadoop Workload	2010	20 Months	120 CSVs	64 Nodes	Hadoop OpenCloud	ML and Parallel Workloads
Alibaba Microservices Trace v1	2021	12 Hours	61.1GB	1,300+ Bare-metal Nodes	Uniform Resource Mgmt	Microservice Online Services
Alibaba Microservices Trace v2	2022	13 Days	2TB	40,000+ Bare-metal Nodes	Uniform Resource Mgmt	Microservice Online Services

Table 2.3: Existing survey and review papers relevant to cloud workload forecasting and related domains (descending year order).

Paper (Authors, Year)	Primary Focus	Model Categories	Key Contribution	Relevance	Gaps
Ding et al. (2024)	App workload prediction	Stat, ML, DL, Hybrid	Taxonomy linking prediction to apps	Maps forecasts to ops	Limited model detail; minimal Transformer coverage
Toopchinezhad & Ahmadi (2024)	ML/DL for AQM	RL, Supervised, Unsupervised	Modern taxonomy	Analogy to CPU prediction	Not CPU-specific; needs adaptation
Smendowski & Nawrocki (2024)	Hybrid MSFS	DL + Hybrid (GRU)	FinOps-driven prediction	Example of DL-hybrid integration	Single-case; lacks broad comparison
Wei et al. (2024)	LLMs for model selection	LLaMA, GPT, Prophet, DeepAR	Automates model choice	Shows future of auto selection	Focus on selection, not design
Lim & Zohren (2021)	DL for time series	RNNs, LSTMs, Transformers	Theoretical foundation	Strengthens DL survey	Lacks cloud deployment
Torres et al. (2021)	DL architectures	CNNs, RNNs, attention	Broad DL review	Complements Lim & Zohren	Not cloud-focused; lacks practical context
Masdari & Khoshnevis (2020)	Cloud forecasting	Stat, ML, DL, Hybrid	Comprehensive taxonomy	Core reference	Misses recent DL/hybrid
Parmezan et al. (2019)	Model comparison	ARIMA, ETS, SVM, MLP, kNN	Benchmarks 11 models	Empirical baseline	Not cloud-specific; no modern DL
Amiri & Mohammad-Khanli (2017)	Provisioning prediction	ARIMA, Regression, ANN, Fuzzy	Links early	Shows method evolution	Pre-Transformer/DL
Lorido-Botran et al. (2014)	Auto-scaling	ARIMA, regression, early NNs	Proactive vs reactive	Forecasting for scaling	Outdated methods
Manvi & Shyam (2014)	IaaS resource mgmt	Stat, ML	Links forecasting to tasks	Expands ops context	Lacks DL
Islam et al. (2012)	Model comparison	Linear Regression, NN	Early ML vs simple models	Historical baseline	Lacks advanced ML/DL

Table 2.4: Datasets used across different forecasting methods.

Dataset	Statistical	ML	DL	Hybrid
Google	[10] [37]	[47]	[31], [123], [136], [168] [144] [55]	[31], [47], [177] [18] [115] [97] [17] [12] [176] [15] [165] [100] [138]
Alibaba	–	–	[39] [16] [90]	
Public	[66]	[64]	–	[66], [92]
Bitbrains	–	–	–	[77]
Energy / Traffic / AQ	–	–	–	[57], [135]
Wiki	[21] [81]	–	–	[135] [112]
PeMSD7	–	–	–	[135]
Prod Cloud	–	–	–	[143] [114] [27]
Electricity	–	–	–	[135] [140]
Web Logs	–	–	[83]	[46]
Alibaba	–	[153]	[31], [181]	–
PlanetLab	–	–	[104], [123] [174]	–
Dinda	–	–	[181]	–
DUX	–	–	[31]	–
Mixed	–	–	–	–
Synthetic	[8] [121] [98] [75] [118] [6]	[103][118], [143], [177]	[180]	[131] [146]
CPU Logs	–	[111]	–	–
AQM	–	[151]	–	–
Real	[164] [78] [91] [80]	[41] [173] [74] [149]	[156] [145] [38] [147] [73]	[126] [56] [142] [79] [122] [24] [141] [54]
FB	[148]	–	–	–
VM	[78]	–	–	–
NASA	–	–	[84]	-[30]
TPC-W	–	[116]	–	–
Hadoop	–	–	–	[139]
General	[58], [125]	–	–	–

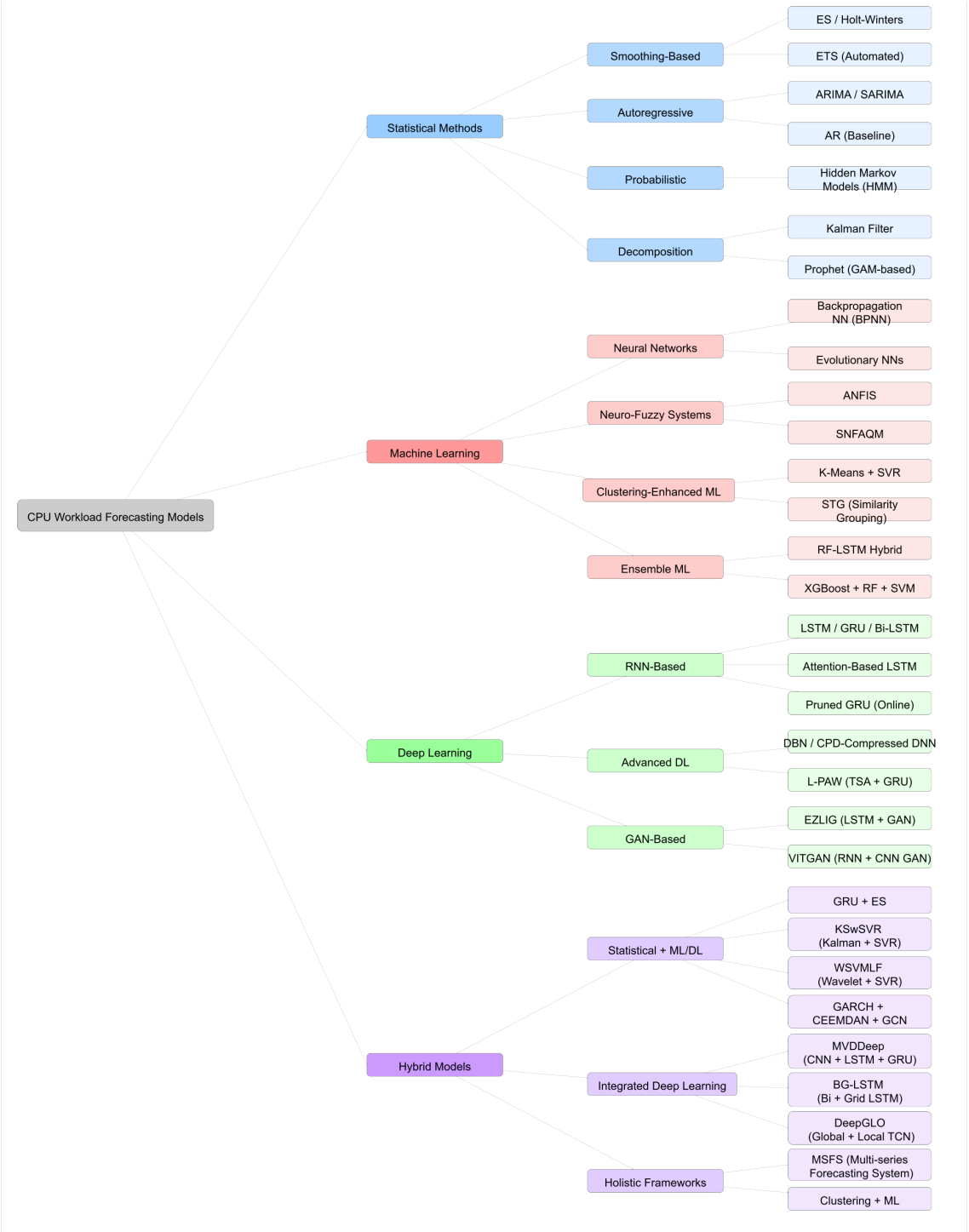


Figure 2.4: Taxonomy of CPU Workload Prediction Models.

Chapter 3

Proposed Methodology

3.1 Overview of the Methodology

The model is a hybrid deep learning architecture designed to forecast cloud workload by combining time and frequency domain information. First, a Fourier Transform extracts frequency features, which are enhanced using a Squeeze-and-Excitation (SE) block and concatenated with the original time-domain features. The combined features are framed into overlapping sequences and permuted to restructure the dimensions. A CNN layer captures local patterns, followed by another SE block to refine important features. The output is flattened, permuted, and passed through stacked LSTM layers with dropout to model temporal dependencies. Post-LSTM, global average pooling is applied, and its output is broadcast-added to the LSTM features. Finally, the enhanced sequences go through additional LSTM layers before making the final prediction via a fully connected layer.

3.2 Model Architecture

The forecasting model is a hybrid deep learning architecture designed to capture both temporal dynamics and frequency-based patterns in the cloud workload data.

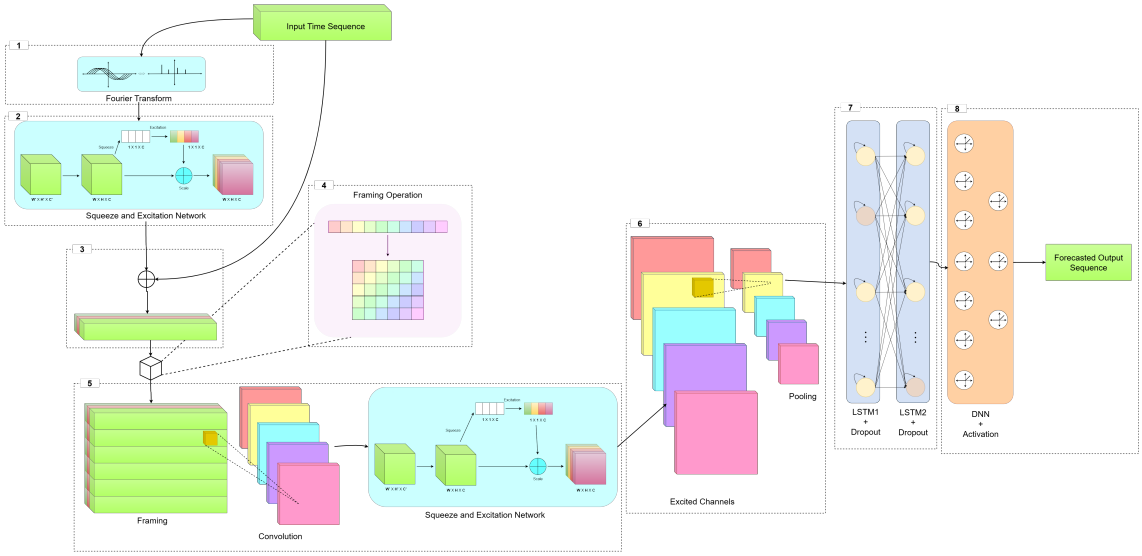


Figure 3.1: Proposed Architecture

3.3 Chronological Breakdown of Components

3.3.1 Fourier Transform (FFT)

The Fourier Transform of input data X is defined as:

$$X_{\text{freq}} = \mathcal{F}(X) = \text{FFT}(X) \tag{3.1}$$

where $\mathcal{F}$ denotes the Fourier Transform operator.

Discrete Fourier Transform (DFT)

The Discrete Fourier Transform (DFT) is given by:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i\frac{2\pi}{N}kn}, \quad k = 0, 1, \ldots, N-1 \tag{3.2}$$

where:

- x_n are the time-domain input data points,
- X_k are the frequency-domain components,
- N is the total number of data points,
- k is the frequency index,
- $e^{-i\frac{2\pi}{N}kn}$ is the complex exponential.

3.3.2 SENet Layer

The Squeeze-and-Excitation Network (SENet) applies a self-attention mechanism to emphasize informative features and suppress less useful ones.

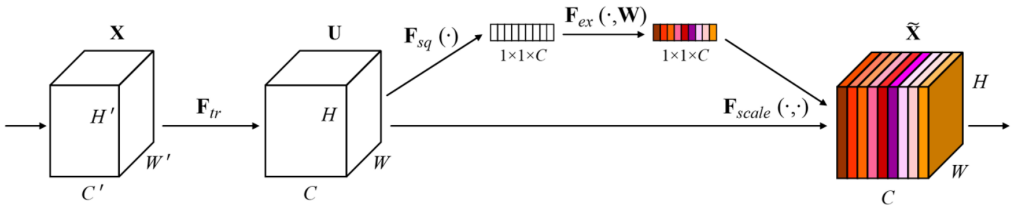


Figure 3.2: Squeeze and Excitation Network

Squeeze Operation

The input feature map X of shape (B, C, H, W), where B is the batch size, c is the number of channels, and H and W are the height and width of the feature map, is squeezed along the spatial dimensions H and W by global average pooling:

$$z_c = \frac{1}{H \times W} \sum_{h=1}^{H} \sum_{w=1}^{W} X_{c,h,w} \tag{3.3}$$

Where z_c is a scalar that aggregates the spatial information for channel c,and the operation aggregates the spatial information across the entire feature map into a single scalar per channel.

Excitation Operation

The squeezed features z are passed through two fully connected layers, creating a channel-wise attention mechanism:

$$s = \sigma(W_2 \cdot \delta(W_1 \cdot z)) \tag{3.4}$$

Where:

- $z \in \mathbb{R}^C$ is the vector of squeezed features,
- $W_1 \in \mathbb{R}^{\frac{C}{r} \times C}$ is the weight matrix for the first fully connected layer (with reduction ratio r),
- $W_2 \in \mathbb{R}^{C \times \frac{C}{r}}$ is the weight matrix for the second fully connected layer,
- $\delta(\cdot)$ is the ReLU activation function, and
- $\sigma(\cdot)$ is the sigmoid activation function.

Scaling Operation

The excitation vector s is used to scale the original input feature map X by channel-wise multiplication:

$$\hat{X}_c = s_c \cdot X_c, \quad \text{for each } c = 1, 2, \ldots, C$$

Where $\hat{X}_c$ is the recalibrated feature map for channel c, and s_c is the scaling factor for that channel.

Thus, the output of the SENet layer is the recalibrated feature map $\hat{X}$, which has enhanced important features and suppressed less relevant ones.

3.3.3 Concatenation of the Original and Frequency Domain Representations

We concatenate the frequency domain features with the time domain features.

Given the input tensor X in the time domain and its corresponding frequency domain representation X_{freq}, the operation is defined as:

$$X_{\text{combined}} = \text{concat}(X, X_{\text{freq}}, \text{dim} = -1) \tag{3.5}$$

Where: - $X \in \mathbb{R}^{N \times T \times F}$ is the original input tensor, with N being the batch size, T the number of time steps, and F the number of features. - $X_{\text{freq}} \in \mathbb{R}^{N \times T \times F_{\text{freq}}}$ is the tensor in the frequency domain, where F_{freq} is the number of frequency features.

The concatenation is performed along the last dimension, which combines the time-domain features and frequency-domain features for each time step:

$$X_{\text{combined}} \in \mathbb{R}^{N \times T \times (F + F_{\text{freq}})} \tag{3.6}$$

This concatenated tensor X_{combined} can now be passed to subsequent layers for further processing, enabling the model to leverage both temporal and frequency-based patterns in the data.

3.3.4 Framing Layer with Permutation

We focus on the operation where the input data is passed through a framing layer followed by a permutation of its dimensions.

The operation consists of two parts:

Framing Layer

The input tensor X is first passed through the framing layer. The framing operation is designed to slide a window of fixed size across the time dimension, creating sub-sequences or "frames" of the original sequence.

Let $X \in \mathbb{R}^{B \times T \times F}$, where:

- B is the batch size,
- T is the number of time steps,
- F is the number of features.

The framing operation generates a tensor of shape $\mathbb{R}^{B \times N \times L \times F}$, where:

- B is the batch size,
- N is the number of frames (windows),
- L is the length of each frame,
- F is the number of features.

This can be expressed as:

$$X_{\text{framed}} = \text{Frame}(X) \in \mathbb{R}^{B \times N \times L \times F} \tag{3.7}$$

Permutation

After framing, the resulting tensor is permuted to change its shape by reordering its dimensions. The permute operation swaps the frame dimension with the feature dimension, resulting in a tensor of shape $\mathbb{R}^{N \times F \times \text{num\_frames} \times \text{frame\_length}}$.

The permuted tensor X_{perm} is given by:

$$X_{\text{framed}} = \text{Permute}(X) \in \mathbb{R}^{B \times N \times F \times L} \tag{3.8}$$

3.3.5 Convolutional Layer (CNN)

The input tensor X is passed through a Convolutional Neural Network (CNN) layer. The CNN operation is used to apply a series of filters (kernels) to extract hierarchical features from the input data, where each filter focuses on local patterns and interactions between adjacent data points.

Let the CNN operation be defined as:

$$X_{\text{cnn}} = \text{CNN}(X) \tag{3.9}$$

where $X \in \mathbb{R}^{B \times N \times F \times L}$ and the output $X_{\text{cnn}} \in \mathbb{R}^{B \times N' \times F' \times L'}$, where:

- B is the batch size,
- N' is the number of frames after convolution,
- F' is the number of features after convolution,
- L' is the length of the frame after convolution.

The CNN layer applies a series of convolution operations across the input tensor, which results in new feature representations at different levels of abstraction. The transformation is parameterized by learned weights that evolve during training.

3.3.6 Squeeze-and-Excitation (SENet) Layer

The SENet layer that has been described earlier is applied yet again here. The input tensor X undergoes the squeeze and excitation operations to recalibrate the feature map by applying learned attention mechanisms. This process improves the model's ability to focus on important features and suppress irrelevant ones.

$$X_{\text{senet}} = \text{SENet}(X) \tag{3.10}$$

3.3.7 Flatten and Permutation

After the CNN and SENet layers, the resulting tensor is flattened and permuted to match the required input shape for further processing.

1. **Flattening:** The tensor X_{senet} is flattened along the frame and feature dimensions, reducing the 4D tensor into a 3D tensor.

$$X_{\text{flattened}} = \text{flatten2DOnwards}(X_{\text{senet}}) \in \mathbb{R}^{B \times (N' \times F' \times L')} \tag{3.11}$$

2. **Permutation:** The flattened tensor is permuted to reorder its dimensions, moving the feature dimension to the second position.

$$X_{\text{final}} = X_{\text{flattened}}.\text{permute}(0, 2, 1) \in \mathbb{R}^{B \times (N' \times F' \times L') \times 1} \tag{3.12}$$

3.3.8 LSTM Layers with Dropout

The LSTM layers in this section are applied sequentially, with dropout regularization after each layer to prevent overfitting. The dropout is applied during training, randomly zeroing some of the activations to make the model more robust.

Let $X \in \mathbb{R}^{B \times T \times F}$ be the input tensor, where:

- B is the batch size,
- T is the sequence length (number of time steps),
- F is the number of features.

LSTM Operation: Each LSTM layer processes the sequence input, capturing temporal dependencies. The output tensor from each LSTM has the shape $\mathbb{R}^{B \times T \times H}$, where H is the hidden state size.

After passing through the series of LSTM layers, the input tensor undergoes transformations in its shape, with the final tensor X_{lstm} having shape $\mathbb{R}^{B \times T \times H_4}$, where H_4 is the hidden size of the last LSTM layer.

$$X_{\text{res}}, = \text{LSTM}_1(X) \quad \Rightarrow \quad X_{\text{res}} \in \mathbb{R}^{B \times T \times H_1} \tag{3.13}$$

$$X_{\text{res}} = \text{Dropout}(X_{\text{res}}) \tag{3.14}$$

$$X_{\text{res}}, = \text{LSTM}_2(X_{\text{res}}) \quad \Rightarrow \quad X_{\text{res}} \in \mathbb{R}^{B \times T \times H_2} \tag{3.15}$$

$$X_{\text{res}} = \text{Dropout}(X_{\text{res}}) \tag{3.16}$$

$$X_{\text{res}}, = \text{LSTM}_3(X_{\text{res}}) \quad \Rightarrow \quad X_{\text{res}} \in \mathbb{R}^{B \times T \times H_3} \tag{3.17}$$

$$X_{\text{res}} = \text{Dropout}(X_{\text{res}}) \tag{3.18}$$

$$X_{\text{lstm}}, = \text{LSTM}_4(X_{\text{res}}) \quad \Rightarrow \quad X_{\text{lstm}} \in \mathbb{R}^{B \times T \times H_4} \tag{3.19}$$

The final tensor X_{lstm} is the output after all LSTM layers and dropout regularization. Dropout is a technique used during training to randomly set a fraction of input units to zero at each update, helping to prevent overfitting and improve generalization.

3.3.9 Integration and Interconnections

This section covers the final transformations applied to the data after the LSTM layers, which include a combination of global average pooling, broadcasting, and additional

LSTM layers. The output is then passed through a fully connected neural network (DNN) for the final prediction.

1) Frequency-domain extraction

Cloud workloads often contain strong cyclical patterns (e.g., diurnal or weekly usage cycles) that are more easily identified in the frequency domain [23]. The raw input time series X is transformed into the frequency domain using the Fast Fourier Transform (FFT) algorithm [42] which helps to isolate these global periodicities and can mitigate the effects of random noise. We take the absolute magnitude of the 2D FFT applied across the time and feature dimensions to obtain a stable spectral representation:

$$X_{\text{freq}} = |\mathcal{F}(X)| \tag{3.20}$$

where $\mathcal{F}$ denotes the 2D Discrete Fourier Transform, computed for each batch element as:

$$X_{k,l} = \sum_{t=0}^{T-1} \sum_{j=0}^{d-1} x_{t,j} \cdot e^{-i2\pi\left(\frac{kt}{T} + \frac{lj}{d}\right)} \tag{3.21}$$

for frequency indices $k = 0, \ldots, T-1$ and $l = 0, \ldots, d-1$. This spectral representation, illustrated in Figure-3.3 [50], highlights cyclical patterns essential for modeling cloud workloads.

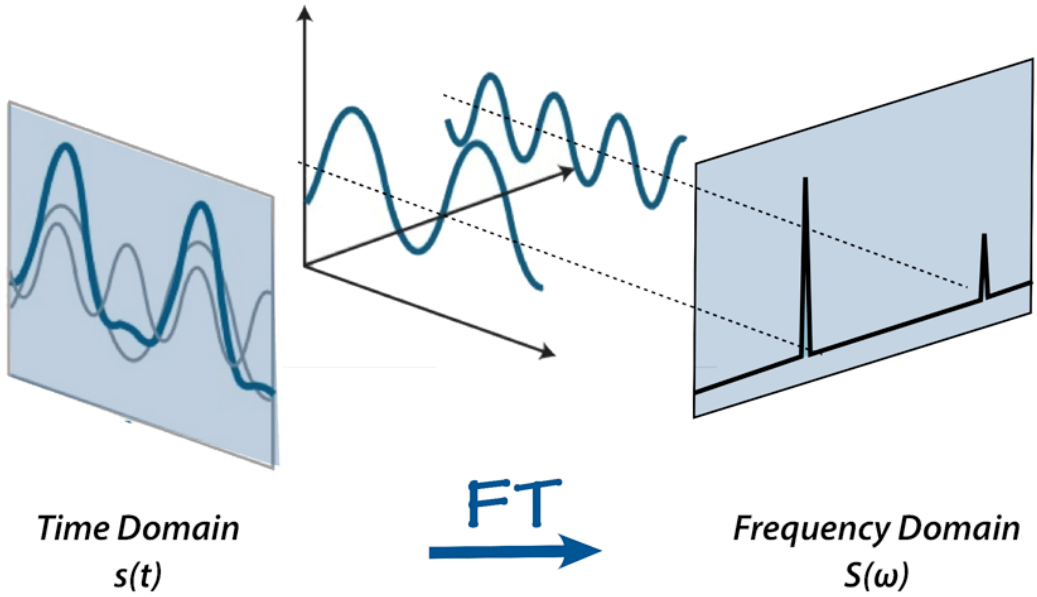


Figure 3.3: Conversion from time domain to frequency domain

2) Adaptive Spectral Feature Refinement

The raw frequency features may include redundant or uninformative channels. An attention mechanism is needed to dynamically select the most relevant spectral components for the forecasting task. To adaptively emphasize these components, a Squeeze-and-Excitation Network (SENet)[63] is applied. The process involves three steps:

i. **Squeeze:** Global information from each channel is aggregated using global average pooling across the time dimension. For each channel c, we compute a statistic z:

$$z = \frac{1}{T}\sum_{t=1}^{T} X_{\text{freq},t} \tag{3.22}$$

This equation calculates a channel descriptor by averaging the feature values $X_{freq,t}$ across the entire time dimension T. This effectively "squeezes" the temporal information for each channel into a single scalar value, representing that channel's global summary.

ii. **Excitation:** The aggregated information is fed through two fully-connected layers to learn channel-wise relationships and produce a vector of attention weights s:

$$s = \sigma(W_2 \cdot \delta(W_1 \cdot z)) \tag{3.23}$$

where W_1 and W_2 are weight matrices, δ is the ReLU activation function, and σ is the sigmoid activation function.

iii. **Recalibration:** The original frequency features are re-weighted using the learned attention scores:

$$\hat{X}_{\text{freq}} = s \cdot X_{\text{freq}} \tag{3.24}$$

The SENet adaptively re-weights each frequency channel based on its global importance, effectively amplifying significant cyclical patterns and suppressing noise. The output is a refined feature map, $\hat{X}_{\text{freq}}$.

3) Hybrid Time-Frequency Fusion

To create a comprehensive input representation, we must combine the global periodic information from the frequency domain with the precise, local event information from the time domain.

The refined frequency-domain features, $\hat{X}_{\text{freq},c}$, are concatenated with the original time-

domain input X, along the feature dimension to creat a unified tensor,

$$X_{\text{combined}} = \text{concat}(X, \hat{X}_{\text{freq}}, \text{dim} = -1) \in \mathbb{R}^{B \times T \times (2d)} \tag{3.25}$$

This dual-domain representation enables the model to integrate abrupt changes (e.g., load spikes) with underlying cycles.

4) Local Context Capture via Framing

Convolutional Neural Networks (CNNs) are highly effective in extracting spatial patterns from grid-like data. To apply CNNs on input time series, it must be converted into a format that captures local sequential contexts, similar to how an image is composed of overlapping patches. For that the combined tensor is transformed using a sliding window operation (*framing*) with frame length $L = 30$ and step size 1:

$$X_{\text{framed}} = \text{Unfold}(X_{\text{combined}}, \text{dim} = 1, \text{size} = 30, \text{step} = 1) \in \mathbb{R}^{B \times N_w \times 30 \times 2F} \tag{3.26}$$

where $N_w = T - 29$ is the number of windows. The tensor is then permuted to image-like format for convolutional operations. This framing, depicted in Figure-3.4, transforms the $1D$ time series into a set of overlapping $2D$ patches, preserving short-term dependencies and making it suitable for processing by 2D convolution layers.

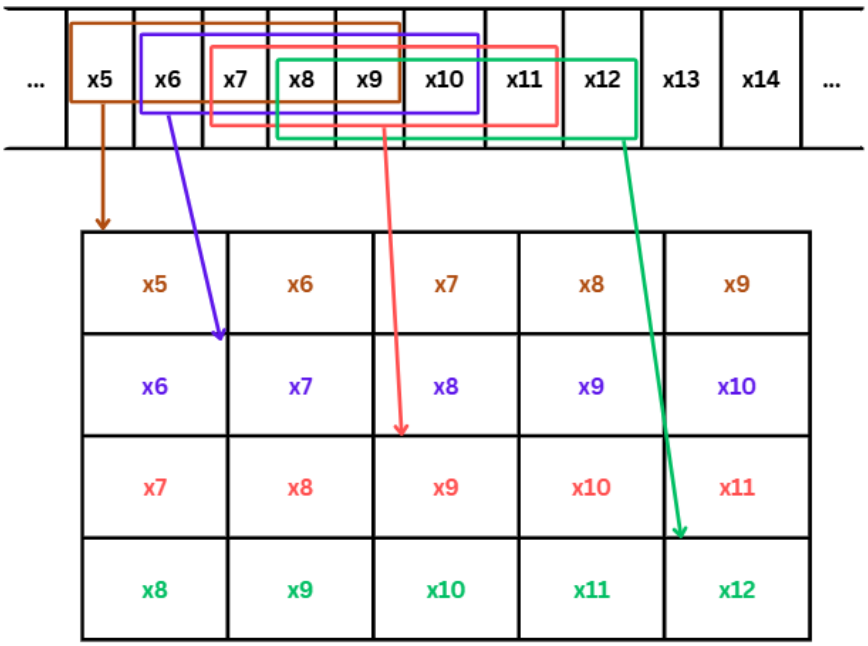


Figure 3.4: Framing layer working process [figure-**??**(4)]

5) Spatio-Temporal Feature Extraction:

A convolutional neural network (CNN) is applied to the framed input in order to capture localized short-term patterns within each window. The extracted representations are further refined by an attention mechanism. Specifically, the CNN module consists of two layers: $X_{\text{cnn}} = \text{CNN}(X_{\text{framed}})$; The first convolutional layer employs 64 output channels and 3×3 kernels, followed by ReLU and batch normalization; the second layer uses 128 output channels and 1×1 kernels, again followed by ReLU activation and batch normalization.. Subsequently, a second SENet block is introduced to enhance the resulting feature maps (X_{cnn}) through channel-wise attention:

$$X_{\text{refined}} = \text{SENet}(X_{\text{cnn}}) \tag{3.27}$$

This step adaptively re-weights the feature channels, emphasizing the most informative spatio-temporal patterns for downstream processing.

6) Feature Aggregation and Dimensionality Reduction:

To prepare the features for the final modeling stage while ensuring computational efficiency, the high-dimensional feature maps are condensed into a more compact representation. This is accomplished through Adaptive Global Average Pooling (GAP)[124], which reduces each spatial feature map to a single representative value. Specifically, for every channel in the refined feature maps $X_{refined}$, GAP computes the mean across all spatial dimensions, making a concise yet information-preserving representation suitable for subsequent processing. This operation can be formally expressed for each batch element b and channel c as:

$$X_{\text{pool}} = \frac{1}{H \times W} \sum_{h=0}^{H-1} \sum_{w=0}^{W-1} X_{\text{refined},b,c,h,w} \in \mathbb{R}^{B\times 128\times 1\times 1} \tag{3.28}$$

where H and W denote the height and width of the input feature map's spatial dimensions (in this architecture, $H = N_w$ and $W = 30$). This process reduces the tensor $\mathbb{X}_{\text{refined}} \in \mathbb{R}^{B\times 128\times N_w\times 30}$ to $X_{\text{pool}} \in \mathbb{R}^{B\times 128\times 1\times 1}$. The pooled tensor is then reshaped as $X_{\text{seq}} = \text{View}(X_{\text{pool}}, (B, 1, 128))$, forming a feature vector suitable for recurrent processing. By aggregating spatio-temporal information across the entire look-back window into a single fixed-size representation, this pooling step not only improves computational efficiency but also helps mitigate overfitting.

7) Long-Range Sequential Modeling:

The CNN module captures localized short-term patterns; however, modeling long-range temporal dependencies and capturing the overall sequence dynamics requires a recurrent

neural network. To achieve this, the aggregated feature sequence is processed by a stack of two Long Short-Term Memory (LSTM) layers [53], with dropout applied between layers for regularization:

$$X_{\text{lstm1}}, \_ = \text{LSTM}_1(X_{\text{seq}}) \quad \text{(input: 128, hidden: 64)}, \tag{3.29}$$

$$X_{\text{lstm1}} = \text{Dropout}(X_{\text{lstm1}}, p = 0.2), \tag{3.30}$$

$$X_{\text{lstm2}}, \_ = \text{LSTM}_2(X_{\text{lstm1}}) \quad \text{(input: 64, hidden: 32)}, \tag{3.31}$$

$$X_{\text{lstm2}} = \text{Dropout}(X_{\text{lstm2}}, p = 0.2). \tag{3.32}$$

LSTMs are well-suited for processing sequential information, and stacking them allows for learning more complex feature hierarchies.

8) Final Prediction:

Finally, the learned high-level feature representations are fed into a small dense neural network (DNN) to map to the desired multi-step forecast horizon:

$$\hat{y} = \text{DNN}(X_{\text{lstm2}}[:, -1, :]) \tag{3.33}$$

This acts as the prediction head, comprises a linear layer ($32 \rightarrow 32$), ReLU activation, and another linear layer ($32 \rightarrow H$), with H being the forecast horizon. This network outputs a vector of length H, corresponding to the CPU usage forecast.

Chapter 4

Experimental Setup

4.1 Datasets

To ensure a robust and comprehensive evaluation, our proposed model was tested on two large-scale, real-world datasets that represent diverse cloud computing workloads: **Alibaba Cluster Trace (2018)** [3] and **vmCloud Dataset** [43]. The selection criteria focused on datasets with high-resolution, multivariate time series data that exhibit the complex periodic and non-stationary patterns characteristic of production environments.

Alibaba Cluster Trace is a public, large-scale datacenter workload trace that provides insight into production system behavior. We utilized *Version 2*, which contains 8 days of monitoring data from more than 4,000 machines. Its richness, including fine-grained resource metrics and metadata for containers and batch jobs, makes it ideal for testing model scalability and its ability to handle complex high-dimensional data. For this experiments, a feature set encompassing temporal context (e.g., `hour`, `weekday`), core resource utilization (`cpu_util_percent`, `mem_util_percent`), network I/O (`net_in`, `net_out`), and disk activity (`disk_io_percent`) were selected.

vmCloud Dataset provides a multivariate time series of cloud computing performance metrics, with approximately 2 million observations capturing virtual machine (VM) activity. Core resource features include CPU usage, memory utilization, network traffic, execution time, number of executed instructions, power consumption, and derived measures of energy efficiency. Each record is time-stamped and linked to a unique VM identifier, enabling temporal analysis and per VM tracking. In addition, the dataset includes task metadata (e.g., `task type`, `priority`, and `status`), which supports evaluating models across heterogeneous workload types. The explicit inclusion of power and efficiency metrics makes this dataset particularly valuable for studies focused on energy-aware cloud

management and sustainable computing.

4.2 Data Preprocessing

A multi-stage preprocessing pipeline was applied to clean the data, handle missing values, and engineer relevant features to improve model performance and training stability. While several steps were shared across datasets, others were tailored to the specific characteristics of the *Alibaba Cluster Trace* and *vmCloud* datasets.

For data cleaning and missing value handling, the Alibaba dataset required imputing missing entries in the `mem_gps` and `mkpi` columns with a constant placeholder to signify absence. In contrast, the vmCloud dataset contained irregular timestamps, which were forward-filled to ensure temporal continuity. The series was then sorted to maintain chronological order, and any remaining numerical gaps were filled using median imputation.

The temporal structuring of the datasets also differed. The Alibaba dataset was resampled to a fixed 30-minute interval to ensure a consistent time series resolution. The vmCloud dataset did not require resampling, but the forward-filling of missing timestamps served to maintain a regular sequence.

Feature engineering was particularly dataset-specific. For the Alibaba dataset, instances with a coefficient of variation greater than 0.1 were filtered to remove potential outliers or unstable periods. The vmCloud dataset, on the contrary, underwent extensive feature engineering. Time-based features were extracted from timestamps, including month, day, hour, minute, weekday, and weekend indicators, enabling the model to capture cyclical patterns. Additionally, domain-informed features were created, such as first-order differences of CPU usage to capture rate changes, rolling averages to smooth short-term fluctuations, lag features, and exponential moving averages. The trend and seasonality components were also explicitly decomposed to better characterize workload dynamics.

Finally, after these dataset-specific operations, both datasets underwent a normalization step. All numerical features were scaled to the range $[0, 1]$ using a `MinMaxScaler`. This step ensured that features with larger scales did not disproportionately influence the model and helped achieve faster convergence and greater stability during training.

4.3 Baseline and Comparative Models

To rigorously evaluate the performance of SpectraNet, we selected a diverse set of baseline models representing the state-of-the-art in time series forecasting. The baselines were chosen to benchmark our model against a range of architectural approaches, from established recurrent and convolutional networks to more recent hybrid designs. The models are summarized in Table-4.1 and are grouped into the following categories for our comparative analysis:

- **Recurrent and Convolutional Baselines:** This group includes foundational deep learning models such as a simple 1D CNN (`cnn_1d`) [88], an LSTM with dense layers (`lstm_dnn`) [60], [88], and a Temporal Convolutional Network (`tcn`) [87]. These models serve to establish a performance baseline using standard single-domain (time-only) architectures.
- **Attention-Enhanced and Hybrid Architectures:** To compare against more advanced models, we include architectures that incorporate attention mechanisms or hybrid components. This includes an LSTM with Squeeze-and-Excitation attention (`se_lstm_dnn`), a CNN-LSTM with SE attention (`t_cnn_se_lstm`), and Convolutional LSTM variants (`c_lstm`, `c_lstm_ae`) [60]. These models allow us to assess the performance contribution of our specific hybrid design against other fusion and attention strategies.
- **State-of-the-Art and Competing Hybrid Models:** This category includes recent, high-performing models designed for complex forecasting tasks. We compare against `mcdfn`, a multi-scale convolutional network [72], and `frets`, a hybrid frequency-temporal model [171]. These models represent a direct comparison against other sophisticated architectures that, like SpectraNet, aim to capture complex data patterns.

This comprehensive set of baselines allows us to situate **SpectraNet**'s performance in the broader landscape of time series forecasting and specifically to demonstrate its advantages in balancing accuracy and computational efficiency.

4.4 Training and Evaluation Protocol

A rigorous protocol was established for training and evaluating all models to ensure fair comparison and reproducible results.

Table 4.1: Baseline Models for Time Series Forecasting

Model Name	Type / Architecture
MCDFN [72]	Multi-scale convolutional deep forecasting network
SE LSTM DNN [60], [88]	LSTM with dense layers + attention
LSTM DNN [60], [88]	LSTM with dense layers
TCN [87]	Temporal Convolutional Network
FreTS [171]	Hybrid frequency-temporal sequence model
T CNN SE LSTM [60], [88]	CNN-LSTM with SE attention
C LSTM AE [60]	Convolutional LSTM with autoencoder
C LSTM [60]	Convolutional LSTM without autoencoder
CNN 1D [88]	Simple 1D CNN

Data Splitting and Validation To maintain temporal integrity and prevent look-ahead bias, which is critical for time series forecasting, the dataset was split chronologically into training (70%), validation (20%), and testing (10%) sets. All feature scaling models (e.g., MinMaxScaler) were fitted exclusively on the training data and then applied to the validation and test sets to prevent data leakage. The validation set was used for hyperparameter tuning and to identify the best-performing model epoch during training.

Model Training All models were trained using the Adam optimizer, chosen for its adaptive learning rate capabilities, with an initial learning rate of 1×10^{-3} and a weight decay of 1×10^{-5} for regularization. The Mean Squared Error (MSE) was used as the loss function, as its sensitivity to larger errors is highly desirable in CPU usage forecasting, where significant prediction misses can lead to severe resource provisioning issues. Training was conducted for a maximum of 100 epochs, with model checkpoints saved every two epochs to ensure reproducibility.

Regularization and Overfitting Prevention Several techniques were employed to promote generalization and prevent overfitting. **Early stopping** was implemented with a patience of 10 epochs; training was halted if no improvement in validation loss was observed over this period. This ensures the model with the best generalization performance is selected. Additionally, **dropout** with a rate of $p = 0.2$ was applied after each LSTM block to further regularize the model.

Evaluation The final evaluation was performed on the held-out test set using the model weights that achieved the lowest validation loss. To provide a comprehensive assessment of prediction accuracy, performance was measured using three metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE). We

evaluated performance across multiple forecast horizons ($H = 1, 5, 10, 30$) to understand the model's capabilities for both short-term and long-term prediction scenarios in cloud resource management.

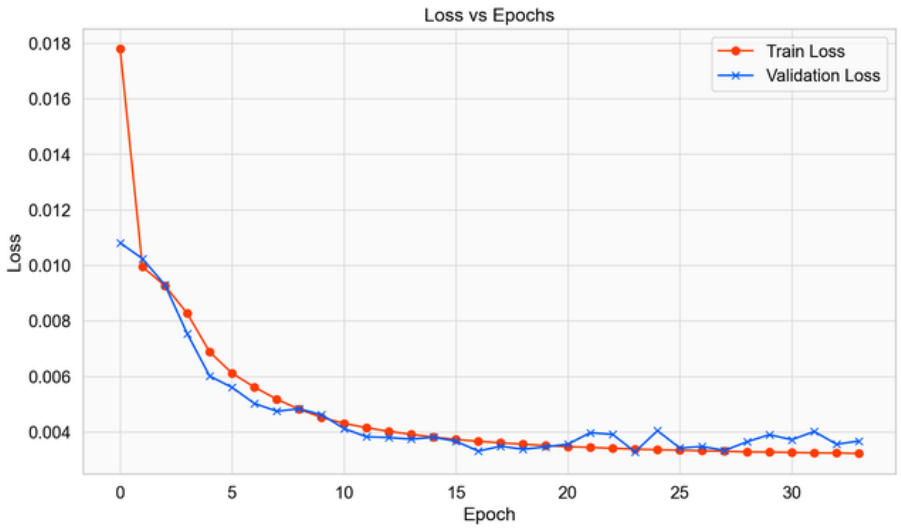


Figure 4.1: An example of the training and validation loss curves, demonstrating model convergence and the point at which early stopping might be triggered.

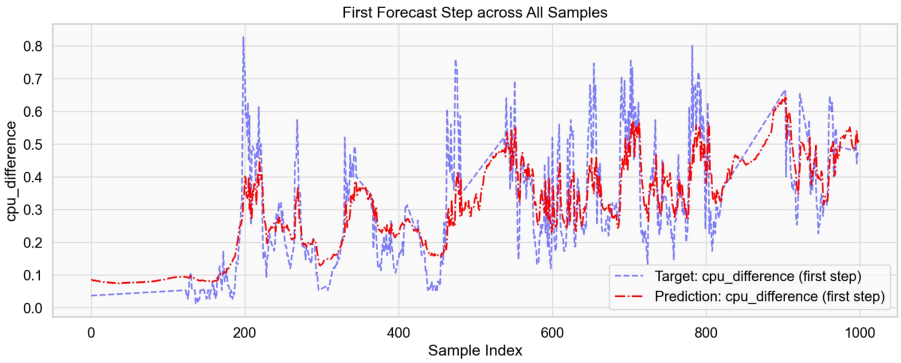


Figure 4.2: A sample prediction from the trained model versus the actual ground truth values for a forecast horizon of H=10.

4.5 Evaluation Metrics

The performance of the forecasting models was assessed using widely adopted error-based evaluation metrics, including *Mean Squared Error* (MSE), *Mean Absolute Error* (MAE), and *Root Mean Squared Error* (RMSE) [61]. These metrics quantify the deviation between the predicted values and the ground truth observations, providing complementary perspectives on prediction accuracy. Formally, the metrics are defined as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4.1}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{4.2}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{4.3}$$

Here, y_i denotes the actual value, $\hat{y}_i$ the predicted value, and n the total number of observations. The MSE metric emphasizes larger errors due to the squaring of differences, making it sensitive to outliers. MAE measures the average absolute deviation between predictions and actual values, treating all errors linearly. RMSE provides an interpretable error in the same units as the original data, combining sensitivity to larger errors with direct interpretability.

In practice, MAE is useful for capturing general prediction deviation, while MSE and RMSE are more appropriate when larger errors need to be penalized. RMSE is often preferred when reporting results in the original scale of the variable, facilitating intuitive interpretation. By combining these metrics, we ensure a comprehensive evaluation of forecasting accuracy, balancing sensitivity to outliers and overall prediction deviation [61].

4.6 Implementation and Environment

The model is implemented using PyTorch, providing flexibility and scalability for training deep learning models. The training process is carried out on an Nvidia GPU to accelerate computations, particularly when working with large datasets.

- **Software Setup:** The environment includes the following key dependencies:
  - PyTorch
  - NumPy, pandas, scikit-learn for data manipulation and preprocessing
  - Matplotlib and Seaborn for result visualization
  - Some other miscellaneous libraries
- **Operating System:** Ubuntu 20.04 was used for setting up the development environment.

4.7 Hyperparameters

The model training process uses the following hyperparameters:

- Learning rate: 0.001
- Batch size: 64
- Number of epochs: 100
- Patience for early stopping: 10
- Checkpoint interval: 2

4.8 Conclusion of the Chapter

In summary, this chapter presented a comprehensive overview of the datasets, preprocessing steps, training procedures, evaluation metrics, and implementation details used in this study. By leveraging the Azure and VMCloud datasets, robust preprocessing pipelines, and careful feature engineering, the groundwork was laid for effective CPU usage forecasting in cloud environments. The structured training approach, supported by checkpointing and early stopping, ensured optimal model generalization, while the use of diverse evaluation metrics provided a thorough assessment of model performance. Finally, the PyTorch-based implementation, accelerated by GPU computing, allowed efficient handling of large-scale data and complex model architectures. The following chapter will delve into the experimental results and insights drawn from the model evaluations.

Chapter 5

Results and Discussion

This section evaluates the performance of the proposed **SpectraNet** model against recent deep learning baselines on the **Alibaba Cluster Trace (2018)** [3] and **vmCloud** [43] dataset, focusing on cloud workload forecasting. Models are compared using Mean Absolute Error (MAE) and Mean Squared Error (MSE) for predictive accuracy across forecast horizons (t=1, 5, 10, 30). Additionally, model complexity (trainable parameters) and inference time are analyzed to assess computational efficiency, critical for real-time resource allocation. The subsequent sections present a comprehensive evaluation of the proposed approach through quantitative comparisons, visual analyses, and detailed error analysis to validate its effectiveness

5.1 Quantitative Comparison

Table 5.1 highlights the model complexity and average per-sample inference times across different forecast horizons. Ultra-lightweight models such as **LSTM DNN**[60], [88], containing only 26,366 parameters, achieve the fastest inference speeds, reaching 0.0044 ms at $t = 10$. However, this efficiency comes at the expense of predictive accuracy, with an MAE of 0.0664 at $t = 30$. In contrast, larger architectures, including **FreTS**[171] with 2.0 million parameters and **T CNN LSTM**[60], [88] with 343,578 parameters, exhibit substantially longer inference times ranging from 0.38 to 0.83 ms at $t = 30$, making them less suited for real-time deployment. **SpectraNet**, however, provides a balanced profile, delivering near-best accuracy at $t = 30$, leading performance at $t = 10$, and among the fastest inference speeds for longer horizons, making it a strong candidate for resource-constrained environments.

Table 5.2 presents a detailed performance breakdown across forecast horizons of $t = 1$,

$t = 5$, $t = 10$, and $t = 30$. **SpectraNet** demonstrates robust adaptability, with MAE decreasing from 0.1200 at $t = 1$ to 0.0447 at $t = 30$, and corresponding MSE values from 0.0200 to 0.0050. This trend indicates that the model's predictive capability strengthens over longer horizons, particularly where temporal and spectral fusion plays a significant role.

At $t = 10$, **SpectraNet** achieves the best overall results, with an MAE of 0.0549 and MSE of 0.0058, outperforming all baselines. At $t = 30$, it ranks second, with an MAE of 0.0447 and MSE of 0.0050, surpassed only by **MCDFN**[72], a model with 1.76 million parameters, in contrast to **SpectraNet**'s compact 89,805 parameters as reported in Table 5.1. These results demonstrate that even when not leading, **SpectraNet** remains highly competitive, achieving comparable accuracy while requiring a fraction of the computational resources of models more than 19 times its size.

Table 5.1: Model complexity and per-sample inference time comparison

Model	**Params**	**Inference Time (ms)**			
		t=1	**t=5**	**t=10**	**t=30**
SpectraNet	89,805	0.0279	0.0250	0.0148	**0.13**
MCDFN	1.76M	0.0113	0.0151	0.1300	0.68
FreTS	2.0M	0.0067	0.0095	0.0068	0.38
C LSTM	64,958	0.0070	0.0054	0.0077	0.77
C LSTM AE	257,152	0.0084	0.0069	0.0095	0.83
CNN 1D	207,155	0.0091	0.0074	0.0106	0.80
LSTM DNN	**26,366**	**0.0038**	**0.0027**	**0.0044**	0.79
T CNN LSTM	343,578	0.1000	0.1300	0.0245	0.68

Table 5.2: Model performance across forecast horizons

Model	**Forecast Horizon**							
	t=1		**t=5**		**t=10**		**t=30**	
	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE
SpectraNet	0.1200	0.0200	0.1500	0.0300	**0.0549**	**0.0058**	0.0447	0.0050
MCDFN	**0.0362**	0.0032	0.0542	**0.0054**	0.0584	0.0069	**0.0272**	**0.0020**
FreTS	0.0370	0.0031	0.0518	**0.0054**	0.0593	0.0070	0.0704	0.0095
C LSTM	0.0401	0.0034	0.0539	0.0057	0.0607	0.0067	0.0640	0.0078
C LSTM AE	0.0364	**0.0030**	**0.0495**	0.0055	0.0565	0.0064	0.0645	0.0076
CNN 1D	0.0430	0.0034	0.0705	0.0077	0.0586	0.0067	0.0704	0.0085
LSTM DNN	0.0376	0.0031	0.0508	0.0055	0.0597	0.0066	0.0664	0.0083
T CNN LSTM	0.1000	0.0100	0.1300	0.0250	0.0651	0.0082	0.0683	0.0088

5.2 Visual Analysis

Figure 5.1 visualizes the trade-off between model complexity and accuracy. The bar plot compares normalized MSE and parameter counts, showing **SpectraNet** and **LSTM DNN** as efficient, with moderate sizes and competitive errors. Larger models like **FreTS** (2.0M parameters) show diminishing returns, with marginal accuracy gains.

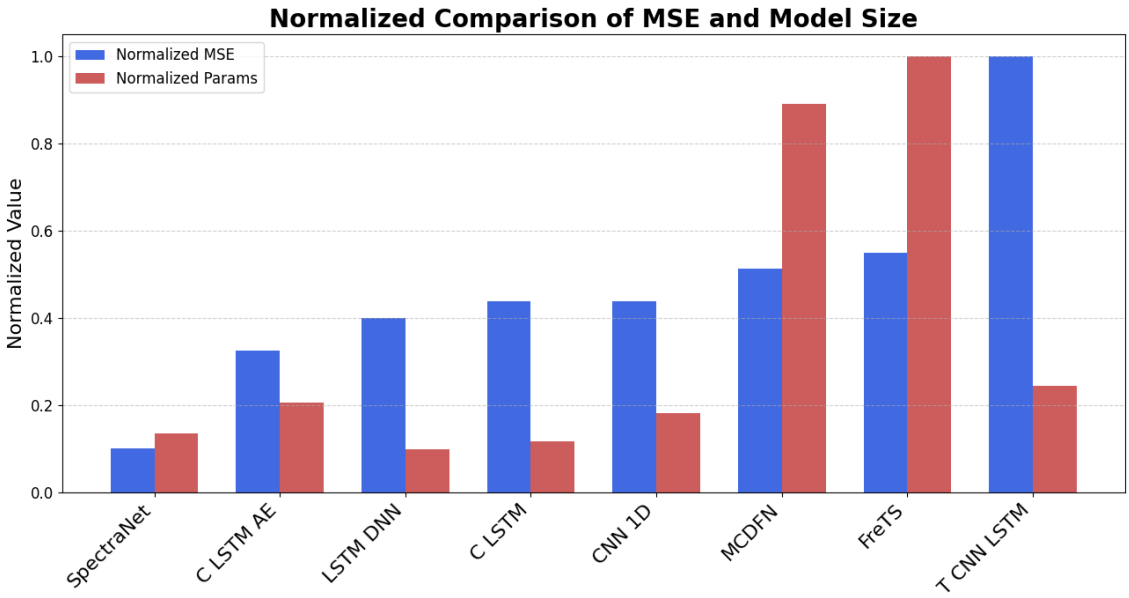


Figure 5.1: Relationship between model size and prediction accuracy (Test MSE Loss) on normalized scale

Figure 5.2, presents a bubble chart comparing the models across three dimensions: mean squared error (MSE), inference time, and model parameter size. The results highlight **SpectraNet** as a balanced solution, achieving low MSE and reduced inference time while maintaining a modest parameter size, making it well-suited for cloud workload forecasting.

- **X-axis (Inference Time)**: Models like **LSTM DNN** (0.0044 ms at t=10) and **SpectraNet** (0.13 ms at t=30) cluster on the left, indicating suitability for real-time applications.
- **Y-axis (Test MSE Loss)**: Lower values (e.g., **MCDFN**, **SpectraNet** at t=30) reflect higher accuracy.
- **Bubble Size (Parameters)**: Smaller bubbles (e.g., **LSTM DNN**, **SpectraNet**) denote leaner models with less number of trainable parameter, while larger bubbles (e.g., **FreTS**, **MCDFN**) indicate higher complexity.

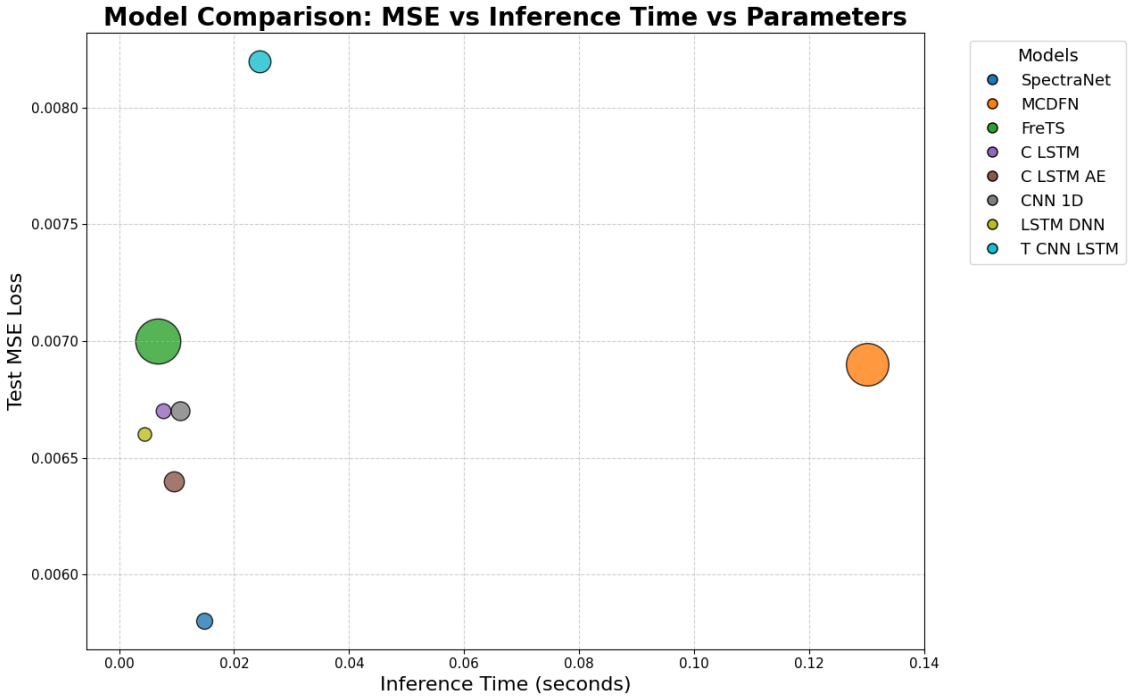


Figure 5.2: Trade-off between model accuracy, speed, and size. This bubble chart visualizes the relationship between normalized MSE loss and normalized inference time for various models. The size of each bubble corresponds to the model's parameter count, illustrating the classic trade-off between accuracy, speed, and model complexity.

5.3 Error Analysis

The primary evaluation metric, Mean Squared Error (MSE), highlights **SpectraNet**'s strength, particularly at longer forecast horizons. At $t = 10$ and $t = 30$, the model achieves MSE values of 0.0058 and 0.0050 respectively shown in Table 5.2, outperforming most baseline models, with the exception of **MCDFN** at $t = 30$. Notably, despite this slight gap at the longest horizon, **SpectraNet** maintains a significant advantage in computational efficiency: it is approximately 19 times lighter than **MCDFN**, with only 89,805 parameters compared to 1,760,000, and demonstrates around fivefold faster inference at $t = 30$, with per-sample inference time reaching 0.13 ms for **SpectraNet** and 0.68 ms for **MCDFN**, making it highly suitable for real-time deployment scenarios.

5.4 Discussion

Table 5.2 and Table 5.1 together highlight the performance and efficiency of different models across forecast horizons. **SpectraNet** demonstrates robust adaptability, with MAE decreasing from 0.1200 at $t = 1$ to 0.0447 at $t = 30$, and corresponding MSE values from 0.0200 to 0.0050, indicating that its predictive capability strengthens over longer horizons

where temporal and spectral fusion plays a critical role. At $t = 10$, it achieves the best overall results, with an MAE of 0.0549 and MSE of 0.0058, outperforming all baselines. At $t = 30$, it ranks second, with an MAE of 0.0447 and MSE of 0.0050, surpassed only by **MCDFN**[72], a model with 1.76 million parameters, in contrast to **SpectraNet**'s compact 89,805 parameters.

In terms of efficiency, ultra-light models such as **LSTM DNN**[60], [88] (26,366 parameters) achieve the fastest inference speeds (0.0044 ms at $t = 10$) but at the cost of a 48.55% reduction in accuracy (MAE: 0.0664 at $t = 30$) compared to **SpectraNet**(MAE: 0.0447 at $t = 30$). Larger architectures, including **FreTS**[171] with 2.0 million parameters and **T CNN LSTM**[60], [88] with 343,578 parameters, require substantially longer inference times (0.38–0.83 ms at $t = 30$), limiting their suitability for real-time applications. In contrast, **SpectraNet** maintains near-leading accuracy while delivering fast inference speeds across all horizons, offering a well-balanced profile that combines high predictive performance with computational efficiency, making it an optimal choice for resource-constrained environments.

The proposed SpectraNet model can be seamlessly integrated into modern cloud orchestration frameworks to enable proactive resource management. In a Kubernetes environment, for instance, CPU usage metrics can be collected via Prometheus, and our model can be exposed as a REST API using a connector such as PredictKube to provide near-real-time workload forecasts. These predictions can then inform an auto- scaler (e.g., the Horizontal Pod Autoscaler) to adjust resources dynamically. Such a pipeline can be implemented on local clusters (using Minikube) or multi-node setups (using kubeadm) for evaluation under realistic workloads generated by tools like stress-ng or auto-scalable web applications. Accurate forecasts directly benefit cloud providers and users by preventing both over-provisioning and under-provisioning. This leads to lower operational costs, reduced energy consumption, and consequently, a smaller carbon footprint. Moreover, proactive allocation minimizes cold-start delays, helping systems maintain compliance with Service Level Agreements (SLAs). These outcomes are well supported by prior studies that established quantitative relationships between predictive accuracy and improvements in cost efficiency, energy use, and sustainability metrics.

Chapter 6

Conclusion

The Conclusion chapter serves to reflect on the research conducted, summarizing the main insights gained, discussing their implications, acknowledging the limitations encountered, and proposing directions for future research. This chapter ties together the key aspects of the study, emphasizing its contributions to the field of cloud workload forecasting and lightweight deep learning models.

6.1 Restating Research Objectives and Questions

This research was primarily motivated by the need to develop a lightweight yet accurate model for CPU usage forecasting in cloud computing environments. The core research questions focused on: (i) how to effectively combine time domain and frequency domain information for better workload prediction, and (ii) how to design a model architecture that remains computationally efficient while delivering competitive forecasting performance. These objectives have guided every aspect of the research, from dataset preparation to model design and evaluation.

6.2 Summary of Key Findings

Through extensive experimentation on both the Azure VM CPU Readings dataset and the VMCloud dataset, the proposed lightweight hybrid model demonstrated strong predictive performance, effectively capturing temporal patterns and periodic trends. Key findings include:

- Incorporating frequency domain features significantly improved forecasting accuracy compared to time-domain-only approaches.

- The lightweight architecture, utilizing components like FFT, CNN, SENet, and LSTM layers, achieved a good balance between model complexity and prediction performance.
- Evaluation metrics such as MAE, RMSE, MSE, and Huber Loss consistently indicated that the model maintained robustness across different workloads and scenarios

6.3 Discussion of Implications

The findings of this study have several important implications. First, they demonstrate that integrating frequency domain representations in cloud workload forecasting is not only feasible but also beneficial for improving predictive performance without introducing excessive computational overhead. Second, the study highlights that lightweight architectures can still achieve competitive results, supporting more sustainable and scalable solutions for real-world cloud resource management. These insights contribute to the growing literature advocating for efficient models that are suitable for deployment in production cloud systems where resources are limited.

6.4 Contributions of the Research

The contributions of this research are fourfold:

- A novel lightweight hybrid deep learning architecture that fuses time and frequency domain features for time series forecasting.
- An empirical demonstration of the benefits of frequency domain analysis in cloud workload prediction.
- A comprehensive preprocessing pipeline and experimental framework applicable to multivariate cloud datasets.
- Practical insights into model design choices that balance accuracy, robustness, and efficiency for resource-constrained environments.

6.5 Acknowledging Limitations

Despite the promising results, several limitations must be acknowledged. The datasets used, although diverse, may not fully represent all types of cloud workload behaviors encountered in large-scale commercial environments. Furthermore, while the model achieves a balance between accuracy and efficiency, there is still room for optimization in

terms of training time and adaptability to sudden workload shifts. The experiments also focused primarily on CPU usage, with other resource types such as memory and network I/O receiving less attention.

While this study focuses on CPU usage to validate the core concept of parallel time–frequency forecasting, the proposed framework is resource agnostic and can be extended to joint prediction of CPU, memory, I/O, and network metrics. Future work will explore multi-resource forecasting and integration with full-stack cloud management systems.

6.6 Suggestions for Future Research

Future work can expand on this research in several directions. Incorporating uncertainty estimation into predictions could make the model even more reliable for real-world deployment. Exploring online learning methods would allow the model to adapt dynamically to changing workload patterns. Additionally, extending the hybrid model to jointly predict multiple resource usage metrics (e.g., CPU, memory, disk I/O) would provide a more holistic solution for cloud resource management. Benchmarking the model against more recent transformer-based time series models would also be valuable for comparative analysis.

6.7 Final Reflections

The research journey undertaken in this study highlights the intricate balance required between accuracy and efficiency in the context of cloud workload forecasting. The exploration of frequency domain techniques alongside traditional time series modeling opened new pathways for improving prediction quality without excessive computational cost. This project has been both challenging and rewarding, reinforcing the importance of thoughtful model design in modern cloud computing applications.

6.8 Concluding Remarks

In conclusion, this thesis presents a lightweight and effective approach for forecasting CPU usage in cloud environments by combining time and frequency domain insights. The findings contribute meaningfully to both academic research and practical applications, offering a scalable solution for cloud service providers aiming to enhance resource efficiency. The work lays a solid foundation for future exploration into lightweight, adaptive, and multi-metric predictive models in the dynamic field of cloud computing.

References

[1] M. Alhartomi, A. Salh, L. Audah, S. Alzahrani, and A. Alzahmi, "Enhancing sustainable edge computing offloading via renewable prediction for energy harvesting," *IEEE Access*, vol. 12, pp. 74 011–74 023, 2024.

[2] Alibaba, *Alibaba cluster trace data v2017*, `https://github.com/alibaba/clusterdata/tree/master/cluster-trace-v2017`, Accessed: 2025-05-28, 2017.

[3] Alibaba, *Alibaba cluster trace data v2018*, `https://github.com/alibaba/clusterdata/tree/master/cluster-trace-v2018`, Accessed: 2025-05-28, 2018.

[4] M. Alonge and O. Isreal, "Time series forecasting with deep learning: New frontiers in predictive analytics,"

[5] M. Amiri and L. Mohammad-Khanli, "Survey on prediction models of applications for resources provisioning in cloud," *Journal of Network and Computer Applications*, vol. 82, pp. 93–113, 2017.

[6] A.-F. Antonescu and T. Braun, "Simulation of sla-based vm-scaling algorithms for cloud-distributed applications," *Future Generation computer systems*, vol. 54, pp. 260–273, 2016.

[7] A. Anwar et al., "Improving docker registry design based on production workload analysis," in *16th USENIX Conference on File and Storage Technologies (FAST 18)*, 2018, pp. 265–278.

[8] D. Ardagna, S. Casolari, and B. Panicucci, "Flexible distributed capacity allocation and load redirect algorithms for cloud systems," in *2011 IEEE 4th International Conference on Cloud Computing*, IEEE, 2011, pp. 163–170.

[9] S. Bai, Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[10] F. J. Baldan, S. Ramirez-Gallego, C. Bergmeir, F. Herrera, and J. M. Benitez, "A forecasting methodology for workload forecasting in cloud systems," *IEEE Transactions on Cloud Computing*, vol. 6, no. 4, pp. 929–941, 2016.

[11] K. Bandara, C. Bergmeir, and S. Smyl, "Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach," *Expert Systems with Applications*, vol. 140, p. 112 896, 2020.

[12] M. Barati and S. Sharifian, "A hybrid heuristic-based tuned support vector regression model for cloud load prediction," *The Journal of Supercomputing*, vol. 71, no. 11, pp. 4235–4259, 2015.

[13] K. Benidis et al., "Deep learning for time series forecasting: Tutorial and literature survey," *ACM Computing Surveys*, vol. 55, no. 6, pp. 1–36, 2022.

[14] S. Bharany et al., "A systematic survey on energy-efficient techniques in sustainable cloud computing," *Sustainability*, vol. 14, no. 10, p. 6256, 2022.

[15] J. Bi, S. Li, H. Yuan, and M. Zhou, "Integrated deep learning method for workload and resource prediction in cloud systems," *Neurocomputing*, vol. 424, pp. 35–48, 2021.

[16] J. Bi, H. Ma, H. Yuan, and J. Zhang, "Accurate prediction of workloads and resources with multi-head attention and hybrid lstm for cloud data centers," *IEEE Transactions on Sustainable Computing*, vol. 8, no. 3, pp. 375–384, 2023.

[17] J. Bi, H. Yuan, S. Li, K. Zhang, J. Zhang, and M. Zhou, "Arima-based and multi-application workload prediction with wavelet decomposition and savitzky–golay filter in clouds," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 54, no. 4, pp. 2495–2506, 2024.

[18] J. Bi, L. Zhang, H. Yuan, and M. Zhou, "Hybrid task prediction based on wavelet decomposition and arima model in cloud data center," in *2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC)*, IEEE, 2018, pp. 1–6.

[19] T. Bollerslev, "Generalized autoregressive conditional heteroskedasticity," *Journal of Econometrics*, vol. 31, no. 3, pp. 307–327, 1986.

[20] G. E. Box and D. A. Pierce, "Distribution of residual autocorrelations in autoregressive-integrated moving average time series models," *Journal of the American statistical Association*, vol. 65, no. 332, pp. 1509–1526, 1970.

[21] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using arima model and its impact on cloud applications' qos," *IEEE transactions on cloud computing*, vol. 3, no. 4, pp. 449–458, 2014.

[22] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. Rose, and R. Buyya, "Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011. DOI: 10.1002/spe.995

[23] M. C. Calzarossa, M. L. Della Vedova, L. Massari, D. Petcu, M. I. Tabash, and D. Tessera, "Workloads in the clouds," in *Principles of Performance and Reliability Modeling and Evaluation: Essays in Honor of Kishor Trivedi on his 70th Birthday*, Springer, 2016, pp. 525–550.

[24] J. Cao, J. Fu, M. Li, and J. Chen, "Cpu load prediction for cloud environment based on a dynamic ensemble model," *Software: Practice and Experience*, vol. 44, no. 7, pp. 793–804, 2014.

[25] A. Casolaro, V. Capone, G. Iannuzzo, and F. Camastra, "Deep learning for time series forecasting: Advances and open problems," *Information*, vol. 14, no. 11, p. 598, 2023.

[26] V. Cerqueira, L. Torgo, and C. Soares, "Machine learning vs statistical methods for time series forecasting: Size matters," *arXiv preprint arXiv:1909.13316*, 2019.

[27] K. Cetinski and M. B. Juric, "Ame-wpc: Advanced model for efficient workload prediction in the cloud," *Journal of Network and Computer Applications*, vol. 55, pp. 191–201, 2015.

[28] T. Chai and R. R. Draxler, "Root mean square error (rmse) or mean absolute error (mae)? – arguments against avoiding rmse in the literature," *Geoscientific Model Development*, vol. 7, no. 3, pp. 1247–1250, 2014.

[29] B. Chen, M. Fang, and X. Li, "Denoising matrix factorization for high-dimensional time series forecasting," *Neural Computing and Applications*, vol. 36, no. 2, pp. 993–1005, 2024.

[30] C. Chen, N. Lu, B. Jiang, and C. Wang, "A risk-averse remaining useful life estimation for predictive maintenance," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 2, pp. 412–422, 2021.

[31] Z. Chen, J. Hu, G. Min, A. Y. Zomaya, and T. El-Ghazawi, "Towards accurate prediction for high-dimensional and highly-variable cloud workloads with deep learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 4, pp. 923–934, 2019.

[32] C. Clemm, L. Stobbe, K. Wimalawarne, and J. Druschke, "Towards green ai: Current status and future research," in *2024 Electronics Goes Green 2024+(EGG)*, IEEE, 2024, pp. 1–11.

[33] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, and R. Bianchini, "Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms," in *Proceedings of the 26th Symposium on Operating Systems Principles*, 2017, pp. 153–167.

[34] V. Cozzolino, L. Tonetto, N. Mohan, A. Y. Ding, and J. Ott, "Nimbus: Towards latency-energy efficient task offloading for ar services," *IEEE Transactions on Cloud Computing*, vol. 11, no. 2, pp. 1530–1545, 2022.

[35] F. Dama and C. Sinoquet, "Time series analysis and modeling to forecast: A survey," *arXiv preprint arXiv:2104.00164*, 2021.

[36] T. DeStefano, R. Kneller, and J. Timmis, "Cloud computing and firm growth," *The Review of Economics and Statistics*, pp. 1–47, Nov. 2023, ISSN: 0034-6535. DOI: 10.1162/rest_a_01393 eprint: https://direct.mit.edu/rest/article-pdf/doi/10.1162/rest_a_01393/2186107/rest_a_01393.pdf. [Online]. Available: https://doi.org/10.1162/rest%5C_a%5C_01393

[37] S. Di, D. Kondo, and W. Cirne, "Host load prediction in a google compute cloud with a bayesian model," in *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, IEEE, 2012, pp. 1–11.

[38] B. Dietrich, S. Nunna, D. Goswami, S. Chakraborty, and M. Gries, "Lms-based low-complexity game workload prediction for dvfs," in *2010 IEEE International conference on computer design*, IEEE, 2010, pp. 417–424.

[39] J. Dogani, F. Khunjush, and M. Seydali, "Host load prediction in cloud computing with discrete wavelet transformation (dwt) and bidirectional gated recurrent unit (bigru) network," *Computer Communications*, vol. 198, pp. 157–174, 2023.

[40] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," *Advances in neural information processing systems*, vol. 9, 1997.

[41] J. Duggan, Y. Chi, H. Hacigümüş, S. Zhu, and U. Cetintemel, "Packing light: Portable workload performance prediction for the cloud," in *2013 IEEE 29th International Conference on Data Engineering Workshops (ICDEW)*, IEEE, 2013, pp. 258–265.

[42] P. Duhamel and M. Vetterli, "Fast fourier transforms: A tutorial review and a state of the art," *Signal processing*, vol. 19, no. 4, pp. 259–299, 1990.

[43] Entony, *Cloud computing performance metrics*, 2023. DOI: 10.34740/KAGGLE/DSV/6165137 [Online]. Available: https://www.kaggle.com/dsv/6165137

[44] B. Feng and Z. Ding, "Application-oriented cloud workload prediction: A survey and new perspectives," *Tsinghua Science and Technology*, vol. 30, no. 1, pp. 34–54, 2024.

[45] J. Gama et al., "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.

[46] A. Gandhi, Y. Chen, D. Gmach, M. Arlitt, and M. Marwah, "Minimizing data center sla violations and power consumption via hybrid resource provisioning," in *2011 International Green Computing Conference and Workshops*, IEEE, 2011, pp. 1–8.

[47] J. Gao, H. Wang, and H. Shen, "Machine learning based workload prediction in cloud computing," in *2020 29th international conference on computer communications and networks (ICCCN)*, IEEE, 2020, pp. 1–9.

[48] E. S. Gardner Jr, "Exponential smoothing: The state of the art," *Journal of forecasting*, vol. 4, no. 1, pp. 1–28, 1985.

[49] E. S. Gardner Jr, "Exponential smoothing: The state of the art—part ii," *International journal of forecasting*, vol. 22, no. 4, pp. 637–666, 2006.

[50] E. Gomede, *The fourier transform and its application in machine learning*, Medium, TDS Archive, Accessed: 2025-08-14, Nov. 2023. [Online]. Available: https://medium.com/the-modern-scientist/the-fourier-transform-and-its-application-in-machine-learning-edecfac4133c

[51] J. A. González Martínez, M. Bote-Lorenzo, E. Gómez-Sánchez, and R. Cano-Parra, "Cloud computing and education: A state-of-the-art survey," *Computers & Education*, vol. 80, pp. 132–151, Jan. 2015. DOI: 10.1016/j.compedu.2014.08.017

[52] P. Goodwin, "Forecasting arma processes with systematic errors when estimating the parameters over a short series," *International Journal of Forecasting*, vol. 15, no. 4, pp. 405–414, 1999.

[53] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[54] J. Guo, J. Wu, J. Na, and B. Zhang, "A type-aware workload prediction strategy for non-stationary cloud service," in *2017 IEEE 10th Conference on Service-Oriented Computing and Applications (SOCA)*, IEEE, 2017, pp. 98–103.

[55] S. Gupta and D. A. Dinesh, "Resource usage prediction of cloud workloads using deep bidirectional long short term memory networks," in *2017 IEEE international conference on advanced networks and telecommunications systems (ANTS)*, IEEE, 2017, pp. 1–6.

[56] E. Hai-hong et al., "Host load prediction in cloud based on classification methods," *The Journal of China Universities of Posts and Telecommunications*, vol. 21, no. 4, pp. 40–46, 2014.

[57] H. Han et al., "Time series forecasting model for non-stationary series pattern extraction using deep learning and garch modeling," *Journal of Cloud Computing*, vol. 13, no. 1, p. 2, 2024.

[58] A. C. Harvey, "Forecasting, structural time series models and the kalman filter," 1990.

[59] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[60] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[61] M. Hossin and M. N. Sulaiman, "A review on evaluation metrics for data classification evaluations," *International journal of data mining & knowledge management process*, vol. 5, no. 2, p. 1, 2015.

[62] C.-J. Hsu, V. Nair, V. W. Freeh, and T. Menzies, "Arrow: Low-level augmented bayesian optimization for finding the best cloud vm," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, 2018.

[63] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[64] R. Hu, J. Jiang, G. Liu, and L. Wang, "Cpu load prediction using support vector regression and kalman smoother for cloud," in *2013 IEEE 33rd international conference on distributed computing systems workshops*, IEEE, 2013, pp. 88–92.

[65] R. Hu, J. Jiang, G. Liu, and L. Wang, "Kswsvr: A new load forecasting method for efficient resources provisioning in cloud," in *2013 IEEE International Conference on Services Computing*, IEEE, 2013, pp. 120–127.

[66] R. Hu, J. Jiang, G. Liu, and L. Wang, "Efficient resources provisioning based on load forecasting in cloud," *The Scientific World Journal*, vol. 2014, no. 1, p. 321 231, 2014.

[67] P. J. Huber, "Robust estimation of a location parameter," *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, 1964.

[68] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.

[69] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International journal of forecasting*, vol. 22, no. 4, pp. 679–688, 2006.

[70] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, 2006.

[71] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 155–162, 2012.

[72] M. A. Jahin, A. Shahriar, and M. A. Amin, "Mcdfn: Supply chain demand forecasting via an explainable multi-channel data fusion network model," *Evolutionary Intelligence*, vol. 18, no. 3, p. 66, 2025.

[73] D. Janardhanan and E. Barrett, "Cpu workload forecasting of machines in data centers using lstm recurrent neural networks and arima models," in *2017 12th international conference for internet technology and secured transactions (ICITST)*, IEEE, 2017, pp. 55–60.

[74] A. Javeed, A. Borg, H. Grahn, L. Lundberg, D. Patel, and S. Shirinbab, "Improving cloud efficiency: A machine learning-based stacking model for cpu utilization prediction," in *2025 8th International Conference on Data Science and Machine Learning Applications (CDMA)*, IEEE, 2025, pp. 120–125.

[75] J.-J. Jheng, F.-H. Tseng, H.-C. Chao, and L.-D. Chou, "A novel vm workload prediction using grey forecasting model in cloud data center," in *The International Conference on Information Networking 2014 (ICOIN2014)*, IEEE, 2014, pp. 40–45.

[76] J. Jiang, J. Lu, G. Zhang, and G. Long, "Optimal cloud resource auto-scaling for web applications," in *2013 13th IEEE/ACM international symposium on cluster, Cloud, and Grid Computing*, IEEE, 2013, pp. 58–65.

[77] M. E. Karim, M. M. S. Maswood, S. Das, and A. G. Alharbi, "Bhyprec: A novel bi-lstm based hybrid recurrent neural network model to predict the cpu workload of cloud virtual machine," *IEEE Access*, vol. 9, pp. 131 476–131 495, 2021.

[78] A. Khan, X. Yan, S. Tao, and N. Anerousis, "Workload characterization and prediction in the cloud: A multiple time series approach," in *2012 IEEE Network Operations and Management Symposium*, IEEE, 2012, pp. 1287–1294.

[79] I. K. Kim, W. Wang, Y. Qi, and M. Humphrey, "Forecasting cloud application workloads with cloudinsight for predictive resource management," *IEEE Transactions on Cloud Computing*, vol. 10, no. 3, pp. 1848–1863, 2020.

[80] F. Kluge, S. Uhrig, J. Mische, B. Satzger, and T. Ungerer, "Dynamic workload prediction for soft real-time applications," in *2010 10th IEEE International Conference on Computer and Information Technology*, IEEE, 2010, pp. 1841–1848.

[81] A. S. Kumar and S. Mazumdar, "Forecasting hpc workload using arma models and ssa," in *2016 International conference on information technology (ICIT)*, IEEE, 2016, pp. 294–297.

[82] J. Kumar, D. Saxena, A. K. Singh, and A. V. Vasilakos, "A quantum controlled-not neural network-based load forecast and management model for smart grid," *IEEE Systems Journal*, vol. 17, no. 4, pp. 5714–5725, 2023.

[83] J. Kumar, R. Goomer, and A. K. Singh, "Long short term memory recurrent neural network (lstm-rnn) based workload forecasting model for cloud datacenters," *Procedia computer science*, vol. 125, pp. 676–682, 2018.

[84] J. Kumar and A. K. Singh, "Workload prediction in cloud using artificial neural network and adaptive differential evolution," *Future Generation Computer Systems*, vol. 81, pp. 41–52, 2018.

[85] A. Lackinger, A. Morichetta, and S. Dustdar, " Time Series Predictions for Cloud Workloads: A Comprehensive Evaluation," in *2024 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, Los Alamitos, CA, USA: IEEE Computer Society, Jul. 2024, pp. 36–45. DOI: 10.1109/SOSE62363.2024.00011 [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SOSE62363.2024.00011

[86] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long- and short-term temporal patterns with deep neural networks," *The 41st International ACM SIGIR Conference*, 2018.

[87] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 156–165.

[88] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[89] Y. C. Lee and A. Y. Zomaya, "Energy efficient utilization of resources in cloud computing systems," *The Journal of Supercomputing*, vol. 60, pp. 268–280, 2012.

[90] J. Li, J. Yao, D. Xiao, D. Yang, and W. Wu, "Evogwp: Predicting long-term changes in cloud workloads using deep graph-evolution learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 35, no. 3, pp. 499–516, 2024.

[91] S. Li, Y. Wang, X. Qiu, D. Wang, and L. Wang, "A workload prediction-based multi-vm provisioning mechanism in cloud computing," in *2013 15th Asia-Pacific network operations and management symposium (APNOMS)*, IEEE, 2013, pp. 1–6.

[92] S. Li, J. Bi, H. Yuan, M. Zhou, and J. Zhang, "Improved lstm-based prediction method for highly variable workload and resources in clouds," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, 2020, pp. 1206–1211.

[93] X. Li, D. Zhong, B. Ren, G. Fan, and B. Cui, "Prediction of curtain grouting efficiency based on anfis," *Bulletin of Engineering Geology and the Environment*, vol. 78, no. 1, pp. 281–309, 2019.

[94] Y. Li et al., "Spatial-temporal fusion graph neural networks for traffic flow forecasting," *Proceedings of AAAI*, 2021.

[95] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *International Conference on Learning Representations (ICLR)*, 2018.

[96] B. Lim and S. Zohren, "Time-series forecasting with deep learning: A survey," *Philosophical Transactions of the Royal Society A*, vol. 379, no. 2194, p. 20 200 209, 2021.

[97] C. Liu, C. Liu, Y. Shang, S. Chen, B. Cheng, and J. Chen, "An adaptive prediction approach based on workload pattern discrimination in the cloud," *Journal of Network and Computer Applications*, vol. 80, pp. 35–44, 2017.

[98] Y. Liu, B. Gong, C. Xing, and Y. Jian, "A virtual machine migration strategy based on time series workload prediction using cloud model," *Mathematical Problems in Engineering*, vol. 2014, no. 1, p. 973 069, 2014.

[99] T. Lorido-Botran, J. Miguel-Alonso, and J. A. Lozano, "A review of auto-scaling techniques for elastic applications in cloud environments," *Journal of grid computing*, vol. 12, no. 4, pp. 559–592, 2014.

[100] Y. Lu, J. Panneerselvam, L. Liu, and Y. Wu, "Rvlbpnn: A workload forecasting model for smart cloud computing," *Scientific Programming*, vol. 2016, no. 1, p. 5 635 673, 2016.

[101] S. Luo et al., "Characterizing microservice dependency and performance: Alibaba trace analysis," in *Proceedings of the ACM Symposium on Cloud Computing*, 2021, pp. 412–426.

[102] S. Luo et al., "The power of prediction: Microservice auto scaling via workload learning," in *Proceedings of the ACM Symposium on Cloud Computing*, 2022.

[103] H. Lyu, P. Li, R. Yan, A. Masood, B. Sheng, and Y. Luo, "Load forecast of resource scheduler in cloud architecture," in *2016 International Conference on Progress in Informatics and Computing (PIC)*, IEEE, 2016, pp. 508–512.

[104] A. I. Maiyza, N. O. Korany, K. Banawan, H. A. Hassan, and W. M. Sheta, "Vtgan: Hybrid generative adversarial networks for cloud workload prediction," *Journal of Cloud Computing*, vol. 12, no. 1, p. 97, 2023.

[105] S. Makridakis, "Accuracy measures: Theoretical and practical concerns," *International Journal of Forecasting*, vol. 9, no. 4, pp. 527–529, 1993.

[106] S. S. Manvi and G. K. Shyam, "Resource management for infrastructure as a service (iaas) in cloud computing: A survey," *Journal of network and computer applications*, vol. 41, pp. 424–440, 2014.

[107] M. Masdari and A. Khoshnevis, "A survey and classification of the workload forecasting methods in cloud computing," *Cluster Computing*, vol. 23, no. 4, pp. 2399–2424, 2020.

[108] M. Masdari, S. Nabavi, and V. Ahmadi, "An overview of virtual machine placement schemes in cloud computing," *Journal of Network and Computer Applications*, vol. 66, Jan. 2016. DOI: 10.1016/j.jnca.2016.01.011

[109] M. Masdari, S. ValiKardan, Z. Shahi, and S. Azar, "Towards workflow scheduling in cloud computing: A comprehensive analysis," *Journal of Network and Computer Applications*, vol. 66, Feb. 2016. DOI: 10.1016/j.jnca.2016.01.018

[110] R. P. Masini, M. C. Medeiros, and E. F. Mendes, "Machine learning advances for time series forecasting," *Journal of economic surveys*, vol. 37, no. 1, pp. 76–111, 2023.

[111] K. Mason, "Advances in evolutionary neural networks with applications in energy systems and the environment," Ph.D. dissertation, PhD thesis, NUI Galway, 2018.

[112] S. Mazumdar and A. S. Kumar, "Forecasting data center resource usage: An experimental comparison with time-series methods," in *International Conference on Soft Computing and Pattern Recognition*, Springer, 2016, pp. 151–165.

[113] B. K. Nelson, "Time series analysis using autoregressive integrated moving average (arima) models," *Academic emergency medicine*, vol. 5, no. 7, pp. 739–744, 1998.

[114] H. M. Nguyen, S. Woo, J. Im, T. Jun, and D. Kim, "A workload prediction approach using models stacking based on recurrent neural network and autoencoder," in *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, IEEE, 2016, pp. 929–936.

[115] T. Nguyen, T. Nguyen, B. M. Nguyen, and G. Nguyen, "Efficient time-series forecasting using neural network and opposition-based coral reefs optimization," *International Journal of Computational Intelligence Systems*, vol. 12, no. 2, pp. 1144–1161, 2019.

[116] A. Y. Nikravesh, S. A. Ajila, and C.-H. Lung, "Towards an autonomic auto-scaling prediction system for cloud resource provisioning," in *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, IEEE, 2015, pp. 35–45.

[117] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-beats: Neural basis expansion analysis for interpretable time series forecasting," *arXiv preprint arXiv:1905.10437*, 2019.

[118] A. R. S. Parmezan, V. M. Souza, and G. E. Batista, "Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model," *Information sciences*, vol. 484, pp. 302–337, 2019.

[119] J. Pomeroy, "Transforming business intelligence: Leveraging generative ai and predictive analytics in cloud environments," 2024.

[120] O. Project, *Opencloud hadoop workload dataset*, `https://opencloudproject.example.org/hadoop-workloads`, Accessed: 2025-07-20, 2010.

[121] K. Qazi, Y. Li, and A. Sohn, "Power: Prediction of workload for energy efficient relocation of virtual machines," in *Proceedings of the 4th annual Symposium on Cloud Computing*, 2013, pp. 1–2.

[122] K. Qazi, Y. Li, and A. Sohn, "Workload prediction of virtual machines for harnessing data center resources," in *2014 IEEE 7th International Conference on Cloud Computing*, IEEE, 2014, pp. 522–529.

[123] F. Qiu, B. Zhang, and J. Guo, "A deep learning approach for vm workload prediction in the cloud," in *2016 17th IEEE/ACIS International Conference on Software*

Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), IEEE, 2016, pp. 319–324.

[124] S. Qiu, "Global weighted average pooling bridges pixel-level localization and image-level classification," *arXiv preprint arXiv:1809.08264*, 2018.

[125] L. R. Rabiner, C.-H. Lee, B.-H. Juang, and J. G. Wilpon, "Hmm clustering for connected word recognition," in *International Conference on Acoustics, Speech, and Signal Processing,*, IEEE, 1989, pp. 405–408.

[126] F. Ramezani and M. Naderpour, "A fuzzy virtual machine workload prediction method for cloud environments," in *2017 IEEE International conference on fuzzy systems (FUZZ-IEEE)*, IEEE, 2017, pp. 1–6.

[127] R. Reddy, "Sustainable computing: A comprehensive review of energy-efficient algorithms and systems," *Authorea Preprints*, 2024.

[128] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: Format+ schema," *Google Inc., White Paper*, vol. 1, pp. 1–14, 2011.

[129] A. Rossi, A. Visentin, S. Prestwich, and K. N. Brown, "Bayesian uncertainty modelling for cloud workload prediction," in *2022 IEEE 15th International Conference on Cloud Computing (CLOUD)*, IEEE, 2022, pp. 19–29.

[130] M. S. Sahay, S. Ganta, B. N. V. Vardhan, and K. Srilakshmi, "A comprehensive study on improving time series forecasting precision," in *Seminars in Medical Writing and Education*, AG Editor (Argentina), vol. 3, 2024, p. 7.

[131] J. Sahni and D. P. Vidyarthi, "Heterogeneity-aware adaptive auto-scaling heuristic for improved qos and resource usage in cloud environments," *Computing*, vol. 99, no. 4, pp. 351–381, 2017.

[132] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "Deepar: Probabilistic forecasting with autoregressive recurrent networks," *International journal of forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.

[133] S. Sarikaa, S. Niranjana, and K. Sri Vishnu Deepika, "Time series forecasting of cloud resource usage," in *2021 IEEE 6th International Conference on Computing, Communication and Automation (ICCCA)*, IEEE, 2021, pp. 372–382.

[134] D. Saxena and A. K. Singh, "A comprehensive survey on sustainable resource management in cloud computing environments," *Authorea Preprints*, 2024.

[135] R. Sen, H.-F. Yu, and I. S. Dhillon, "Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting," *Advances in neural information processing systems*, vol. 32, 2019.

[136] A. Setayesh, H. Hadian, and R. Prodan, "An efficient online prediction of host workloads using pruned gru neural nets," *arXiv preprint arXiv:2303.16601*, 2023.

[137] M. Shahrad et al., "Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider," in *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, 2020, pp. 205–218.

[138] R. Shariffdeen, D. Munasinghe, H. Bhathiya, U. Bandara, and H. D. Bandara, "Adaptive workload prediction for proactive auto scaling in paas systems," in *2016 2nd International Conference on Cloud Computing Technologies and Applications (CloudTech)*, IEEE, 2016, pp. 22–29.

[139] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "Cloudscale: Elastic resource scaling for multi-tenant cloud systems," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*, 2011, pp. 1–14.

[140] Q. Shi et al., "Block hankel tensor arima for multiple short time series forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 5758–5766.

[141] N. Singh and S. Rao, "Ensemble learning for large-scale workload prediction," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 2, pp. 149–165, 2014.

[142] P. Singh, P. Gupta, and K. Jyoti, "Tasm: Technocrat arima and svr model for workload prediction of web applications in cloud," *Cluster Computing*, vol. 22, no. 2, pp. 619–633, 2019.

[143] M. Smendowski and P. Nawrocki, "Optimizing multi-time series forecasting for enhanced cloud resource utilization based on machine learning," *Knowledge-Based Systems*, vol. 304, p. 112 489, 2024.

[144] B. Song, Y. Yu, Y. Zhou, Z. Wang, and S. Du, "Host load prediction with long short-term memory in cloud computing," *The Journal of Supercomputing*, vol. 74, no. 12, pp. 6554–6568, 2018.

[145] W. Sun et al., "Lstm based link quality confidence interval boundary prediction for wireless communication in smart grid," *Computing*, vol. 103, no. 2, pp. 251–269, 2021.

[146] X. Tang, X. Liao, J. Zheng, and X. Yang, "Energy efficient job scheduling with workload prediction on cloud data center," *Cluster Computing*, vol. 21, no. 3, pp. 1581–1593, 2018.

[147] S. J. Tarsa, A. P. Kumar, and H. Kung, "Workload prediction for adaptive power scaling using deep learning," in *2014 IEEE International Conference on IC Design & Technology*, IEEE, 2014, pp. 1–5.

[148] S. J. Taylor and B. Letham, "Forecasting at scale," *The American Statistician*, vol. 72, no. 1, pp. 37–45, 2018.

[149] C. Tian et al., "Minimizing content reorganization and tolerating imperfect workload prediction for cloud-based video-on-demand services," *IEEE Transactions on Services Computing*, vol. 9, no. 6, pp. 926–939, 2015.

[150] M. Tirmazi et al., "Borg: The next generation," in *Proceedings of the fifteenth European conference on computer systems*, 2020, pp. 1–14.

[151] M. P. Toopchinezhad and M. Ahmadi, "Machine learning approaches for active queue management: A survey, taxonomy, and future directions," *Computer Networks*, p. 111 174, 2025.

[152] J. F. Torres, D. Hadjout, A. Sebaa, F. Martínez-Álvarez, and A. Troncoso, "Deep learning for time series forecasting: A survey," *Big data*, vol. 9, no. 1, pp. 3–21, 2021.

[153] K. Valarmathi and S. Kanaga Suba Raja, "Resource utilization prediction technique in cloud using knowledge based ensemble random forest with lstm model," *Concurrent Engineering*, vol. 29, no. 4, pp. 396–404, 2021.

[154] N. Vasiloglou, A. Chandrayan, V. Kuznetsov, et al., "Meta-learning for time-series forecasting," *arXiv preprint arXiv:2004.11702*, 2021.

[155] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of machine learning research*, vol. 11, pp. 3371–3408, 2010.

[156] G. M. Wamba, Y. Li, A.-C. Orgerie, N. Beldiceanu, and J.-M. Menaud, "Cloud workload prediction and generation models," in *2017 29Th international symposium on computer architecture and high performance computing (SBAC-PAD)*, IEEE, 2017, pp. 89–96.

[157] H. Wang, K. J. Mathews, M. Golec, S. S. Gill, and S. Uhlig, "Amazonaicloud: Proactive resource allocation using amazon chronos based time series model for sustainable cloud computing," *Computing*, vol. 107, no. 3, p. 77, 2025.

[158] W. Wei et al., "Efficient model selection for time series forecasting via llms," *arXiv preprint arXiv:2504.02119*, 2025.

[159] J. Wilkes, "Google cluster-usage traces v3," Google Inc., Mountain View, CA, USA, Technical Report, Apr. 2020, Posted at https://github.com/google/cluster-data/blob/master/ClusterData2019.md.

[160] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance," *Climate research*, vol. 30, no. 1, pp. 79–82, 2005.

[161] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance," *Climate Research*, vol. 30, no. 1, pp. 79–82, 2005.

[162] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," *Advances in neural information processing systems*, vol. 34, pp. 22 419–22 430, 2021.

[163] M. Xu and R. Buyya, "Brownoutcon: A software system based on brownout and containers for energy-efficient cloud computing," *Journal of Systems and Software*, vol. 155, pp. 91–103, 2019.

[164] J. Yang et al., "A cost-aware auto-scaling approach using the workload prediction in service clouds," *Information Systems Frontiers*, vol. 16, no. 1, pp. 7–18, 2014.

[165] Q. Yang, Y. Zhou, Y. Yu, J. Yuan, X. Xing, and S. Du, "Multi-step-ahead host load prediction using autoencoder and echo state networks in cloud computing," *The Journal of Supercomputing*, vol. 71, no. 8, pp. 3037–3053, 2015.

[166] Y. Yang, C. Fan, and H. Xiong, "A novel general-purpose hybrid model for time series forecasting," *Applied Intelligence*, vol. 52, no. 2, pp. 2212–2223, 2022.

[167] Z. Yang et al., "Fftnet: Fusing frequency and temporal awareness in long-term time series forecasting," *Electronics*, vol. 14, no. 7, p. 1303, 2025.

[168] P. Yazdanian and S. Sharifian, "E2lg: A multiscale ensemble of lstm/gan deep learning architecture for multistep-ahead cloud workload prediction," *The Journal of Supercomputing*, vol. 77, no. 10, pp. 11 052–11 082, 2021.

[169] J. Ye et al., "A survey of time series foundation models: Generalizing time series representation with large language model," *arXiv preprint arXiv:2405.02358*, 2024.

[170] K. Yemets, I. Izonin, and I. Dronyuk, "Enhancing the fft-lstm time-series forecasting model via a novel fft-based feature extraction–extension scheme," *Big Data and Cognitive Computing*, vol. 9, no. 2, p. 35, 2025.

[171] K. Yi et al., "Frequency-domain mlps are more effective learners in time series forecasting," *Advances in Neural Information Processing Systems*, vol. 36, pp. 76 656–76 679, 2023.

[172] H. Yu, Z. Peng, X. Ma, C. Hu, and K. Li, "Temporal clustering for predicting workload intensity of virtual machines in cloud computing environments," in *IEEE Trustcom/BigDataSE/ISPA*, 2016, pp. 1243–1248.

[173] L. Zhang, Y. Zhang, P. Jamshidi, L. Xu, and C. Pahl, "Workload patterns for quality-driven dynamic cloud service configuration and auto-scaling," in *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, IEEE, 2014, pp. 156–165.

[174] Q. Zhang, L. T. Yang, Z. Yan, Z. Chen, and P. Li, "An efficient deep learning model to predict cloud workload for industry informatics," *IEEE transactions on industrial informatics*, vol. 14, no. 7, pp. 3170–3178, 2018.

[175] X. Zhang, R. R. Chowdhury, R. K. Gupta, and J. Shang, "Large language models for time series: A survey," *arXiv preprint arXiv:2402.01801*, 2024.

[176] W. Zhong, Y. Zhuang, J. Sun, and J. Gu, "A load prediction model for cloud computing using pso-based weighted wavelet support vector machine," *Applied Intelligence*, vol. 48, no. 11, pp. 4072–4083, 2018.

[177] W. Zhong, Y. Zhuang, J. Sun, and J. Gu, "Load forecasting for cloud computing based on wavelet support vector machine," *International Journal of High Performance Computing and Networking*, vol. 14, no. 3, pp. 315–324, 2019.

[178] H. Zhou, L. Cheng, S. Yu, and C. Chen, "Tasktransfer: A meta-learning framework for few-shot cloud workload forecasting," *IEEE Transactions on Cloud Computing*, 2022. DOI: 10.1109/TCC.2022.3186429

[179] H. Zhou et al., "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of AAAI*, 2021.

[180] X. Zhou et al., "Load balancing prediction method of cloud storage based on analytic hierarchy process and hybrid hierarchical genetic algorithm," *SpringerPlus*, vol. 5, no. 1, p. 1989, 2016.

[181] Y. Zhu, W. Zhang, Y. Chen, and H. Gao, "A novel approach to workload prediction using attention-based lstm encoder-decoder network in cloud environment," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, p. 274, 2019.