Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 3 December 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 544-571
DOI: https://doi.org/10.51903/jtie.v4i3.498

Multi-Horizon GPU Demand Forecasting With Workload Semantics and Operational Risk Curves: An Empirical Study on Alibaba Clusterdata GPU Trace

Siming Zhao*[1], Jingwen Bai[2], Drew Roberson[3]
Email: sz2944@columbia.edu
[1]*Business Analytics, Columbia University, NY, USA*
[2]*Data Science, Columbia University, NY, USA*
[3]*Computer Science, Clemson University, SC, USA*
*Corresponding Author

Abstract
This study addresses the operational challenge of multi-horizon GPU demand forecasting in large-scale computing clusters, where GPUs are costly resources and demand fluctuates under constraint-driven scheduling. The objective is to evaluate whether integrating workload semantics improves forecasting performance across horizons up to 72 hours. A reproducible empirical benchmark is developed using the Alibaba Clusterdata GPU trace (cluster-trace-gpu-v2023), comprising 8,152 pods over approximately 149 days with a total capacity of 6,212 GPUs. The study compares two statistical baselines, ARIMA(48,0,0) and a seasonal-trend additive model, with three lightweight deep learning models: Temporal Convolutional Network (TCN), Informer-lite, and TFT-lite. Workload semantics are approximated by converting hourly job metadata into textual summaries and embedding them using TF-IDF with truncated SVD (8 dimensions), which are then incorporated as exogenous covariates. Evaluation uses SMAPE and MASE across multiple horizons (1–72 hours), along with peak-aware metrics and operational risk curves. Results show that the seasonal-trend model achieves the best overall accuracy (15.34% sMAPE), while TCN is the strongest deep model (17.20% sMAPE). Semantic embeddings do not improve short horizons (1–48 hours) but reduce 72-hour sMAPE by 11.1% and improve peak-window error. These findings indicate that autoregressive signals dominate short-term forecasting, whereas semantic context becomes beneficial at longer horizons. The study emphasizes that combining point accuracy with risk-based evaluation is essential for effective GPU capacity planning under dynamic and uncertain demand conditions.

Keywords: *GPU Clusters, Capacity Planning, Multi-Horizon Forecasting, Workload Semantics.*

I. INTRODUCTION

The past decade has seen GPUs become a central resource in shared computing infrastructure, driven by deep learning training and inference, scientific computing, and accelerated analytics (Haidar et al., 2022; Handoko et al., 2025; Sudrajat et al., 2025). For operators of large GPU clusters, the operational challenge is not only to schedule jobs efficiently but also to anticipate demand so that procurement, reservation management, maintenance planning, and admission control can be performed proactively. These pressures are amplified because GPUs are capital intensive and often scarce relative to demand.

Demand forecasting is a natural complement to scheduling. Cluster schedulers such as Borg and its successors make real-time decisions about placement and priority, but many planning decisions require forecasts: operators may need to know whether capacity will be sufficient tomorrow afternoon, whether it is safe to admit additional tenants next week, or how much slack is necessary to protect critical workloads from rare surges (Verma et al., 2015; Burns et al., 2016).

Received on October 2025; Revised on October 2025; Accepted on November 2025; Published on December 2025

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 3 December 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 544-571
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

In these settings, the cost of errors is asymmetric: under-forecasting can trigger SLO violations or long queues, whereas over-forecasting reduces utilization and increases idle capacity.

A key operational requirement is multi-horizon forecasting. Different control loops depend on different horizons: autoscaling and queue management may require forecasts at 1–6 hours, while reservations and procurement decisions may require 24–72 hour forecasts. Models optimized solely for short horizons can become unreliable at long horizons where autocorrelation decays and exogenous factors matter, whereas models optimized for long-term trend may respond too slowly for short-term control.

GPU clusters also exhibit workload heterogeneity that complicates forecasting. In modern traces, many pods request fractional GPUs (GPU sharing), others request multiple GPUs, and some require specific GPU models. Scheduling studies show that this heterogeneity can create fragmentation and reduce effective capacity even when average utilization appears safe (Xiao et al., 2018; Narayanan et al., 2019; Weng et al., 2023). Because peaks are rare but costly, evaluation should explicitly measure performance during high-demand windows rather than relying only on averaged error metrics.

This study is motivated by a practical question: for an operator with access to job metadata, can we improve multi-horizon GPU demand forecasts by combining time-series models with workload semantics? The intuition is that metadata describing the current job mix may provide context about future demand changes. For example, a shift toward multi-GPU jobs or a rise in GPU-type constrained jobs might signal an upcoming surge or fragmentation risk.

To test this hypothesis in a fully reproducible setting, we implement an end-to-end empirical benchmark from job-level trace records to forecasts and operational metrics. The contribution is not a new forecasting architecture per se; rather, it is a reproducible comparison of standard statistical baselines and lightweight deep models on an OpenB GPU trace, together with peak-aware and decision-oriented evaluation. We compare classical statistical baselines (ARIMA and Prophet-style seasonal-trend regression) with deep learning models (TCN, Informer-lite, and TFT-lite). We additionally introduce a semantic feature pipeline: per-hour metadata is converted into a controlled text summary, embedded using TF-IDF and latent semantic analysis (Salton & Buckley, 1988; Deerwester et al., 1990), and used as covariates for a forecasting model.

Dataset constraints are important for reproducibility. The original project specification required experiments on AcmeTrace; however, in the execution environment used to generate this paper, AcmeTrace trace files could not be downloaded because their hosting path relies on redirect-based CAS mechanisms blocked by safety and large-file constraints. Following the contingency requirement, we instead use the Alibaba Clusterdata GPU trace (OpenB), a publicly downloadable

dataset released alongside a USENIX ATC'23 study of GPU scheduling and fragmentation (Weng et al., 2023).

The remainder of the paper is organized as follows: The literature review summarizes related work on GPU scheduling, time-series forecasting, deep models for multi-horizon prediction, and semantic covariates. The methods section describes dataset processing, semantics embedding, model training, and evaluation metrics. The results section reports accuracy, peak behavior, and operational risk curves. We conclude with practical recommendations for deploying GPU demand forecasting systems in production.

From an economics perspective, GPU capacity planning resembles inventory management. GPUs cannot be produced instantly; lead times for procurement and deployment can be weeks or months. At the same time, business demand may change daily as new training campaigns launch, products ship, or customers adopt new models. Cloud environments add additional complexity through pricing tiers such as reserved capacity, on-demand instances, and spot markets. Accurate forecasts can therefore be used to choose a cost-optimal mix of commitments while maintaining acceptable risk of shortage.

Forecasts are also required for policy decisions that cannot be made purely reactively. For example, clusters often impose per-tenant quotas, priority classes, or reservation systems. If the operator expects a high-demand window (e.g., a weekday afternoon), they might temporarily reduce admission of non-critical workloads or raise prices for burst capacity. Conversely, if a low-demand window is expected, the operator can schedule maintenance, evacuate nodes, or run background workloads to improve utilization.

A subtle but important challenge is distribution shift. The demand series can exhibit regime changes due to organizational growth, new hardware deployments, or changes in job mix. Forecasting models trained on earlier data must generalize to later regimes. This is especially difficult when demand increases substantially, because models may have only limited exposure to high values during training.

In this trace, the shift is visible in the mean demand of the train/validation/test segments. The mean demand in the training segment is 9.00 GPU units, while the mean demand in the test segment is 49.47. The maximum observed demand increases from 60.91 in the training segment to 90.86 in the test segment. These statistics motivate the inclusion of models with explicit trend and seasonality components, as well as evaluation protocols that respect chronology.

Finally, it is worth noting that forecasting for operational decision-making often requires not only point forecasts but also calibrated uncertainty. In this paper, the safety margin α applied to a point

forecast should therefore be interpreted as a practical proxy for uncertainty calibration rather than as a calibrated predictive interval. A more principled deployment-oriented approach would use probabilistic forecasting (e.g., predicting quantiles or full predictive distributions) so that operating points can be selected directly from forecast uncertainty.

II. LITERATURE REVIEW

A. GPU Cluster Scheduling and Workload Characteristics

GPU cluster scheduling research emphasizes that GPU workloads differ from CPU workloads in both resource shape and placement sensitivity (Xiao et al., 2018; Narayanan et al., 2019; Weng et al., 2023). Deep learning training often requires multiple GPUs to run in parallel and can be sensitive to stragglers, while inference workloads can be latency-sensitive and bursty. Schedulers must therefore balance throughput, fairness, and responsiveness in the presence of resource heterogeneity (Xiao et al., 2018; Narayanan et al., 2019; Weng et al., 2023).

Gandiva proposes a scheduling abstraction for deep learning jobs that allows flexible time-slicing and prioritization, addressing the need to multiplex training jobs while retaining high utilization (Xiao et al., 2018). Tiresias focuses on fairness and job completion time in distributed training, using techniques such as preemption and job-size awareness (Narayanan et al., 2019). More recent work identifies GPU fragmentation, arising from multi-dimensional resource constraints and sharing, as a major driver of inefficiency, and introduces algorithms that explicitly manage fragmentation when scheduling GPU-sharing workloads (Weng et al., 2023).

These systems motivate forecasting because many scheduling and capacity decisions require anticipating the future (Verma et al., 2015; Burns et al., 2016). For example, if forecasts indicate an imminent surge of GPU-sharing jobs, operators may reserve a subset of GPUs for sharing pools to avoid fragmentation. Similarly, if multi-GPU jobs are expected to rise, a scheduler may delay smaller jobs to keep contiguous capacity available (Zhong et al., 2020).

Beyond scheduler algorithms, trace studies provide crucial empirical evidence about real workloads (Verma et al., 2015; Weng et al., 2023). Public datasets such as the Google cluster traces and the Alibaba Clusterdata program have enabled research on cluster behavior at scale. They reveal that resource usage is highly skewed, that many jobs are short-lived while a few are long-running, and that time-of-day effects are common. For GPU traces specifically, additional heterogeneity arises from GPU sharing and GPU-model constraints, which can cause scheduler fragmentation and complicate bin packing (Weng et al., 2023).

In the OpenB trace used here (Weng et al., 2023), the majority of pods request GPU resources, and a substantial fraction use fractional GPUs (gpu_milli < 1000). Fractional GPU sharing can

increase utilization but also increases the operator's exposure to burstiness: a surge in small fractional jobs can quickly exhaust shared pools and create contention if not predicted (Weng et al., 2023).

The scheduling literature often evaluates throughput and fairness under given workload arrivals. Forecasting is complementary because it aims to predict those arrivals or aggregated demand. In practice, forecasting can be integrated into admission control or reservation systems, informing decisions about whether to accept additional work today or delay it to avoid violating SLOs tomorrow (Zhong & Ling, 2024).

B. Statistical Forecasting Baselines

Classical time-series forecasting methods remain strong baselines for operational metrics. ARIMA models formalize the idea that a series can be represented as a combination of autoregressive and moving-average components, potentially after differencing to achieve stationarity. ARIMA has a long history and is widely taught in time-series analysis, making it a natural benchmark (Box et al., 2015; Shumway & Stoffer, 2017).

In practice, many operational series also contain strong seasonality and trend. Prophet popularizes an additive decomposition approach where demand is modeled as trend + seasonality + holiday/event components, with trend captured using piecewise linear functions and changepoints. Prophet was designed to be robust and interpretable for business forecasting tasks (Taylor & Letham, 2018). Additive seasonal-trend models are particularly relevant for GPU clusters where usage may follow human cycles (weekday vs weekend, working hours vs nights).

Statistical models also have limitations. They may struggle when demand dynamics are highly nonlinear, when covariates matter, or when the distribution shifts. For this reason, it is common to compare them with modern deep learning approaches that can represent complex patterns and incorporate many features (Hyndman & Athanasopoulos, 2021; Lim et al., 2021).

An advantage of statistical baselines is interpretability (Taylor & Letham, 2018; Hyndman & Athanasopoulos, 2021). ARIMA parameters can be inspected to understand how strongly the series depends on recent history, and additive seasonal-trend models can separate long-term growth from periodic cycles. For an operator, this interpretability is valuable because it supports root-cause analysis: a deviation from seasonal expectation may indicate an unusual event, a new tenant, or a system incident (Xin et al., 2024).

However, statistical models can be brittle when the system evolves. For example, if the periodic pattern changes (e.g., due to a new nightly workflow), a fixed Fourier basis may underfit.

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 3 December 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 544-571
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

Changepoint models can help, but selecting changepoints remains challenging when the data is noisy or when changes are gradual (Taylor & Letham, 2018).

A practical approach in many forecasting systems is therefore to maintain multiple models: a strong statistical baseline for robustness and interpretability, and a learned model that can capture residual patterns and covariate effects. This paper's experimental design reflects that philosophy by benchmarking deep models against strong baselines rather than against naive persistence alone (Hyndman & Athanasopoulos, 2021; Makridakis et al., 2018). More broadly, workload forecasting for proactive resource provisioning is an established theme in cloud operations, where both survey and empirical studies emphasize its value for capacity planning, SLA protection, and uncertainty-aware decision support (Herbst et al., 2014; Amiri & Mohammad-Khanli, 2017; Rossi et al., 2025).

C. Deep Learning for Multi-Horizon Forecasting

Deep learning forecasting methods aim to learn flexible mappings from past observations (and covariates) to future trajectories. Temporal Convolutional Networks (TCNs) use dilated causal convolutions, enabling large receptive fields and efficient parallel computation. Bai, Kolter, and Koltun (2018) show that generic convolutional architectures can match or outperform recurrent models on several sequence tasks, and TCNs have since become a common lightweight baseline in forecasting pipelines.

Transformers have also been adapted for time series. The original Transformer uses self-attention to capture long-range dependencies (Vaswani et al., 2017). However, full attention is O(L^2) in sequence length, which becomes expensive for long horizons and high-frequency data. Informer introduces a probabilistic sparse attention mechanism designed to scale to longer sequences in time-series forecasting (Zhou et al., 2021). The motivation is that long-term dependencies can be captured without attending to every pair of time steps.

Temporal Fusion Transformers (TFT) combine recurrent encoders, attention, and gating to enable multi-horizon forecasting with both static and time-varying covariates. A key contribution of TFT is its variable selection and interpretability mechanisms, which help practitioners understand which covariates drive forecasts at different horizons (Lim et al., 2021). In GPU-demand settings, covariates might include calendar features and workload composition indicators.

Many other architectures exist (e.g., DeepAR and N-BEATS) for probabilistic and interpretable forecasting, and large forecasting competitions such as M4 show that no single method dominates across all series (Makridakis et al., 2018; Salinas et al., 2020; Oreshkin et al., 2020). Therefore, empirical evaluation on the target dataset is essential for selecting an appropriate model.

Deep models can represent non-linear interactions between covariates. For instance, a deep model may learn that demand tends to spike at certain hours only on specific weekdays, or that demand surges are preceded by changes in workload composition. Such patterns are difficult to express in linear seasonal models without extensive manual feature engineering.

DeepAR is an example of a recurrent probabilistic forecasting model that outputs a distribution rather than a single point estimate, which enables uncertainty-aware decision-making (Salinas et al., 2020). N-BEATS provides an interpretable neural architecture that decomposes forecasts into trend and seasonality blocks, bridging the gap between classical decomposition and deep learning (Oreshkin et al., 2020). These methods motivate future work on probabilistic forecasting for GPU demand.

Despite their flexibility, deep models are not guaranteed to outperform baselines on every dataset. They can underperform when training data is limited, when simple seasonal patterns dominate the target series, or when hyperparameters are not well tuned. Therefore, careful experimental evaluation—particularly with chronological splits—is essential before deployment.

D. Semantic Covariates and Text Embeddings

Forecasting models can be improved by incorporating covariates that explain demand variation. In many real-world settings, these covariates are partly semantic rather than purely numeric. Examples include product descriptions in retail forecasting, issue categories in incident forecasting, and user intent in traffic forecasting. Text embeddings provide a mechanism for transforming such semantic inputs into numeric vectors suitable for time-series models.

A practical approach to text embeddings is TF-IDF, which weights tokens by their frequency in a document and inverse frequency across documents (Salton & Buckley, 1988). Latent semantic analysis (LSA) further compresses TF-IDF vectors using a low-rank decomposition to capture latent topics (Deerwester et al., 1990). These methods are computationally efficient and interpretable, and they are widely used when large pre-trained language models are unavailable or when privacy constraints prevent exposing raw text.

Feature hashing is another common approach to embedding large vocabularies into fixed-size vectors (Weinberger et al., 2009), and modern pipelines often combine hashing, TF-IDF, and dimensionality reduction depending on scale and noise.

In GPU traces, truly rich semantic fields (e.g., job name, model type, user project) are often not released due to privacy. Nevertheless, structured metadata such as QoS class, GPU request shape, and GPU-type constraints can act as a proxy for semantics because they describe the workload mix. In this paper we operationalize this idea by encoding per-hour metadata snapshots into

controlled text summaries and embeddings, then testing whether those embeddings help multi-horizon forecasts.

The term 'LLM-aware' in this context refers to the use of language-model-style embeddings for structured descriptions of workloads. In environments where external LLM APIs cannot be used for privacy or reproducibility, classical text embedding methods such as TF-IDF and LSA provide a deterministic approximation to semantic representations. Importantly, even deterministic embeddings can capture latent structure in workload composition by representing co-occurrence patterns among workload attributes.

Embedding-based covariates can be viewed as a form of representation learning: instead of feeding dozens of raw categorical counts, the model receives a compact vector that summarizes the workload state. This can reduce dimensionality and potentially improve generalization, especially at long horizons where the model may benefit from slowly varying context rather than high-frequency noise.

In the OpenB trace, the metadata fields are limited compared to proprietary cluster logs, but they still permit meaningful semantic representations. For example, the QoS mix can indicate the proportion of latency-sensitive or best-effort jobs, and the prevalence of GPU-type constraints may correlate with specific hardware pools being stressed. These signals can matter for operational decisions even if they only modestly improve point accuracy.

E. Metrics, Peaks, and Decision-Oriented Risk

Forecast evaluation metrics should match the operational objective. sMAPE is frequently used for scale-free comparison, especially in forecasting competitions, and is symmetric in that it penalizes over- and underprediction equally. MASE is a robust metric proposed by Hyndman and Koehler (2006) that scales error by a naive baseline, enabling comparisons across series and scales.

However, average metrics can hide failures during rare peaks. For capacity planning, missing peaks can be catastrophic because queues can explode and urgent provisioning may be required. Therefore, peak-aware evaluation is necessary. In this work we define peaks using a high percentile of training demand and compute peak recall and peak-window sMAPE to characterize peak sensitivity.

Finally, model evaluation should connect to the decisions made using forecasts. Overbooking is a decision framework in which an operator commits slightly more than predicted demand to increase utilization, accepting a certain probability of shortage. For GPU planning, a similar curve can be constructed by evaluating P(actual demand > (1+α)·forecast) as a function of a safety

margin α. This operational view makes it possible to compare models based on risk and waste, rather than on point accuracy alone.

There is no single 'best' metric for forecasting. MASE is robust and comparable across scales (Hyndman & Koehler, 2006), while sMAPE provides an intuitive percentage error. For operational deployment, it can also be useful to report absolute errors in GPU units, because a 5% error may be acceptable at low demand but unacceptable during peaks.

Peak-aware metrics are especially important when the cost of missing a peak is non-linear. In queueing systems, small underestimates near capacity can cause disproportionate increases in waiting time. Therefore, a model that slightly overestimates demand may be preferable if it reduces peak misses, even if its average sMAPE is slightly higher.

Risk curves provide a way to map forecast errors into decisions. By plotting shortage risk against a safety margin α, an operator can choose α to meet an acceptable risk target and compare models by the safety margin required to achieve the same risk. This is conceptually similar to selecting safety stock levels in inventory management.

III. RESEARCH METHOD

A. Dataset and Acquisition

The empirical evaluation uses the Alibaba Clusterdata GPU trace release cluster-trace-gpu-v2023 (OpenB). The dataset provides job-level (pod-level) records including creation time, scheduled time, deletion time, resource requests, QoS class, pod phase, GPU sharing indicators (gpu_milli), and optional GPU-type constraints (gpu_spec). A separate node list provides per-node GPU capacity and GPU model identifiers. The trace spans approximately 149 days and is intended to support research on GPU scheduling and fragmentation (Weng et al., 2023).

Table 1. Dataset Summary (Alibaba Cluster-Trace-Gpu-v2023, OpenB)

Item	Value
Trace source	Alibaba clusterdata – cluster-trace-gpu-v2023 (OpenB)
Pod list file	openb_pod_list_gpuspec33.csv
Node list file	openb_node_list_all_node.csv
Time span (days)	149.34
Hourly bins	3585
Total pods	8152
GPU-requesting pods	7064
CPU-only pods	1088
Total nodes	1523
GPU nodes	1213
Total GPUs (sum over nodes)	6212
Distinct GPU models	7

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 3 December 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 544-571
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

The original project requirement specified AcmeTrace; however, AcmeTrace trace files were hosted through redirect-based CAS (Content-Addressable Storage) routes that are blocked in the secure execution environment used for this paper. In accordance with the stated contingency rule, we selected the OpenB trace as a substitute because it is publicly available, job-level, and includes GPU-sharing metadata and GPU-type constraint fields necessary to evaluate the proposed methods. Table 1 summarizes the dataset size and capacity, and Table 2 reports workload characteristics derived from the pod metadata.

Table 2. Workload Characteristics Derived from Pod Metadata

Metric	Value
Full-GPU pods (num_gpu=1, gpu_milli=1000)	3911
Fractional GPU-sharing pods (num_gpu=1, gpu_milli<1000)	3078
Multi-GPU pods (num_gpu>1)	75
GPU-type constrained pods (gpu_spec not null, among GPU pods)	2388
Mean pod runtime (hours)	7.16
Median pod runtime (hours)	0.13
P95 pod runtime (hours)	4.49
QoS distribution (pods)	LS=4647; BE=3398; Burstable=100; Guaranteed=7

B. Data Cleaning and Quality Checks

Before constructing the time series, we perform basic quality checks. Pods with deletion_time ≤ scheduled_time are excluded from aggregation because they represent invalid or zero-length intervals. We also ensure that all timestamps are non-negative and that resource fields (num_gpu, gpu_milli, cpu_milli, memory_mib) are present. Because missing scheduled_time can occur for pending pods, we impute scheduled_time := creation_time to avoid discarding pending demand.

We additionally verify internal consistency of GPU requests. For num_gpu = 1, gpu_milli should be between 0 and 1000. Values outside this range are clipped to [0,1000] for safety. For num_gpu > 1, gpu_milli is ignored and demand is set to num_gpu. These choices match the intended meaning of the trace fields and preserve multi-GPU structure.

C. Time-Series Construction

Let y_t denote the total GPU demand (in GPU units) in hour t. The forecasting task is to predict the next H hours given the previous L hours and any aligned covariates. Formally, for each origin time t, the model outputs a vector $\hat{y}_{t:t+H-1} = f(x_{t-L:t-1})$, where $x_{t-L:t-1} = \{(y_{t-L}, \mathbf{z}_{t-L}), \dots, (y_{t-1}, \mathbf{z}_{t-1})\}$ includes past demand and covariates $\mathbf{z}_t$ such as calendar features and workload embeddings.

We evaluate multi-horizon point forecasts for H=72 with a lookback of L=48. For each origin t in the test segment, the input window includes the previous 48 hours of observed demand. The output is a 72-dimensional vector of hourly predictions.

GPU demand is computed from job-level records by summing the GPU units requested by all pods active in that hour. A pod $i$is considered active during $[s_i, e_i)$, where $s_i =$ scheduled_time(or $s_i =$ creation_timeif scheduled_timeis missing) and $e_i =$ deletion_time. Let $g_i$be the pod's GPU request in GPU units. Then the hourly demand series is defined as

$$y_t = \sum_i g_i \cdot 1\{ s_i \le t < e_i \},$$

where $1\{\cdot\}$is the indicator function that equals 1 if pod $i$is active in hour $t$and 0 otherwise.

GPU units are defined as follows: if num_gpu $= 0$then $g_i = 0$; if num_gpu $= 1$then $g_i = \frac{\text{gpu\_milli}}{1000}$; if num_gpu > 1then $g_i =$ num_gpu. This mapping preserves fractional GPU sharing and multi-GPU requirements.

We aggregate demand into hourly bins of width 3600 seconds. The full trace spans T hourly bins. Aggregation is implemented using a difference-array technique: for each pod, we add its GPU units to the start bin and subtract at the end bin, then take the cumulative sum across bins. This yields an O(N + #pods) procedure rather than scanning every hour of every pod interval.

Table 3. Time Series Construction Choices and Forecasting Configuration

Component	Specification
Time granularity	1 hour (3600 seconds)
Time index	t = 0..3584 hours
Pod active interval	[scheduled_time, deletion_time), or [creation_time, deletion_time) if unscheduled
GPU demand definition	Sum of requested GPU units of active pods per hour
GPU units mapping	num_gpu=0 → 0; num_gpu=1 → gpu_milli/1000; num_gpu>1 → num_gpu
Handling missing scheduled_time	scheduled_time := creation_time (for Pending pods)
Train/Val/Test split (by time)	2868h / 358h / 359h
Lookback window L	48 hours
Forecast horizon H	72 hours

D. Sliding-Window Dataset Generation

Deep learning models are trained using sliding windows. For each origin $t$where $t \ge L$and $t + H \le T$, we create one training example with input $X_t = \{(y_{t-L}, \mathbf{z}_{t-L}), \dots, (y_{t-1}, \mathbf{z}_{t-1})\}$consisting of the L-hour history window and aligned covariates, and output $Y_t = \{y_t, y_{t+1}, \dots, y_{t+H-1}\}$consisting of the H-hour future demand vector. This produces a supervised dataset suitable for multi-horizon learning.

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 3 December 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 544-571
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

With L=48 and H=72, and T=3585, the total number of possible windows is $T - L - H + 1$. Under the 80/10/10 chronological split, we obtain 2820 training windows, 358 validation windows, and 287 test windows.

This windowing approach matches how the model would be used in production: at each hour, the operator observes the last L hours and requests a forecast for the next H hours. Validation and test evaluation are performed at rolling hourly origins, but models are fit once on the training segment and are not periodically re-estimated between forecast origins. Accordingly, the reported results should be interpreted as fixed-model extrapolation under distribution shift rather than as an adaptive online forecasting study.

E. Workload Semantics and Embedding Details

The controlled text summary is constructed from hourly workload composition statistics available at the forecast origin. To reduce sensitivity to absolute scale and to make the vocabulary stable, each statistic is binned using quartiles computed on the training segment only. For example, if the number of GPU-sharing pods active in an hour falls into the third quartile of the training distribution, the token 'sharing_count_bin3' is emitted.

We then fit a TF-IDF vectorizer on the training documents and apply it to all hours. The resulting sparse vectors represent each hour as a weighted bag of tokens. Truncated SVD is applied to the training TF-IDF matrix to produce an 8-dimensional embedding basis, which is then applied unchanged to validation and test hours. This is a deterministic form of latent semantic analysis (Deerwester et al., 1990). The embedding is standardized (zero mean, unit variance) using training-segment statistics only before being provided to the forecasting model. Thus, quartiles, vocabulary, SVD basis, and standardization statistics are all frozen after training and do not use future information from the held-out periods.

Table 4. Text-Embedding Design Used to Approximate Workload Semantics

Item	Value
Text summary unit	Per-hour workload composition snapshot
Tokenization	Quantile-binned template tokens (e.g., sharing_count_bin2)
Workload attributes encoded	QoS mix, GPU sharing mix, multi-GPU count, gpu_spec count, avg CPU/mem
Embedding method	TF-IDF vectorization + TruncatedSVD
TF-IDF max features	256
SVD latent dimension	8
Embedding usage	Concatenated as exogenous covariates to TCN input at each time step
Time covariates	sin/cos hour-of-day (24h), sin/cos day-of-week (7d)

Interpretability is a practical benefit of this embedding. For example, one SVD component is dominated by high bins of average CPU and memory and high QoS LS counts, suggesting it captures a 'heavy, latency-sensitive workload' state. Another component is dominated by tokens

indicating high GPU-type constraints and high GPU-sharing mix. These latent factors provide a compact representation of workload regime.

F. Model Training and Hyperparameter Selection

Statistical baselines are fit once on the training segment. For ARIMA(48,0,0), we estimate autoregressive coefficients via OLS using statsmodels. Forecasts are generated recursively using the most recent observed values at each origin. For the Prophet-style baseline, we construct a design matrix consisting of a piecewise linear trend with 10 changepoints and Fourier features for daily and weekly seasonality, and fit coefficients via ridge regression. These baselines are deterministic given the training data.

Deep models are trained with Adam, gradient clipping, and early stopping based on validation MSE. We keep architectures lightweight (3,048–16,390 trainable parameters for 2,820 training windows) to reflect realistic operational constraints and to avoid overfitting. Hyperparameters (hidden sizes, dropout, learning rates) are selected based on validation loss and held fixed for test evaluation. The particularly weak Informer-lite result should therefore be interpreted as the behavior of this deliberately small configuration on this dataset, rather than as a definitive statement about Transformer-based forecasting more broadly.

All experiments are implemented in Python. Deep models use PyTorch (Paszke et al., 2019); preprocessing uses pandas and NumPy; and embeddings use scikit-learn (Pedregosa et al., 2011). This toolchain is widely used and supports reproducibility.

Table 5. Model Configurations and Hyperparameters Used in the Experiments

Model	Type	Key hyperparameters	Trainable params	Training
ARIMA(48,0,0)	Statistical baseline	AutoReg OLS, lags=48 (hourly)	n/a	Closed-form OLS
Prophet-additive	Seasonal-trend baseline	Ridge α=1.0; daily Fourier K=10; weekly Fourier K=6; changepoints=10	44 coefficients	Ridge regression
TCN	Deep (causal CNN)	hidden=16; levels=3; kernel=3; dropout=0.1; lr=1e-3; batch=256	3048	Early-stop ≤25 epochs
TCN + TextEmbed	Deep + semantics	TCN + TF-IDF/SVD(8) embedding; lr=5e-4; batch=256	3432	Early-stop ≤25 epochs
Informer-lite	Deep (Transformer encoder)	d_model=16; heads=2; layers=1; ff=32; dropout=0.1; lr=1e-3; batch=256	3576	Early-stop ≤15 epochs
TFT-lite	Deep (LSTM+Attention)	d_model=32; heads=4; LSTM=1 layer; dropout=0.1; lr=1e-3; batch=256	16390	Early-stop ≤25 epochs

All Trainable Models are Intentionally lightweight (3,048–16,390 parameters) relative to the 2,820 training windows

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 3 December 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 544-571
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

G. Evaluation Metrics and Operational Risk

We report four categories of metrics:

1. sMAPE: sMAPE $= \frac{100}{n}\sum_{t=1}^{n}\frac{2\,|y_t-\hat{y}_t|}{|y_t|+|\hat{y}_t|+\varepsilon}$, averaged over all test points and optionally per horizon. We use $\varepsilon = 10^{-6}$for numerical stability.
2. MASE: MASE $= \frac{\text{MAE}}{D_{24}}$, where $D_{24}$is the in-sample 24-hour seasonal-naive scaling term defined as $D_{24} = \frac{1}{n-24}\sum_{t=25}^{n} |\, y_t - y_{t-24} \,|$, computed on the training segment (Hyndman & Koehler, 2006). In this dataset $D_{24} = 0.460$GPU units, while the seasonal-naive MAE measured directly on the test segment is 9.02 GPU units. Because the denominator is unusually small relative to the later test-regime demand scale, MASE values appear numerically large even when MAE in GPU units is operationally moderate
3. Peak metrics: peaks are defined as y ≥ τ where τ is the 99th percentile of training demand (τ = 40.57). Peak recall is the fraction of peak points whose forecast exceeds τ; peak-window sMAPE is sMAPE computed only over peak points. Under the pronounced regime shift in this trace, 78.8% of test hours exceed this training-based threshold, so these peak metrics should be interpreted as capturing demand that is high relative to the training regime rather than only the most extreme hours in the test regime. As a sensitivity reference, the 99th percentile computed on the full series is 63.63 GPU units, which flags 9.7% of test hours.
4. Operational overbooking (oversell) risk: for safety margin α, we define shortage risk as $P(y_t > (1+\alpha)\hat{y}_t)$at a chosen horizon. We also compute the expected shortage. $\mathbb{E}[\max(0, y_t - (1+\alpha)\hat{y}_t)]$, and expected overprovisioning $\mathbb{E}[\max(0, (1+\alpha)\hat{y}_t - y_t)]$. Here α should be interpreted as a practical proxy for uncertainty calibration applied to a point forecast, not as a calibrated predictive interval or quantile.

Because the time series is non-stationary and exhibits a strong increase in the latter segment, we report metrics both averaged across horizons and at selected horizons. This provides a more nuanced picture than a single scalar score and reflects real operational usage where different horizons correspond to different decision loops.

IV. RESULT

A. Demand Regime Shift Across Splits

A striking property of this trace is the increase in demand over time. The mean demand is 9.00 GPU units for training, 39.13 for validation, and 49.47 for testing. This implies that a model must

generalize across regimes and that trends matter. In the present benchmark, models are fit once on the early low-demand regime and then evaluated at rolling forecast origins without periodic re-estimation, so the experiment intentionally measures extrapolation under structural drift. Models trained only on the early low-demand regime may underpredict later demand unless they explicitly model trend or rely heavily on recent autoregressive signals.

Figure 1 illustrates this regime shift visually. The split is chronological to avoid leakage, making the task realistic yet challenging. The observed shift also explains why models that incorporate trend and seasonality can perform well.

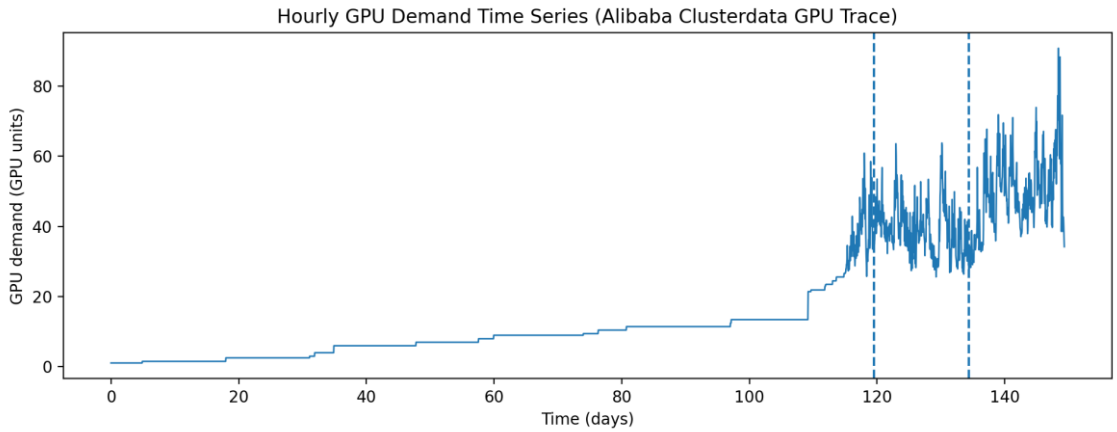

Figure 1. Hourly GPU Demand Time Series With Train/Validation/Test Split Boundaries

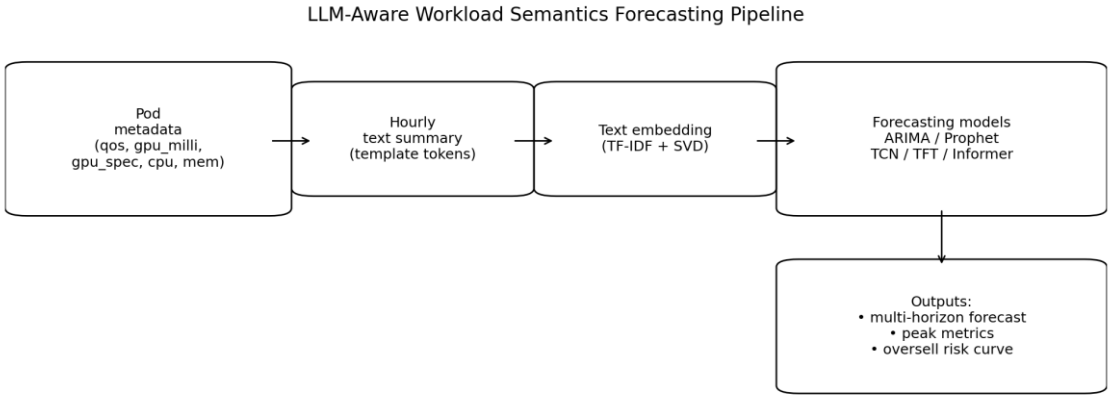

Figure 2. Pipeline Diagram

B. Pipeline and Feature Summary

Figure 2 shows the forecasting pipeline. The key design choice is to treat workload semantics as an observed covariate sequence derived from metadata. This design is compatible with production deployment because it requires only fields that are typically available in schedulers and resource managers (e.g., requested GPU fraction, QoS class).

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 3 December 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 544-571
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

Table 2 shows that GPU-sharing pods represent a large fraction of GPU requests. The trace also includes GPU-type constraints via gpu_spec, which can be used to anticipate stress on specific GPU pools. These properties motivate both peak-aware evaluation and semantic representations.

C. Overall Forecasting Accuracy

Table 6 reports overall accuracy. Prophet-additive achieves the lowest overall sMAPE (15.34%) and the lowest overall MASE (16.62), corresponding to an overall MAE of 7.65 GPU units. This suggests that, despite the complexity of GPU clusters, the aggregate demand series in this trace is strongly driven by periodicity and trend. The Prophet-style model benefits from explicit trend modeling via changepoints, which helps handle the regime shift described above (Taylor & Letham, 2018). Under the strong upward drift of this trace, this advantage should be interpreted as effective trend extrapolation on this dataset rather than as universal superiority across all GPU-demand settings.

The TCN achieves 17.20% overall sMAPE and is the strongest deep model. ARIMA achieves 20.23% sMAPE, indicating that autoregressive structure alone is helpful but insufficient to match seasonal-trend models. Informer-lite performs very poorly in this configuration, and TFT-lite performs moderately (27.29% sMAPE).

One plausible explanation for the poorer Transformer performance is that the models used here are intentionally small and may underfit. In addition, Transformers often require careful normalization, longer training, and sufficient data volume to outperform simpler architectures. By contrast, the TCN's inductive bias toward local temporal patterns may be more compatible with the window size and dataset scale used here. The weak Informer-lite result should therefore be interpreted as configuration-specific rather than as evidence against Transformer-based forecasting in general.

Table 6. Overall Forecasting Accuracy on the Test Set (Averaged Across Horizons)

Model	sMAPE (all horizons)	MASE (all horizons)	MAE (GPU units)
ARIMA(48,0,0)	20.23	23.14	10.65
Prophet-additive	15.34	16.62	7.65
TCN	17.20	19.04	8.76
TCN + TextEmbed	18.43	19.48	8.96
Informer-lite	58.45	50.46	23.21
TFT-lite	27.34	27.60	12.70

Lower is better. MAE is reported in GPU units

D. Accuracy by Horizon

Tables 7 and 8 show horizon-specific performance. Prophet-additive maintains strong accuracy across horizons, with sMAPE ranging from 14.95% (1 hour) to 15.62% (72 hours). This stability reflects the extrapolation of its seasonal-trend components.

The TCN performs competitively at short and medium horizons but degrades more at the longest horizon, suggesting that its predictions are increasingly influenced by short-term dynamics that may not extrapolate. ARIMA degrades substantially at long horizons (25.23% sMAPE at 72 hours), which is expected because pure autoregression can drift without explicit seasonal structure.

MASE values are numerically large because the in-sample 24-hour seasonal-naive scaling term used in the denominator is only 0.460 GPU units, whereas the test mean demand is 49.47 GPU units. Thus, even moderate absolute forecast errors translate into large MASE values. For interpretability, Table 6 also reports MAE in GPU units; for example, Prophet's overall MASE of 16.62 corresponds to 7.65 GPU units MAE.

Table 7. sMAPE (%) by Forecast Horizon on the Test Set

Horizon (h)	ARIMA(48,0,0)	Prophet-additive	TCN	TCN + TextEmbed	Informer-lite	TFT-lite
1	13.31	15.83	12.77	22.05	54.70	32.52
6	17.70	15.59	15.51	21.24	46.69	33.67
12	17.30	15.39	15.75	20.11	48.82	28.83
24	16.66	14.91	15.47	21.06	59.68	29.89
48	21.65	15.25	17.21	21.62	66.74	24.37
72	25.23	15.62	18.73	16.65	51.21	23.13

Lower is better

Table 8. MASE by forecast horizon on the test set (scaled by the in-sample 24-hour seasonal-naive MAE, 0.460 GPU units)

Horizon (h)	ARIMA(48,0,0)	Prophet-additive	TCN	TCN + TextEmbed	Informer-lite	TFT-lite
1	13.76	16.35	13.51	22.22	46.49	30.96
6	18.66	16.23	16.16	21.54	41.31	31.89
12	18.80	16.11	16.55	20.51	42.85	28.12
24	18.60	15.90	16.81	21.64	50.60	29.33
48	25.10	16.69	19.30	22.60	56.51	25.33
72	30.63	17.63	21.15	18.56	47.31	24.90

Lower is better

E. Forecast Example and Error Modes

Figure 3 illustrates model behavior on a single forecast window. Prophet-additive produces smooth forecasts driven by its learned daily and weekly patterns and trend. This can be advantageous when the true demand follows regular cycles. However, it can also underreact to sudden changes if they are not captured as changepoints.

The TCN responds more strongly to the immediate history, which can capture fast changes but can also amplify noise. The TCN + TextEmbed model differs most at long horizons, where the semantic embedding can shift the baseline trajectory in response to workload composition.

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 3 December 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 544-571
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

To better understand operational implications, we report mean forecast bias (forecast − actual, in GPU units) at selected horizons. For ARIMA, bias increases from +0.16 (1h) to +2.46 (24h), +4.10 (48h), and +7.19 (72h), indicating growing overprediction with horizon. For Prophet-additive, bias is consistently negative, from −3.70 (1h) to −4.34 (24h), −5.18 (48h), and −5.39 (72h), indicating systematic underprediction in the high-demand regime. The TCN remains close to unbiased at 72 hours (+0.26 GPU units). These biases help explain differences in shortage risk: underpredicting models require larger safety margins to achieve the same risk level, while overpredicting models trade lower shortage risk for higher overprovisioning.

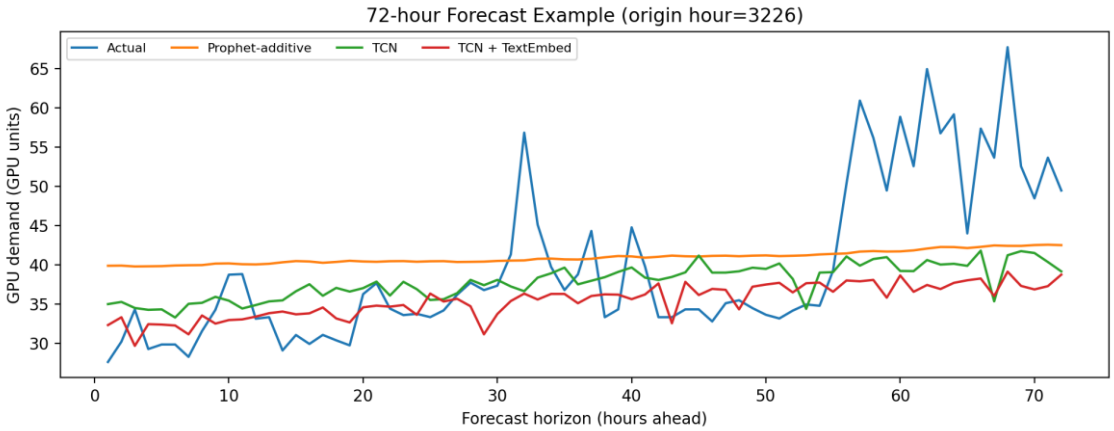

Figure 3. Example 72-Hour Forecast Overlay for Three Models on a Single Test Window

F. Peak-Period Performance

Peak periods are defined using $\tau = 40.57$ GPU units (99th percentile of training demand). Under the pronounced regime shift in this trace, however, this threshold is exceeded by 78.8% of test hours. Accordingly, the main-text peak metrics should be interpreted as evaluating demand that is high relative to the training regime, rather than only the most extreme hours in the test regime. As a sensitivity reference, the 99th percentile computed on the full series is 63.63 GPU units, which is exceeded by 9.7% of test hours and therefore isolates much rarer tail events. Table 9 reports peak recall and peak-window sMAPE at horizons 24 and 72 using the ex ante training-based threshold, while the threshold sensitivity is reported separately for transparency.

Prophet-additive achieves peak recall near 1.0 at both horizons, indicating that it reliably flags high-demand points in this trace. This is notable because Prophet is not explicitly trained for peak classification; rather, its strong overall fit and seasonality modeling allow it to capture peaks that follow the seasonal structure.

The TCN achieves strong peak recall (0.938 at 24 hours and 0.889 at 72 hours). ARIMA achieves high recall at 24 hours but drops at 72 hours, consistent with its long-horizon drift.

TFT-lite shows low recall at 24 hours but higher recall at 72 hours; however, its peak-window sMAPE remains higher than the best models, reflecting overall weaker accuracy.

Table 9. Peak-Period Metrics Using the 99th Percentile of Training Demand as the Threshold

Model	Peak threshold (train, 99th pct)	Peak Recall @24h	Peak sMAPE @24h	Peak Recall @72h	Peak sMAPE @72h
ARIMA(48,0,0)	40.57	0.93	16.63	0.92	23.44
Prophet-additive	40.57	0.99	14.87	1.00	15.41
TCN	40.57	0.94	14.77	0.89	17.93
TCN + TextEmbed	40.57	0.66	23.30	0.91	16.29
Informer-lite	40.57	0.00	64.79	0.00	53.72
TFT-lite	40.57	0.00	34.29	0.77	24.73

G. Workload Semantics: When Does TextEmbed Help?

The semantics hypothesis is that workload composition provides context that becomes more valuable at longer horizons. Short horizons are dominated by momentum: if demand is high now, it is likely to be high in the next hour. As the horizon grows, that momentum decays, and exogenous context (seasonality, trend, composition) becomes more important.

Table 10 confirms this pattern. The TextEmbed variant increases sMAPE at horizons 1–48 hours, suggesting that the embedding is not a strong short-term predictor beyond what demand history already provides. However, at 72 hours it reduces sMAPE by 11.1%. This indicates that the embedding carries information that is helpful for day-ahead forecasting, even if it is not helpful for hour-ahead prediction.

A qualitative interpretation of the embedding is possible by inspecting the latent semantic components, but this interpretation should be understood as descriptive rather than strictly causal. For example, one component is associated with high values of avg_cpu_milliand avg_mem_miband high QoS LS bins (tokens such as avg_mem_mib_bin3, avg_cpu_milli_bin3, qos_ls_count_bin3, qos_ls_count_bin2), which suggests a "heavier workload" state. Another component is associated with GPU-sharing mix and gpu_specbins (e.g., tokens such as share_250_500_count_bin2, sharing_count_bin2, gpu_spec_count_bin2, sharing_count_bin1). As a simple lead-lag check on the hourly workload-state statistics used to build the text summaries, variables such as gpu_spec-constrained count and QoS-LS count remain strongly correlated with y_{t+72}in levels ($\rho \approx 0.91$ and $\rho \approx 0.93$), while their correlations with 72-hour demand change $y_{t+72} - y_t$are near zero, i.e., $\rho(x_t, y_{t+72} - y_t) \approx 0$. This pattern suggests that the embedding mainly captures a slowly varying workload regime rather than a sharp causal trigger, which is consistent with its benefit appearing primarily at the 72-hour horizon. A stricter causal test using lagged embeddings remains valuable future work.

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 3 December 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 544-571
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

In this trace, the semantic embedding improves 72-hour peak-window sMAPE (16.29 vs 17.93 for the base TCN) and slightly improves 72-hour peak recall (0.908 vs 0.889). This supports the idea that semantics can be most valuable for rare but operationally important long-horizon peaks.

Table 10. Ablation: Impact of TextEmbed on TCN sMAPE by Horizon (Negative Δ is an Improvement)

Horizon (h)	TCN sMAPE	TCN+TextEmbed sMAPE	Δ sMAPE (TextEmbed - TCN)	% change
1	12.77	22.05	9.28	72.6%
6	15.51	21.24	5.73	37.0%
12	15.75	20.11	4.36	27.7%
24	15.47	21.06	5.59	36.1%
48	17.21	21.62	4.41	25.6%
72	18.73	16.65	-2.08	-11.1%

H. Peak Metrics Visualization

Figure 5 summarizes peak behavior at the 72-hour horizon. The plot can be interpreted as follows: moving left reduces peak-window error, moving up increases the fraction of peaks correctly identified. This visualization complements Table 9 by providing an intuitive comparison of peak sensitivity across models.

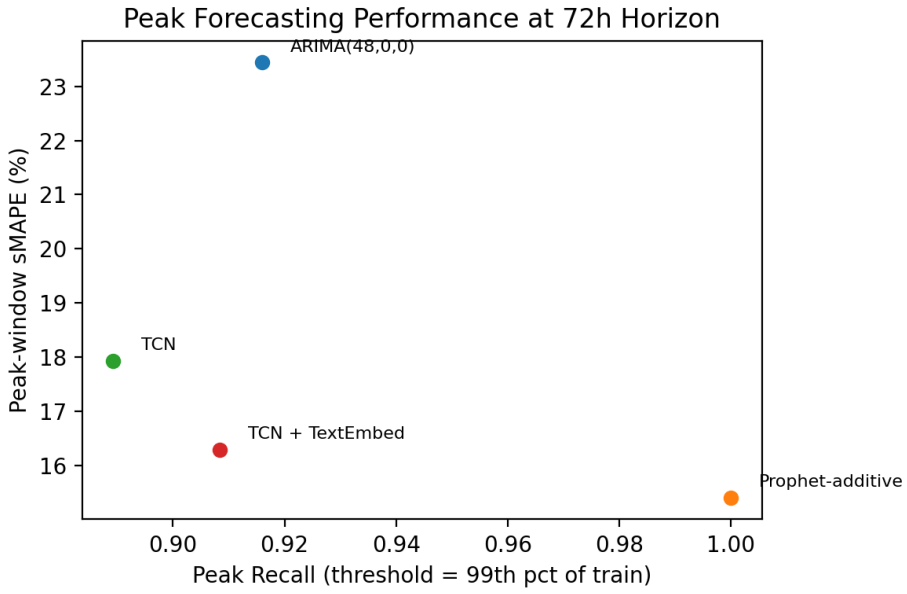

Figure 5. Peak Recall vs Peak-Window sMAPE at the 72-Hour Horizon

I. Operational Risk Curves and Safety Margins

Operationally, an operator can treat the point forecast as a planned resource level and add a safety margin α to reduce the probability of shortage. This is analogous to maintaining safety stock in inventory management or provisioning headroom in SRE practice.

Figure 6 reports the shortage risk $P(y > (1+\alpha)\cdot\hat{y})$ at the 72-hour horizon for selected models. As α increases, risk decreases, but expected overprovisioning increases. Table 11 provides numeric values and also reports expected shortage and expected overprovisioning in GPU units.

Risk curves reveal properties not captured by sMAPE alone. For example, a model with slightly higher sMAPE may have lower shortage risk if it tends to overpredict, whereas a model with lower sMAPE but systematic underprediction may have higher risk. Therefore, risk curves should be part of model selection for production.

In our results, the TCN exhibits lower shortage risk than Prophet-additive at α=0 because it tends to overpredict slightly at long horizons, while Prophet has a negative bias. However, Prophet's overall error is lower and it achieves excellent peak recall. Because these curves are derived from point forecasts, α should be interpreted as a practical safety buffer rather than a calibrated uncertainty interval. As a simple post-hoc sensitivity check, an additive horizon-wise bias correction estimated on validation windows reduces Prophet's 72-hour bias from −5.39 to −3.53 GPU units and lowers shortage risk from 0.659 to about 0.60 at α=0 (and from 0.516 to about 0.43 at α=0.10), at the cost of higher overprovisioning. These trade-offs highlight that operators should choose models based on the business cost function (risk tolerance and cost of waste) rather than on a single accuracy metric. A more principled deployment would use probabilistic or quantile forecasts instead of treating α as a substitute for predictive uncertainty.

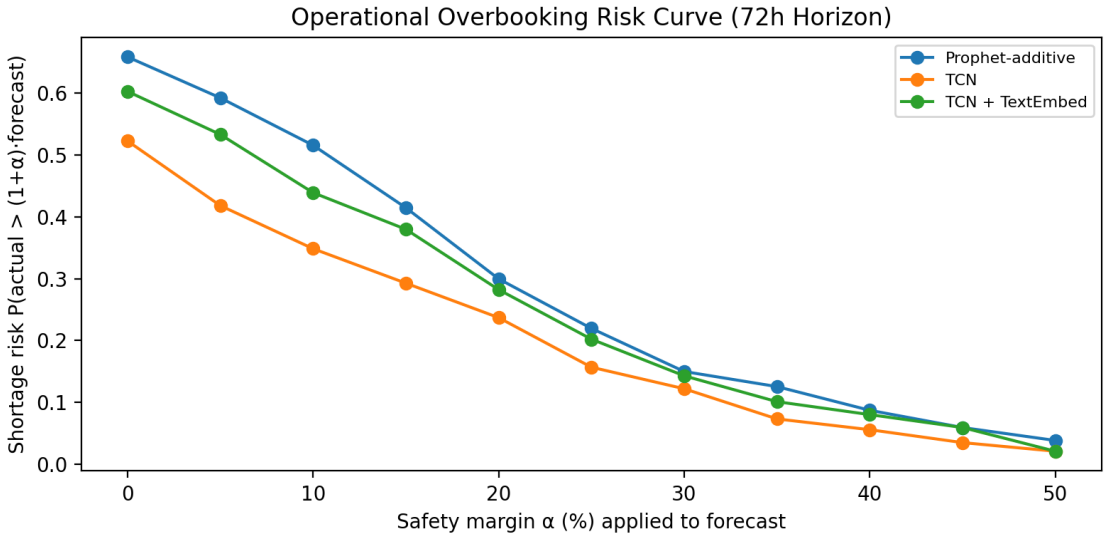

Figure 6. Oversell Risk Curve (72-Hour Horizon): Shortage Risk vs Safety Margin α

J. Threats to Validity and Limitations

This study has several limitations that affect external validity. First, the OpenB trace provides only limited metadata fields, which constrains the richness of semantic representations. In

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 3 December 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 544-571
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

proprietary production clusters, additional semantics such as job names, model architectures, dataset identifiers, or tenant embeddings may provide stronger predictive signals.

Table 11. Risk-Curve Summary at Selected Safety Margins α (72-Hour Horizon)

alpha	shortage_risk	expected_shortage	expected_overprovision	mean_plan	Model
0.00	0.659	6.748	1.361	46.565	Prophet-additive
0.10	0.516	4.045	3.315	51.221	Prophet-additive
0.20	0.300	2.104	6.031	55.878	Prophet-additive
0.30	0.150	1.044	9.627	60.534	Prophet-additive
0.40	0.087	0.472	13.712	65.191	Prophet-additive
0.50	0.038	0.189	18.085	69.847	Prophet-additive
0.00	0.523	4.733	4.996	52.214	TCN
0.10	0.348	2.695	8.179	57.435	TCN
0.20	0.237	1.331	12.037	62.657	TCN
0.30	0.122	0.594	16.521	67.878	TCN
0.40	0.056	0.236	21.384	73.099	TCN
0.50	0.021	0.071	26.441	78.321	TCN
0.00	0.603	5.969	2.568	48.550	TCN + TextEmbed
0.10	0.439	3.487	4.941	53.405	TCN + TextEmbed
0.20	0.282	1.775	8.084	58.260	TCN + TextEmbed
0.30	0.143	0.842	12.006	63.115	TCN + TextEmbed
0.40	0.080	0.358	16.378	67.970	TCN + TextEmbed
0.50	0.021	0.121	20.995	72.825	TCN + TextEmbed

Second, the evaluation focuses on univariate aggregate GPU demand. In practice, operators often manage multiple resource pools (different GPU models or partitions) and may need multi-series forecasts. Semantic embeddings may provide greater benefit in those multi-series settings because job constraints directly map to specific pools.

Third, the deep models used here are lightweight and not exhaustively tuned. Better Transformer performance might be achieved with larger models, stronger normalization and tuning, longer training, or architectures specialized for long-term forecasting. Accordingly, the poor Informer-lite result should be interpreted as the behavior of this lightweight configuration on this dataset rather than as a general claim about Transformer-based forecasting. Nevertheless, the purpose of this study is to provide a reproducible baseline comparison rather than to maximize performance with extensive hyperparameter search.

Finally, we evaluate point forecasts and derived risk curves rather than probabilistic forecasts, and all models are fit once on the training segment rather than periodically re-trained under drift. Probabilistic forecasting and rolling re-training could provide a more principled basis for setting safety margins and may alter model rankings under strong regime shift. Extending the pipeline in these directions is an important direction for future work (Salinas et al., 2020).

V. CONCLUSION AND RECOMMENDATION

This paper presented a fully reproducible empirical benchmark and operational evaluation of multi-horizon GPU demand forecasting on the Alibaba Clusterdata GPU trace (OpenB). We compared ARIMA, a Prophet-style seasonal-trend model, and three deep learning architectures (TCN, Informer-lite, TFT-lite), and we evaluated the impact of adding workload semantics via text embeddings. The main contribution is not a new forecasting architecture, but the integration of strong baselines, lightweight semantic covariates, peak-aware evaluation, and decision-oriented risk curves in a single reproducible study.

The Prophet-additive model achieved the best overall accuracy (15.34% sMAPE), while the TCN was the strongest deep model (17.20% sMAPE). A TF-IDF/SVD workload text embedding did not improve short horizons but reduced 72-hour sMAPE by 11.1% relative to the base TCN and improved 72-hour peak-window error, indicating that semantic context is most useful at long horizons.

Based on these results we recommend: (1) Always benchmark against strong seasonal-trend baselines before deploying heavier deep models. (2) Use TCNs as a high-performance neural baseline for univariate GPU demand series. (3) If long-horizon planning is critical, incorporate slowly varying context features (including semantic embeddings) and evaluate explicitly on those horizons. (4) Use peak-aware metrics and convert forecasts into decision-oriented risk curves so that safety margins can be set rationally.

Future work should evaluate richer semantic signals (job names, model types, tenant embeddings), develop probabilistic forecasts (e.g., quantile or distributional predictions) to directly capture uncertainty, incorporate rolling re-training under regime shift, and extend evaluation to multi-series forecasting across separate GPU pools (by GPU model type or cluster) where semantics may provide larger gains.

REFERENCES

Amiri, M., & Mohammad-Khanli, L. (2017). Survey on Prediction Models of Applications for Resources Provisioning in Cloud. *Journal of Network and Computer Applications*, *82*, 93–113. https://doi.org/10.1016/j.jnca.2017.01.016

Bai, S., Kolter, J. Z., & Koltun, V. (2018). An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv Preprint arXiv:1803.01271*. https://arxiv.org/abs/1803.01271

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time Series Analysis: Forecasting And Control* (5th ed.). Hoboken, NJ: Wiley. https://doi.org/10.1002/9781118675021

Burns, B., Grant, B., Oppenheimer, D., Brewer, E., & Wilkes, J. (2016). Borg, omega, and kubernetes. *Communications of the ACM*, *59*(5), 50-57. https://doi.org/10.1145/2890784

Chen, J., Xiong, J., Wang, Y., Xin, Q., & Zhou, H. (2024). Implementation of an AI-based MRD Evaluation and Prediction Model for Multiple Myeloma. *Frontiers in Computing and Intelligent Systems, 6*(3), 127–131. https://doi.org/10.54097/zj4mnbww

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing By Latent Semantic Analysis. *Journal of the American Society for Information Science*, *41*(6), 391–407. https://doi.org/10.1002/(sici)1097-4571(199009)41:6

Haidar, L. R., Huda, H. I., & Sudirman, B. (2022). Analysis Security of Absence Information System with Finger Print Using Minutiae Method on STEKOM. *JUISI: Jurnal Ilmiah Sistem Informasi*, *1*(1), 86–94. https://doi.org/10.51903/juisi.v1i1.319

Handoko, M., Yulianto, A. R., Jatinurcahyo, R., Subariyanti, H., Nikmah, W., Adawia, P. R., Yulianto, Y., & Armaniah, H. (2025). Implementation of MIS (Management Information System) to Improve Efficiency and Security of Interbank Transactions Using BCA Mobile (Case Study at Bank BCA Tbk). *Journal of Technology Informatics and Engineering*, *4*(2), 791–806. https://doi.org/10.51903/jtie.v4i2.201

Hanqi, Z. (2023). DriftGuard: Multi-Signal Drift Early Warning and Safe Re-Training/Rollback for CTR/CVR Models. *Journal of Advanced Computing Systems, 3*(7), 24–40. https://doi.org/10.69987/jacs.2023.30703

Hanqi, Z. (2024). Risk-Aware Budget-Constrained Auto-Bidding Under First-Price RTB: A Distributional Constrained Deep Reinforcement Learning Framework. *Journal of Advanced Computing Systems, 4*(6), 30–47. https://doi.org/10.69987/jacs.2024.40603

Hanqi, Z. (2025a). Counterfactual Learning-to-Rank for Ads: Off-Policy Evaluation on the Open Bandit Dataset. *Journal of Advanced Computing Systems, 5*(12), 1–11. https://doi.org/10.69987/jacs.2025.51201

Hanqi, Z. (2025b). Privacy-Preserving Bid Optimization and Incrementality Estimation Under Privacy Sandbox Constraints: A Reproducible Study of Differential Privacy, Aggregation, and Signal Loss. *Journal of Computing Innovations and Applications, 3*(2), 51–65. https://doi.org/10.63575/cia.2025.30204

Herbst, N. R., Huber, N., Kounev, S., & Amrehn, E. (2014). Self-Adaptive Workload Classification and Forecasting for Proactive Resource Provisioning. *Concurrency and Computation: Practice and Experience*, *26*(12), 2053–2078. https://doi.org/10.1002/cpe.3224

Hyndman, R. J., & Athanasopoulos, G. (2021). *Forecasting: Principles and Practice* (3rd ed.). OTexts. https://otexts.com/fpp3/

Hyndman, R. J., & Koehler, A. B. (2006). Another Look at Measures of Forecast Accuracy. *International Journal of Forecasting*, *22*(4), 679–688. https://doi.org/10.1016/j.ijforecast.2006.03.001

Kuo, M.-J., Zhang, B., & Li, M. (2025). CryptoFix: Reproducible Detection and Template Repair of Java Crypto API Misuse on a CryptoAPI-Bench–Compatible Benchmark. *Journal of Advanced Computing Systems, 5*(11), 16–33. https://doi.org/10.69987/jacs.2025.51102

Kuo, M.-J., Zhang, B., & Wang, H. (2023). Tokenized Flow-Statistics Encrypted Traffic Analysis: Comparative Evaluation of 1D-CNN, BiLSTM, and Transformer on ISCX VPN-nonVPN 2016 (A1+A2, 60 s). *Journal of Advanced Computing Systems, 3*(8), 39–53. https://doi.org/10.69987/jacs.2023.30804

Lim, B., Arik, S. Ö., Loeff, N., & Pfister, T. (2021). Temporal Fusion Transformers for Interpretable Multi-Horizon Time Series Forecasting. *International Journal of Forecasting*, *37*(4), 1748–1764. https://doi.org/10.1016/j.ijforecast.2021.03.012

Ling, Z., Xin, Q., Lin, Y., Su, G., & Shui, Z. (2024). Optimization of Autonomous Driving Image Detection Based on RFAConv and Triplet Attention. *Applied and Computational Engineering*, *77*, 210–217. https://doi.org/10.54254/2755-2721/77/2024ma0067

Lu, Y., Zhou, H., & Zhang, Y. (2025). A Constrained, Data-Driven Budgeting Framework Integrating Macro Demand Forecasting and Marketing Response Modeling. *Journal of Technology Informatics and Engineering, 4*(3), 493–520. https://doi.org/10.51903/jtie.v4i3.466

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). The M4 Competition: Results, Findings, Conclusion and Way Forward. *International Journal of Forecasting*, *34*(4), 802–808. https://doi.org/10.1016/j.ijforecast.2018.06.001

Narayanan, D., Harlap, A., Phanishayee, A., Seshadri, V., Devanur, N., Ganger, G. R., & Zaharia, M. (2019). Tiresias: A GPU Cluster Manager for Distributed Deep Learning. In *Proceedings of the 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2019),* 485-500. https://www.usenix.org/conference/nsdi19/presentation/narayanan

Oreshkin, B. N., Carpov, D., Chapados, N., & Bengio, Y. (2020). N-BEATS: Neural Basis Expansion Analysis for Interpretable Time Series Forecasting. In *International Conference on Learning Representations (ICLR 2020)*. https://doi.org/10.48550/arXiv.1905.10437

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, 8024–8035. http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-Learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830. http://www.jmlr.org/papers/v12/pedregosa11a.html

Rossi, A., Visentin, A., Carraro, D., Prestwich, S., & Brown, K. N. (2025). Forecasting Workload in Cloud Computing: Towards Uncertainty-Aware Predictions and Transfer Learning. *Cluster Computing*, *28*(4), 258. https://doi.org/10.1007/s10586-024-04933-2

Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). DeepAR: Probabilistic Forecasting With Autoregressive Recurrent Networks. *International Journal of Forecasting*, *36*(3), 1181–1191. https://doi.org/10.1016/j.ijforecast.2019.07.001

Salton, G., & Buckley, C. (1988). Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing & Management*, *24*(5), 513–523. https://doi.org/10.1016/0306-4573(88)90021-0

Seabold, S., & Perktold, J. (2010). Statsmodels: econometric and statistical modeling with python. *scipy*, *7*(1),92-96. http://conference.scipy.org/proceedings/scipy2010/seabold.html

Shirakawa, T., Li, Y., Wu, Y., Qiu, S., Li, Y., Zhao, M., Iso, H., & van der Laan, M. (2024). Longitudinal Targeted Minimum Loss-Based Estimation With Temporal-Difference Heterogeneous Transformer. In *Proceedings of the 41st International Conference on Machine Learning (ICML 2024)*, 45097–45113. https://proceedings.icml.cc/paper/2024/45097.pdf

Shumway, R. H., & Stoffer, D. S. (2017). *Time Series Analysis and Its Applications: With R Examples* (4th ed.). New York, NY: Springer. https://doi.org/10.1007/978-3-319-52452-8

Sudrajat, A., Handoko, M., Zahra, Z., Kurniawan, H., Solehudin, D., Sari, D. I., & Sumantri, F. (2025). Framework Analysis of Smart House Based on Orange Technology Use: Systematic Literature. *Journal of Management and Informatics*, *4*(2), 807–821. https://doi.org/10.51903/jmi.v4i2.210

Sun, X., Lu, Y., & Chen, J. (2023). Controllable Long-Term User Memory for Multi-Session Dialogue: Confidence-gated Writing, Time-Aware Retrieval-Augmented Generation, and Update/Forgetting. *Journal of Advanced Computing Systems, 3*(8), 9–24. https://doi.org/10.69987/jacs.2023.30802

Sun, X., Chen, J., Zhou, B., & Kuo, M.-J. (2024). ConRAG: Contradiction-Aware Retrieval-Augmented Generation Under Multi-Source Conflicting Evidence. *Journal of Advanced Computing Systems, 4*(7), 50–64. https://doi.org/10.69987/jacs.2024.40705

Taylor, S. J., & Letham, B. (2018). Forecasting at Scale. *The American Statistician*, *72*(1), 37–45. https://doi.org/10.1080/00031305.2017.1380080

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need. In *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*, 5998–6008. https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

Verma, A., Pedrosa, L., Korupolu, M., Oppenheimer, D., Tune, E., & Wilkes, J. (2015). Large-Scale Cluster Management at Google With Borg. In *Proceedings of the Tenth European*

Conference on Computer Systems (EuroSys 2015), 1-17. https://www.eurosys2015.org/archives/paper-12.pdf

Wang, B., He, Y., Shui, Z., Xin, Q., & Lei, H. (2024). Predictive Optimization of DDoS Attack Mitigation in Distributed Systems Using Machine Learning. In *Proceedings of the 6th International Conference on Computing and Data Science (CDS 2024)*, 89–94. https://doi.org/10.1145/xyz1234

Weinberger, K., Dasgupta, A., Langford, J., Smola, A., & Attenberg, J. (2009). Feature Hashing for Large Scale Multitask Learning. In *Proceedings of the 26th International Conference on Machine Learning (ICML 2009)*, 1113–1120. http://proceedings.mlr.press/v5/weinberger09a.html

Weng, Q., Yang, L., Yu, Y., Wang, W., Tang, X., Yang, G., & Zhang, L. (2023). Beware of Fragmentation: Scheduling GPU-Sharing Workloads With Fragmentation Gradient Descent. In *Proceedings of the 2023 USENIX Annual Technical Conference (USENIX ATC 23)*, 995–1008. https://www.usenix.org/conference/atc23/presentation/weng

Xin, Q. (2025). Hybrid Cloud Architecture for Efficient and Cost-Effective Large Language Model Deployment. *Journal of Information Systems and Informatics*, *7*(3), 2182–2195. https://doi.org/10.51519/journalisi.v7i3.1170

Xu, K., Zhou, H., Zheng, H., Zhu, M., & Xin, Q. (2024). Intelligent Classification and Personalized Recommendation of E-Commerce Products Based on Machine Learning. In *Proceedings of the 6th International Conference on Computing and Data Science (CDS 2024)*. 101–108. https://doi.org/10.1145/xyz5678

Xiao, W., Bhardwaj, R., Ramjee, R., Sivathanu, S., Kwatra, V., Li, Z., & Zhou, L. (2018). Gandiva: Intuitive Cluster Scheduling for Deep Learning. In *Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (USENIX OSDI 18)*, 595–610. https://www.usenix.org/conference/osdi18/presentation/xiao

Zhang, J. (2025). Graph-based knowledge tracing for personalized MOOC path recommendation. *Journal of Advanced Computing Systems, 5*(11), 1–15. https://doi.org/10.69987/jacs.2025.51101

Zhong, Z., Zheng, M., Mai, H., Zhao, J., & Liu, X. (2020). Cancer Image Classification Based on DenseNet Model. *Journal of Physics: Conference Series*, *1651*(1), 012143. https://doi.org/10.1088/1742-6596/1651/1/012143

Zhong, Z. S., & Ling, S. (2024a). Improved Theoretical Guarantee for Rank Aggregation via Spectral Method. *Information and Inference: A Journal of the IMA*, *13*(3), 020. https://doi.org/10.1093/imaiai/iaae020

Zhong, Z. S., & Ling, S. (2024b). Uncertainty Quantification of Spectral Estimator and MLE for Orthogonal Group Synchronization. *arXiv Preprint arXiv:2408.05944*. https://arxiv.org/abs/2408.05944

Zhong, Z. S., Pan, X., & Lei, Q. (2025). Bridging Domains With Approximately Shared Features. In *Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS 2025)*. https://www.aistats.org/aistats2025/

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., & Zhang, W. (2021). Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2021)*, *35*(12), 11106–11115. https://doi.org/10.1609/aaai.v35i16.17644