SG-LR QoS: Sparse-Gating and Layered Low-Rank Based Lightweight QoS Monitoring for Multi-tenant Clouds

Pengwei Liu, Minxian Xu, Jiongjiong Gu, Chuanfei Xu, Weipeng Cao*, Zhong Ming

***Abstract*—Non-intrusive Quality of Service (QoS) monitoring is a fundamental capability for proactive Service Level Agreement (SLA) management and resource optimization in multi-tenant public clouds, yet state-of-the-art models face a dual bottleneck in production deployments due to high-dimensional feature overhead and the prohibitive memory footprint of deep-stacked architectures. To address these challenges, we propose SG-LR QoS, a Sparse-Gating and Layered Low-Rank based lightweight QoS monitoring framework that co-designs data acquisition and model inference under a unified online resource budget. The framework first incorporates AnnealGate, an automated feature selection module representing the sparse-gating component, which employs differentiable L_0 regularization to learn a structured sparse input subset and prune redundant features. Building upon this, the layered low-rank component is realized through TrunkShare-LoRABank (TS-LB), a parameter-efficient mechanism centered on a shared trunk encoder layer that iteratively reuses parameters across multiple steps while injecting incremental representations via a bank of low-rank adapters. To ensure robust and stable QoS degradation predictions, this recursive process is seamlessly integrated with a Soft Routing Fusion strategy that dynamically aggregates intermediate representations from each step. Experimental evaluations on a large-scale production dataset involving 13 representative cloud applications demonstrate that SG-LR QoS achieves high-fidelity performance with a Mean Absolute Error (MAE) within a 5% margin. Compared to the state-of-the-art model, our method achieves a 40.85% feature sparsity rate while delivering a 79.2% reduction in parameter count, a 45.8% decrease in memory footprint, and a 2.75× speedup in inference latency.**

***Index Terms*—Quality of Service (QoS), Feature selection, Low-Rank Adaptation, Recursive neural networks, Cloud computing**

I. Introduction

Public clouds widely employ multi-tenant consolidation to maximize resource utilization and mitigate operational costs [1]. However, the severe resource contention triggered by highly volatile and bursty workloads frequently causes Quality of Service (QoS) degradation, manifesting as performance jitter and tail latency spikes [2], [3]. Accurate online detection of QoS degradation is thus foundational for enabling proactive elasticity and automated management [4]. Nevertheless, deploying deep learning-based QoS monitoring in production environments faces three primary challenges:

Pengwei Liu, Chuanfei Xu, Weipeng Cao, and Zhong Ming are with Guangdong Laboratory of Artificial Intelligence and Digital Economy (Shenzhen), Shenzhen, China. Minxian Xu is with Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China. Jiongjiong Gu is with Huawei Technologies Co., Ltd., Shenzhen, China.

*Corresponding author: Weipeng Cao (Email: caoweipeng123@gmail.com)

First, black-box infrastructure limits visibility into application internals. Privacy regulations and isolation policies restrict cloud providers from accessing in-guest application logs or semantics [5]. Service quality must therefore be inferred exclusively from external, host-level observables.

Second, heavy models induce deployment bottlenecks. State-of-the-art multi-layer Transformers impose substantial memory footprints and inference latency [6]. Since host-side monitoring agents must co-exist with tenant workloads, excessive model overhead directly compromises service performance.

Third, high-dimensional telemetry incurs unsustainable data costs. Modern hosts emit hundreds of metrics; collecting this high-dimensional stream at fine granularity generates prohibitive I/O and network transmission overhead, necessitating joint optimization of data efficiency and model complexity.

To address these challenges, we propose SG-LR QoS, a deployable framework that co-optimizes data acquisition and model inference under a unified resource budget. The core contributions of this work are summarized as follows:

(1) **AnnealGate for structured sparse feature selection**: We propose an automated selector that learns a deployable, structured sparse input subset via differentiable regularization. By unifying training-phase learnability and inference-phase pruning, the gating vector solidifies into a fixed mask during deployment, eliminating redundant data acquisition and transmission overhead.

(2) **TrunkShare-LoRABank (TS-LB) for parameter-efficient sequence modeling**: To circumvent deep parameter stacking, we design a recursive encoder with a shared parameter trunk. By injecting step-specific low-rank adapters from a specialized Low-Rank Adaptation (LoRA) Bank across successive recursions, the framework converts depth-driven capacity into controllable step-level overhead while multi-step insights are fused via soft routing.

(3) **Joint optimization and validated efficiency**: Evaluated on a production dataset across 13 cloud applications, SG-LR QoS achieves a 40.85% feature sparsity while maintaining high fidelity (MAE within 5%). Compared to the state-of-the-art Multi-Task QoS detection algorithm (MTQoS) [5], our framework slashes parameter count by 79.2%, decreases memory footprint by 45.8%, and accelerates inference by 2.75×.

2836-3868/26/$31.00 ©2026 IEEE
DOI 10.1109/ICWS72778.2026.00127

The rest of this paper is structured as follows: Section II reviews related work. Section III details the proposed SG-LR QoS framework. Section IV presents experimental evaluations and comparative analyses. Finally, Section V concludes the paper and outlines future research directions.

II. Related Work

A. Deep Learning for QoS Monitoring

Early QoS monitoring relied on static thresholds [7]–[9] or statistical models like ARIMA [10]–[12]. However, their linearity and stationarity assumptions fail to capture the volatile, high-dimensional interactions of modern cloud environments [1]. To address this, deep learning based algorithms—particularly Transformers like MTQoS [5]—became dominant due to their non-linear modeling capacity. Recently, foundation models (Large Language Models, LLMs) have also been introduced to cloud telemetry [13]–[16]. Nevertheless, both deep-stacked Transformers and massive foundation models impose severe memory footprints and inference latencies that violate the strict resource budgets of host-side monitoring agents. This creates a critical tension where excessive monitoring overhead risks degrading the very QoS it intends to protect, underscoring the urgent need for lightweight, deployable architectures.

B. Feature Selection in System Monitoring

To mitigate dependency on high-dimensional inputs, existing feature selection strategies generally fall into two categories [17]–[20]. One line of work constructs features robust to system noise, such as FEDGE [17], which introduces an interference-aware feature construction method to sustain generalization under varying workloads. Another approach leverages attention mechanisms for implicit feature weighting; for example, AFS [19], EAR-FS [20], and AEFS [21] utilize trainable modules to dynamically assign importance scores, suppressing noise while highlighting informative signals.

However, a fundamental limitation is that such soft selection operates exclusively within the model's internal computational graph, leaving the upstream data acquisition process unaltered. During deployment, all raw features must still be fully collected, transmitted, and preprocessed. Consequently, this post-acquisition recalibration fails to reduce actual Input/Output (I/O), bandwidth, or computational overhead. Furthermore, embedding complex selection modules (e.g., multi-layer attention or graph networks [22]) often inflates parameter counts, while optimizing solely for prediction accuracy overlooks host-side resource budgets. Thus, truly minimizing end-to-end monitoring overhead remains unrealized.

C. Parameter-Efficient Model Architectures

Parameter-efficient architectures typically rely on lightweight Transformer hybrids (e.g., AutoLDT [23], TBiGNet [24]), layer-wise parameter sharing (e.g., ALBERT [25]), or edge-tailored solutions like LPAP [26] and LightESD [27]. Although these model-centric optimizations effectively reduce inference complexity, they inherently presume a fixed, full-dimensional input pipeline. Consequently, they neglect the dominant overheads of upstream data acquisition and transmission, which frequently bottleneck end-to-end host-side monitoring and offset any inference-stage efficiency gains.

In contrast, the proposed SG-LR framework overcomes this limitation by co-designing a sparse data acquisition pipeline (via AnnealGate) with a parameter-efficient recursive encoder (TS-LB). Under a unified online resource budget, this joint optimization prevents computational burden from shifting between monitoring stages, achieving a predictable, minimized total footprint for robust real-time QoS monitoring.

III. Details of the SG-LR QoS Detection Framework

The SG-LR QoS framework is customized for host-side online service monitoring under tight resource constraints. Two bottlenecks commonly co-occur in production deployments for cloud services: high-dimensional host signals introduce non-trivial acquisition overhead, and deep-stacked temporal encoders incur memory growth under multi-instance concurrency. A co-designed pipeline is established to jointly optimize the input-side footprint and the model-side inference overhead in a unified objective, maintaining service-level monitoring fidelity under varying workload conditions.

A. Network Structure of SG-LR QoS model

The network structure of our proposed SG-LR QoS model is illustrated in Fig. 1, which comprises three core components: AnnealGate, TrunkShare-LoRABank (TS-LB), and Soft Routing Fusion. AnnealGate first learns a structured sparse feature subset at the input source, reducing data acquisition overhead for cloud service monitoring. The gated sequence is then processed by TS-LB, which implements a recursive low-rank incremental encoding mechanism. Specifically, a shared trunk encoder layer is reused across recursive steps, while step-specific temporal features are captured by injecting low-rank incremental residuals through a LoRA Bank. Finally, Soft Routing Fusion aggregates the intermediate representations from all steps to produce the estimated service degradation ratio $\hat{\delta}_{\text{qos}}$.

B. AnnealGate: Automated Structured Feature Selection

AnnealGate applies an explicit, upstream selection layer to raw inputs to minimize data acquisition costs. Post-training, the learned gating vector is hardened into a deterministic binary mask ($\{0,1\}^d$), ensuring that non-activated features are omitted or constant-filled during deployment, directly reducing tangible I/O and bandwidth overhead.

Specifically, an input matrix $\boldsymbol{X} \in \mathbb{R}^{L\times d}$ is transformed into the gated matrix $\tilde{\boldsymbol{X}}$ via a feature-wise gating vector $\boldsymbol{z} \in [0,1]^d$ to enforce temporal consistency for hardware efficiency:

$$\tilde{\boldsymbol{X}} = \boldsymbol{X} \odot (\mathbf{1}_L \boldsymbol{z}^\top). \tag{1}$$

To enable gradient-based optimization for L_0 regularization, we adopt the Hard-Concrete distribution. The continuous gate

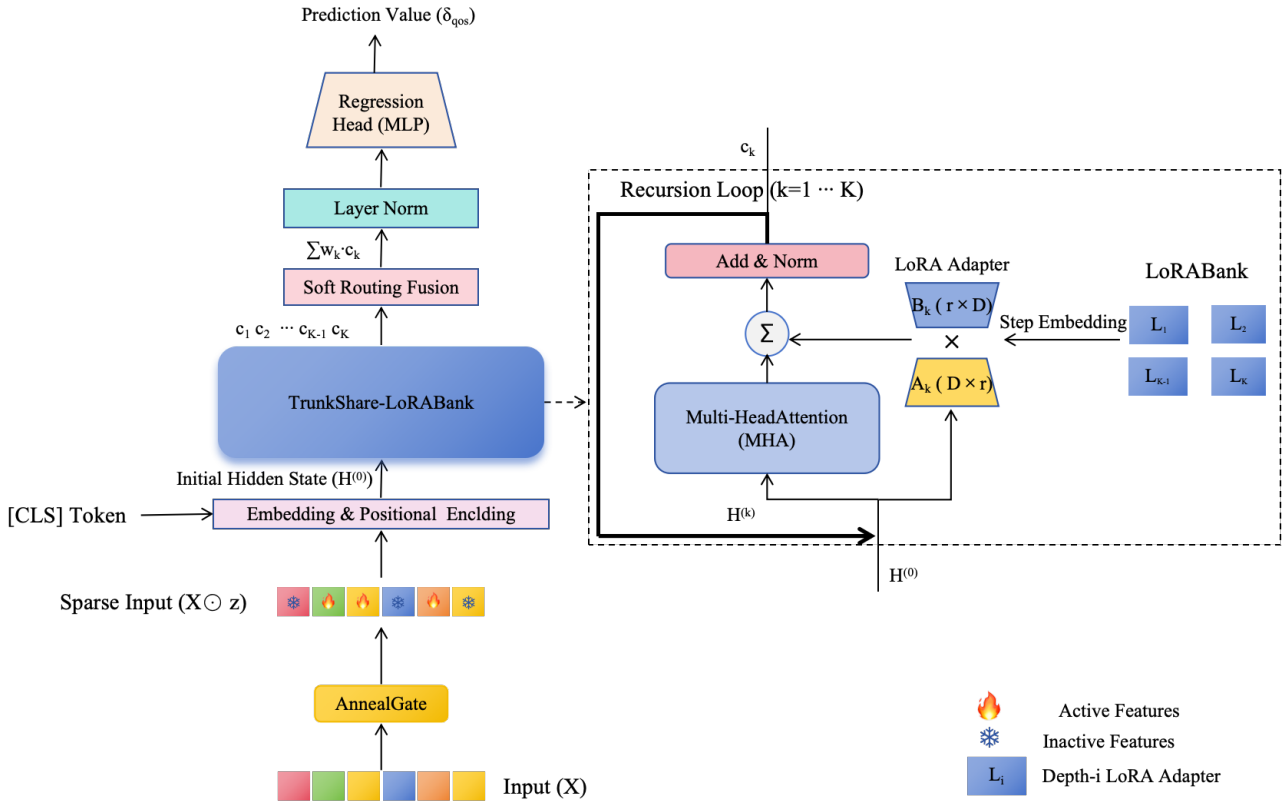

Fig. 1. The network structure of SG-LR, consisting of the AnnealGate for automated feature selection, the TrunkShare-LoRABank (TS-LB) for recursive representation learning, and the Soft Routing Fusion followed by a Regression Head for final prediction.

value z_i for each dimension i is derived via the stretch-and-clip transformation:

$$z_i = \min\left(1, \max\left(0, \sigma\left(l_i/\tau_t\right)\left(\zeta - \gamma\right) + \gamma\right)\right), \tag{2}$$

where $l_i = \log u_i - \log(1 - u_i) + \alpha_i$ is the stochastic logit ($u_i \sim \mathcal{U}(\varepsilon, 1-\varepsilon)$), and $\gamma < 0, \zeta > 1$ are stretching hyperparameters. The trainable parameters α_i are optimized via the sparsity penalty to approximate the expected L_0 norm and prune redundant dimensions:

$$\mathcal{L}_{\text{gate}} = \sum_{i=1}^{d} \sigma(\alpha_i - \tau_t \log(-\gamma/\zeta)). \tag{3}$$

C. TrunkShare-LoRABank (TS-LB) and Soft Routing Fusion

TS-LB replaces deep-stacked layers with a single shared trunk encoder $\mathcal{F}_\theta$ recursively reused over K steps to break the linear parameter scaling with depth. The target regression task tracks the QoS degradation ratio $\delta_{\text{qos},t}$ normalized against a baseline performance y_{base}:

$$\delta_{\text{qos},t} = 1 - \frac{y_t}{y_{\text{base}}}. \tag{4}$$

To supplement the expressivity of the shared trunk, a step-specific LoRA Bank injects low-rank incremental knowledge at each step k [28]–[30], updating states via Eq. (5):

$$\boldsymbol{H}^{(k)} = \mathcal{F}_\theta(\boldsymbol{H}^{(k-1)}) \oplus \Delta\boldsymbol{H}^{(k)}, \tag{5}$$

where $\oplus$ denotes element-wise addition with residual connection and layer normalization. The step representation is extracted from the classification token position: $\boldsymbol{c}^{(k)} = \boldsymbol{H}^{(k)}_{[\text{CLS}]} \in \mathbb{R}^D$.

Soft Routing Fusion dynamically aggregates these representations via learned weights w_k to yield the final representation $\boldsymbol{c}$:

$$w_k = \frac{\exp(g_\phi(\boldsymbol{c}^{(k)})/\tau_f)}{\sum_{j=1}^{K}\exp(g_\phi(\boldsymbol{c}^{(j)})/\tau_f)}, \quad \boldsymbol{c} = \sum_{k=1}^{K} w_k \boldsymbol{c}^{(k)}. \tag{6}$$

A Multi-Layer Perceptron (MLP) head maps $\boldsymbol{c}$ to the prediction $\hat{\delta}_{\text{qos}}$. To improve training robustness to monitoring noise, the framework optimizes the Huber loss:

$$\mathcal{L}_{\text{task}} = \frac{1}{B}\sum_{b=1}^{B}\kappa(\hat{\delta}_{\text{qos},b} - \delta_{\text{qos},b}),$$
$$\text{where } \kappa(e) = \begin{cases} \frac{1}{2}e^2, & |e| \le \delta \\ \delta|e| - \frac{1}{2}\delta^2, & |e| > \delta \end{cases}. \tag{7}$$

The global objective $\mathcal{L}_{\text{global}}$ jointly optimizes prediction accuracy, feature sparsity, and recursive stability:

$$\mathcal{L}_{\text{global}} = \mathcal{L}_{\text{task}} + \lambda_{\text{gate}}\mathcal{L}_{\text{gate}} + \lambda_{\text{fusion}}\mathcal{R}(\boldsymbol{w}), \tag{8}$$

where the training regularizer $\mathcal{R}(\boldsymbol{w})$ penalizes excessive deep dependencies and prevents weight collapse:

$$\begin{aligned}\mathcal{R}(\boldsymbol{w}) = \lambda_{\text{ponder}}\mathbb{E}_b\left[\sum_{k=1}^{K} k \cdot w_{b,k}\right] \\ + \lambda_{\text{bal}} K \sum_{k=1}^{K} \bar{w}_k \bar{f}_k - \lambda_{\text{H}}\mathbb{E}_b[\mathcal{H}(\boldsymbol{w}_b)].\end{aligned} \tag{9}$$

Here, $\bar{w}_k$ and $\bar{f}_k$ denote the average weight and dominance frequency of step k. This co-design yields high-capacity QoS modeling bounded within strict host-side resource budgets.

D. Training and Optimization Procedure

The joint training of the SG-LR QoS framework co-optimizes data efficiency and model compactness within a unified loop, as formalized in Algorithm 1.

Algorithm 1 Joint Training of SG-LR QoS

Input: $\{\boldsymbol{X}, \delta_{\text{qos}}\}$, depth K, coefficients $\{\lambda_{\text{gate}}, \lambda_{\text{ponder}}, \lambda_{\text{bal}}, \lambda_{\text{H}}\}$, initial τ_t, τ_f.
Output: Optimized parameters θ, ϕ, α, and LoRA Bank $\{\Delta\boldsymbol{H}^{(k)}\}_{k=1}^{K}$.

```
Initialize parameters θ, φ, α, and LoRA Bank;
while not converged do
    Sample a mini-batch of QoS signals (X, δ_qos);
    Phase 1: Generate gate z via (2) and obtain masked input X̃ = X ⊙ (1_L z^⊤) via (1);
    Phase 2: Set H^(0) = Embedding(X̃);
    for k = 1 to K do
        Compute H^(k) = F_θ(H^(k−1)) ⊕ ΔH^(k) via (5) and extract token c^(k) = H^(k)_[CLS];
    end for
    Phase 3: Compute weights w_k (6) and generate prediction δ̂_qos = MLP(Σ_{k=1}^K w_k c^(k));
    Phase 4: Minimize global loss L_global (8) (bundling L_task (7), L_gate (3), and R(w) (9));
    Update parameters via backpropagation, and decay gate temperature τ_t via cosine schedule;
end while
return Optimized SG-LR QoS model.
```

(1) *Structured Feature Selection (Phase 1):* AnnealGate samples a stochastic gating vector to mask non-informative input channels, enforcing structured feature pruning uniformly across temporal dimensions.

(2) *Recursive Forward Pass (Phase 2):* The gated input is iteratively refined by the shared trunk encoder and step-specific adapters from the LoRA Bank, capturing progressive temporal dependencies without parameter replication.

(3) *Soft Routing and Aggregation (Phase 3):* A lightweight routing head generates adaptive fusion weights via a temperature-scaled softmax to aggregate step-wise hidden representations for the regression prediction.

(4) *Global Optimization (Phase 4):* The framework co-optimizes prediction accuracy and structural sparsity by minimizing the global objective via backpropagation, paired with a cosine temperature annealing schedule.

E. Complexity and Parameter Efficiency

AnnealGate uses a fixed, sample-agnostic gate with only d trainable parameters and $\mathcal{O}(Bd)$ forward complexity, comparable to a linear layer, while eliminating redundant features before acquisition. TS-LB replaces stacked layers with a recursive shared trunk and low-rank LoRA increments, cutting parameters from $\mathcal{O}(KD^2)$ to $\mathcal{O}(D^2 + KrD)$. When rank $r \ll D$, increasing depth K yields only mild memory growth, enabling deep refinement under strict host-side budgets.

IV. Experimental Evaluation

This section presents a comprehensive evaluation of the proposed SG-LR QoS detection model. The experiments are designed to assess the framework's capability to achieve high-fidelity QoS degradation detection while operating within the stringent resource constraints of host-side production environments.

A. Datasets and Evaluation Setup

Experiments are conducted on a public production QoS dataset consisting of 13 cloud application types, such as etcd, MongoDB, Kafka and MySQL, spanning CPU-, I/O- and memory-heavy workloads. The input space consists of 71-dimensional host-level kernel metrics (CPU, memory, network, disk, cache) collected non-intrusively at 30-second intervals, Z-score normalized, and formatted via a sliding window of length $L = 16$. Predictive accuracy is assessed via Mean Squared Error (MSE), Mean Absolute Error (MAE), and the Coefficient of Determination (R^2), while operational efficiency is measured by parameter count, memory footprint, and inference latency. We compare SG-LR QoS against four state-of-the-art baselines: MTQoS [5], Informer [31], DLinear [32], and LimiX [33].

B. Experimental Setup

Datasets and Scenarios: We evaluate our model using a large-scale QoS monitoring dataset collected from a production cloud environment, covering diverse workloads such as web services and distributed databases [5]. The raw input space consists of 71-dimensional system metrics (e.g., CPU utilization, I/O wait, and network throughput). The labels represent the QoS degradation ratio δ_{qos} normalized against a pre-defined performance baseline.

Test Environment: To simulate resource-constrained host-side monitoring, all performance evaluations are conducted on an edge-level computing node equipped with a 0.5-core CPU and 2GB of Random Access Memory (RAM). This setup ensures that the recorded inference latency and memory footprint accurately reflect the model's operational efficiency in stringent production environments.

Implementation Details: The proposed framework is optimized via the Adam optimizer under a cosine annealing schedule for the AnnealGate temperature. All hyperparameter

configurations are empirically selected to strike an optimal balance between representation capacity and computational efficiency. A comprehensive summary of the training and architectural configurations is provided in Table I.

TABLE I
KEY HYPERPARAMETER AND TRAINING CONFIGURATIONS

Component	Hyperparameter	Value
Training	Epochs	100
	Batch size	512
	Random seed	$40 \sim 42$
Optimizer	Learning rate	1×10^{-4}
AnnealGate	Initial temperature (τ_t)	2.0
	Stretching factors (γ, ζ)	$-0.1, 1.1$
	Sparsity penalty (λ_{gate})	0.001
TS-LB	Max recursion depth (K)	4
	LoRA rank (r)	8
	Scaling factor (α_l)	16
Fusion	Temperature (τ_f)	0.5
	Regularization ($\lambda_{\text{ponder}}, \lambda_{\text{bal}}, \lambda_{\text{H}}$)	$0.005, 0.05, 0.05$
	Global weight (λ_{fusion})	0.01

C. Experimental Results

The main experimental results evaluated across multiple random seeds are summarized in Table II. In terms of predictive precision, only MTQoS and the proposed SG-LR QoS achieve an MAE within a 5% margin, satisfying the stringent accuracy requirements for production cloud environments. The R^2 scores further corroborate this, confirming their superior explanatory power over volatile QoS signals. Conversely, Informer, DLinear, and LimiX suffer from higher error rates and significantly degraded R^2 scores. This performance gap stems from data characteristics: cloud QoS signals exhibit transient, evolving correlations rather than stable long-range dependencies. Consequently, Informer's long-sequence alignment and LimiX's generalist structured modeling are highly susceptible to tracking noise under severe burstiness, while DLinear's linear decomposition fails to capture such complex, non-linear dynamics.

To evaluate deployment feasibility, operational efficiency is benchmarked against MTQoS under a resource-constrained host (0.5-core CPU, 2GB RAM). As shown in Fig. 2, the complete SG-LR QoS framework strikes a superior balance between accuracy and host-side efficiency, driven by the co-design of AnnealGate and TS-LB. Specifically, through the differentiable selection of AnnealGate, the framework isolates 42 essential features from 71 dimensions (40.85% sparsity). This allows deployment-stage agents to physically collect only this activated subset while bypassing the remainder, directly translating input sparsity into tangible reductions in host-side I/O and preprocessing overhead. Concurrently, TS-LB's recursive low-rank mechanism avoids layer replication, slashing parameters by 79.2% (from 260K to 54K) and memory footprint by 45.8% (from 794 MB to 430 MB). This lightweight pipeline yields a 2.75× speedup, dropping per-sample latency from 77 ms to 28 ms. These metrics confirm that SG-LR QoS achieves rapid, high-fidelity monitoring under strict host budgets.

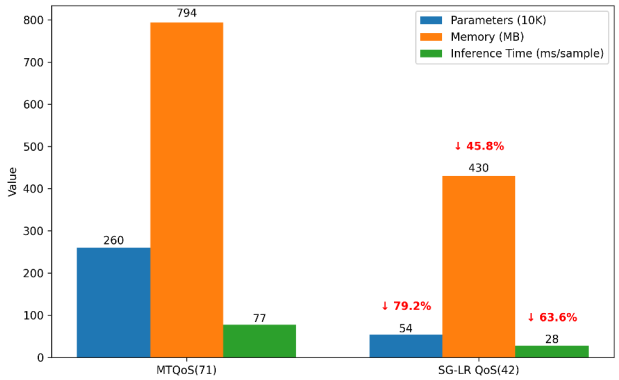

Fig. 2. Comparison of resource efficiency between MTQoS and SG-LR QoS. The red labels indicate the percentage reduction achieved by the proposed framework across parameters, memory, and inference latency.

In summary, while AnnealGate primarily minimizes upstream data acquisition costs and TS-LB empowers parameter-efficient capacity, the complete SG-LR QoS framework strikes a superior, balanced trade-off between predictive accuracy and host-side operational efficiency.

D. Ablation Study

To investigate the contribution of each component within the SG-LR QoS framework, we conduct systematic ablation experiments on the test split.

Sensitivity to LoRA Configuration: We first evaluate the framework's sensitivity to the LoRA rank r and its scaling factor α_l under a fixed recursion depth $K = 4$. As shown in Fig. 3, the model achieves the optimal balance between representational capacity and parameter efficiency at $r = 8$ and $\alpha_l = 16$, yielding the lowest MAE (0.0526) and MSE (0.0091). Reducing the rank to $r = 4$ ($\alpha_l = 8$) limits step-wise refinement and degrades performance (MAE 0.0532, MSE 0.0094). Conversely, inflating the rank to $r = 16$ ($\alpha_l = 32$) introduces redundant low-rank updates that interfere with the stable optimization of the shared trunk, leading to a slight performance decline.

Impact of Recursion Depth: Next, we examine the maximum recursion depth K to minimize computational and memory access overhead during inference. As illustrated in Fig. 4, $K = 4$ stands as the optimal operating point (MAE 0.0526, MSE 0.0091). Restricting recursion to $K = 2$ fails to fully capture higher-order temporal dynamics, raising the MSE to 0.0096. Meanwhile, extending the depth to $K = 6$ provides no further representational gains and slightly diminishes performance (MAE 0.0527, MSE 0.0093), validating that four steps offer the most efficient trade-off for host-side monitoring.

Component-level Analysis: To isolate module contributions, we evaluate three configurations in Table III. The complete framework integrates both AnnealGate and TS-LB, achieving the lowest MSE (0.0091) and highest R^2 (0.6812). Removing TS-LB while retaining AnnealGate triggers a severe performance drop (MSE spikes to 0.0118, R^2 plunges to 0.6032),

TABLE II
COMPARISON OF ACCURACY METRICS AMONG DIFFERENT MODELS

Metrics	SG-LR QoS	MTQoS	Informer	DLinear	LimiX
MAE	0.0523 ± 0.0026†	0.0515 ± 0.0012*	0.1227 ± 0.0045	0.0720 ± 0.0032	0.1026 ± 0.0068
MSE	0.0092 ± 0.0006†	0.0093 ± 0.0007*	0.0223 ± 0.0011	0.0139 ± 0.0007	0.0236 ± 0.0006
R^2	0.6731 ± 0.0288†	0.7183 ± 0.0112*	0.3951 ± 0.0241	0.5002 ± 0.0030	0.3667 ± 0.0149

*The best performance; †The second-best performance.

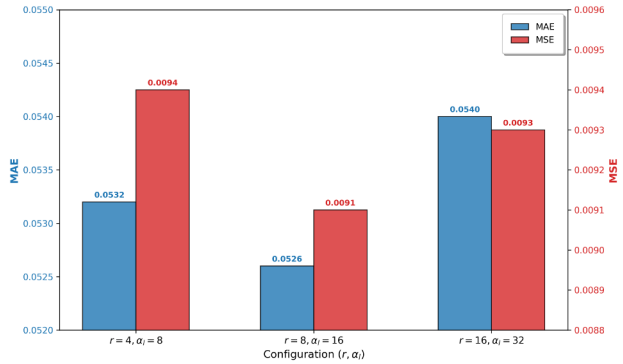

Fig. 3. Sensitivity Analysis of LoRA Rank (r) and Scaling Factor (α_l). Lower values indicate better performance.

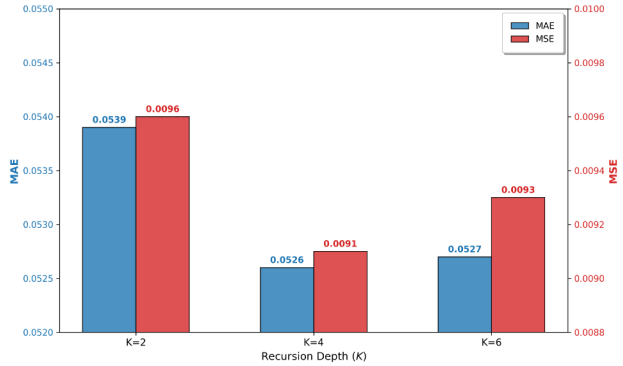

Fig. 4. Impact of Maximum Recursion Depth (K)

confirming that recursive low-rank increments are indispensable for modeling sudden QoS fluctuations.

Conversely, disabling AnnealGate to utilize all 71 features yields a lower average error (MAE 0.0414) but degrades prediction stability (MSE 0.0105) and variance explanation (R^2 0.6362). This highlights AnnealGate's strategic trade-off: pruning inputs to 42 features (40.85% sparsity) slightly alters the average error margin but substantially suppresses system noise. Crucially, this input-level sparsity demonstrates significant potential to minimize host-side I/O, bandwidth, and collection costs by allowing non-activated metrics to be bypassed or padded during deployment. In conclusion, AnnealGate and TS-LB are highly complementary, balancing deployment-stage feature collection costs with parameter-efficient, noise-robust QoS modeling.

TABLE III
COMPONENT-LEVEL ABLATION ANALYSIS

AnnealGate	TS-LB	MAE	MSE	R^2
+	-	0.0575	0.0118	0.6032
-	+	0.0414*	0.0105†	0.6362†
+	+	0.0526†	0.0091*	0.6812*

* Best performance; † Second-best performance.

V. CONCLUSIONS

This study presents SG-LR QoS, a lightweight framework for host-side QoS monitoring in resource-constrained clouds. By co-designing the AnnealGate for structured feature sparsification and the TS-LB recursive encoder with low-rank incremental adapters, our framework jointly addresses the dual challenges of data acquisition cost and inference overhead. Experiments show that SG-LR QoS maintains high predictive fidelity (MAE within 5% of the state-of-the-art MTQoS) while drastically reducing operational resource consumption, and achieves competitive variance explanation (R^2) for bursty QoS patterns. Specifically, AnnealGate prunes 40.85% of input features (from 71 to 42), enabling leaner data collection. Moreover, compared to MTQoS, our design achieves a 79.2% reduction in parameters, a 45.8% lower memory footprint, and a 2.75× inference speedup, validating its deployability in production environments. Future work will explore adaptive recursion depth and more efficient incremental structures to further balance accuracy and efficiency.

ACKNOWLEDGMENT

This work was supported by Guangdong Basic and Applied Basic Research Foundation (2025A1515011259).

DECLARATION

The large language model (i.e., Deepseek) was only used for language polishing in this paper. All research content, including the core idea, algorithm design, experiments, and analysis, was independently accomplished by the authors.

REFERENCES

[1] W. Cao, X. Tao, Y. Pan, Y. Liu, and Z. Ming, "QoS perception for cloud databases: Necessity, Trends, and Challenges," in *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*, 2024, pp. 1–2.

[2] X. Tao, W. Cao, Y. Pan, Y. Liu, and Z. Ming, "Improving qos of workloads with cpu pinning: A deep reinforcement learning approach," in *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*. IEEE, 2024, pp. 1–2.

[3] X. Tao, D. Lu, W. Cao, J. Gu, Z. Cai, C. Xu, L.-J. Zhang, and Z. Ming, "Qos-aware deep reinforcement learning for dynamic cpu pinning of co-located cloud workloads," *IEEE Transactions on Services Computing*, pp. 1–12, 2026.

[4] Z. Xiao, Y. Qiu, P. Liu, W. Cao, and Z. Ming, "Mts-cr: Contrastive representation learning for real-time qos degradation detection in media cloud shared instances," in *ICASSP 2026-2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2026, pp. 12 392–12 396.

[5] W. Cao, J. Gu, Z. Ming, Z. Cai, Y. Wang, C. Ji, Z. Xiao, Y. Feng, Y. Liu, and L.-J. Zhang, "Flexible computing: A new framework for improving resource allocation and scheduling in elastic computing," *IEEE Transactions on Services Computing*, vol. 18, no. 1, pp. 198–211, 2025.

[6] Y. Zheng, Y. Chen, B. Qian, X. Shi, Y. Shu, and J. Chen, "A review on edge large language models: Design, execution, and applications," *ACM Comput. Surv.*, vol. 57, no. 8, Mar. 2025.

[7] H. Wu, Z. Zhang, J. Luo, K. Yue, and C.-H. Hsu, "Multiple attributes QoS prediction via deep neural model with contexts," *IEEE Transactions on Services Computing*, vol. 14, no. 4, pp. 1084–1096, 2021.

[8] Y. Zhang, Z. Zheng, and M. R. Lyu, "Exploring latent features for memory-based QoS prediction in cloud computing," in *2011 IEEE 30th International Symposium on Reliable Distributed Systems*, 2011, pp. 1–10.

[9] J. Zhu, P. He, Z. Zheng, and M. R. Lyu, "Online QoS prediction for runtime service adaptation via adaptive matrix factorization," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 10, pp. 2911–2924, 2017.

[10] C. Yan, Y. Zhang, W. Zhong, C. Zhang, and B. Xin, "A truncated svd-based arima model for multiple qos prediction in mobile edge computing," *Tsinghua Science and Technology*, vol. 27, no. 2, pp. 315–324, 2021.

[11] Z. Ye, S. Mistry, A. Bouguettaya, and H. Dong, "Long-term qos-aware cloud service composition using multivariate time series analysis," *IEEE Transactions on Services Computing*, vol. 9, no. 3, pp. 382–393, 2014.

[12] W. Hussain, J. Merigó, M. Raza, and H. Gao, "A new QoS prediction model using hybrid iowa-anfis with fuzzy c-means, subtractive clustering and grid partitioning," *Information Sciences*, vol. 584, pp. 280–300, 2022.

[13] A. F. Ansari, L. Stella, A. C. Turkmen, X. Zhang, P. Mercado, H. Shen, O. Shchur, S. S. Rangapuram, S. P. Arango, S. Kapoor, J. Zschiegner, D. C. Maddix, H. Wang, M. W. Mahoney, K. Torkkola, A. G. Wilson, M. Bohlke-Schneider, and B. Wang, "Chronos: Learning the language of time series," *Transactions on Machine Learning Research*, 2024.

[14] Z. Yao, Z. Tang, W. Yang, and W. Jia, "Enhancing llm qos through cloud-edge collaboration: A diffusion-based multi-agent reinforcement learning approach," *IEEE Transactions on Services Computing*, vol. 18, no. 3, pp. 1412–1427, 2025.

[15] H. Liu, Z. Zhang, L. Sang, Q. Wu, and Y. Zhang, "Quality of service prediction via large language models," in *2025 IEEE International Conference on Web Services (ICWS)*, 2025, pp. 87–95.

[16] Z. Ding, D. Ding, H. Zhang, J. Cao, and G. Xue, "Llm4load: An llm prompt-driven framework for multi-scale workload prediction," in *2025 IEEE International Conference on Web Services (ICWS)*, 2025, pp. 293–304.

[17] Y. Cheng, X. Huang, Z. Liu, J. Chen, X. Gao, Z. Fang, and Y. Yang, "FEDGE: An interference-aware QoS prediction framework for black-box scenario in IaaS clouds with domain generalization," in *2024 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2024, pp. 128–138.

[18] R. A. Ghalehtaki, A. Ebrahimzadeh, and R. H. Glitho, "Context-aware feature selection using denoising auto-encoder for fault detection in cloud environments," in *2022 IEEE Cloud Summit*, 2022, pp. 83–90.

[19] N. Gui, D. Ge, and Z. Hu, "AFS: An Attention-Based mechanism for supervised feature selection," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 33, no. 1, 2019, pp. 3705–3713.

[20] T. Yasuda, M. Bateni, L. Chen, M. Fahrbach, G. Fu, and V. Mirrokni, "Sequential attention for feature selection," *arXiv preprint arXiv:2209.14881*, 2022.

[21] K. Han, Y. Wang, C. Zhang, C. Li, and C. Xu, "Autoencoder inspired unsupervised feature selection," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 2941–2945.

[22] L. Sun, M. Li, W. Ding, and J. Xu, "Agnsa: Adaptive graph learning-based unsupervised feature selection with non-convex sparse autoencoder," *Applied Soft Computing*, vol. 181, p. 113550, 2025.

[23] P. Wang, K. Wang, Y. Song, and X. Wang, "AutoLDT: a lightweight spatio-temporal decoupling transformer framework with AutoML method for time series classification," *Scientific Reports*, vol. 14, no. 1, p. 29801, 2024.

[24] L. Wang, Y. Li, H. Liu, and T. Liu, "A lightweight Transformer edge intelligence model for RUL prediction classification," *Sensors*, vol. 25, no. 13, 2025.

[25] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*, 2019.

[26] Y. Wu, H. Zhang, and Y. Zhang, "Stgnn-based microservice autoscaling and placement for fast qos recovery at the edge," in *2025 IEEE International Conference on Web Services (ICWS)*, 2025, pp. 892–898.

[27] R. Das and T. Luo, "Lightesd: Fully-automated and lightweight anomaly detection framework for edge computing," in *2023 IEEE International Conference on Edge Computing and Communications (EDGE)*, 2023, pp. 150–158.

[28] W. Zou, Z. Shen, Q. Li, J. Ge, C. Li, X. Chen, X. Shen, L. Huang, and B. Luo, "Experimental evaluation of parameter-efficient fine-tuning for software engineering tasks," vol. 34, no. 7, Aug. 2025.

[29] H. Liu, D. Tam, M. Mohammed, J. Mohta, T. Huang, M. Bansal, and C. Raffel, "Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning," in *Advances in Neural Information Processing Systems*, 2022.

[30] E. J. Hu, yelong shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *International Conference on Learning Representations*, 2022.

[31] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 35, no. 12, 2021, pp. 11 106–11 115.

[32] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are Transformers effective for time series forecasting?" in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 37, no. 9, 2023, pp. 11 121–11 128.

[33] X. Zhang, G. Ren, H. Yu, H. Yuan, H. Wang, J. Li, J. Wu, L. Mo, L. Mao, M. Hao *et al.*, "Limix: Unleashing structured-data modeling capability for generalist intelligence," *arXiv preprint arXiv:2509.03505*, 2025.