OceanBase Database Health Status Prediction System Based on the HPSA-LSTM Model

Kai Yan[a,*], Jingyu Jia[b], Lei Lv[c], Wanglong Han[d], Haitong Wu[e], Junpeng An[f]

PipeChina Digital Co., Ltd., Beijing, 100020, China
[a]kidyan2012@gmail.com, [b]jia_de163@163.com, [c]lvlei01@pipechina.com.cn, [d]mator_007@163.com, [e]wuht01@pipechina.com.cn, [f]569982258@qq.com
[*]Corresponding author

Abstract: *Databases serve as the core of enterprise data management, and their stable and efficient operation is directly tied to business continuity. Any failure or performance bottleneck can severely impact operations and decision-making. Therefore, ensuring database health is crucial for enterprise digitalization. To accurately and comprehensively evaluate database health, this paper proposes HPSA-LSTM (Host, Performance, SQL, Alert-based LSTM), a predictive framework for database health metrics based on four dimensions: host, performance, SQL, and alerts. The framework integrates multi-dimensional data such as performance indicators, security status, and resource utilization to compute a comprehensive score that reflects the current health state of the database. With this approach, operators can quickly grasp the overall health condition without delving into individual monitoring metrics, enabling timely identification and resolution of potential issues. Experiments on multiple database instances demonstrate that the proposed model achieves excellent performance in predicting overall database health.*

Keywords: *Intelligent Operation and Maintenance; Health Scoring; Status Prediction; Database Performance Monitoring*

1. Introduction

Against the backdrop of rapid global digital economic development, database health management has become a critical technology for ensuring data integrity, system stability, and operational security. According to IDC research, the global datasphere is projected to exceed 200ZB within the next five years[1]. Amid explosive data growth and the widespread adoption of emerging technologies such as cloud-native architectures and edge computing, establishing a database health management system powered by deep learning is essential for enhancing enterprises' digital competitiveness and driving high-quality economic growth.

However, current database health assessment faces two major challenges. First, most database systems lack a unified, standardized framework for quantifying health metrics, making it difficult to conduct systematic evaluations of performance degradation, resource utilization efficiency, and potential security risks[2]. Second, existing management practices are largely reactive—relying on manual, point-in-time interventions after failures occur—lacking both holistic analysis of end-to-end performance deterioration and the ability to establish cross-component preventive optimization mechanisms.

To address these challenges, this study proposes HPSA-LSTM, a database health prediction framework based on four dimensions—Host, Performance, SQL, and Alert—leveraging the LSTM neural network[3] to model and forecast database health trends. The framework first predicts various monitoring metrics collected from OceanBase databases[4], then converts them into health scores using a dedicated transformation algorithm. These score sequences are fused through a sequence fusion module, and finally, a comprehensive health score reflecting the system's overall state is generated by the health aggregation module. Experimental results on multiple database instances demonstrate that the proposed model achieves excellent performance in predicting overall database health.

2. Methods

2.1 Overall Model Architecture

Figure 1 illustrates the proposed HPSA-LSTM (Host, Performance, SQL, Alert-based LSTM) framework for database health indicator trend prediction, which is based on host, performance, SQL, alert, and LSTM. The framework first collects multiple monitoring indicators from the OceanBase, including CPU usage, disk usage, and QPS. These raw data are then preprocessed using an algorithm. Secondly, the preprocessed indicator sequences are input into the LSTM prediction module and the indicator conversion module respectively for sequence prediction and conversion operations. Subsequently, the sequence fusion module performs feature fusion on these sequences. Finally, the health aggregation module integrates the various results and outputs the comprehensive health score of the database.

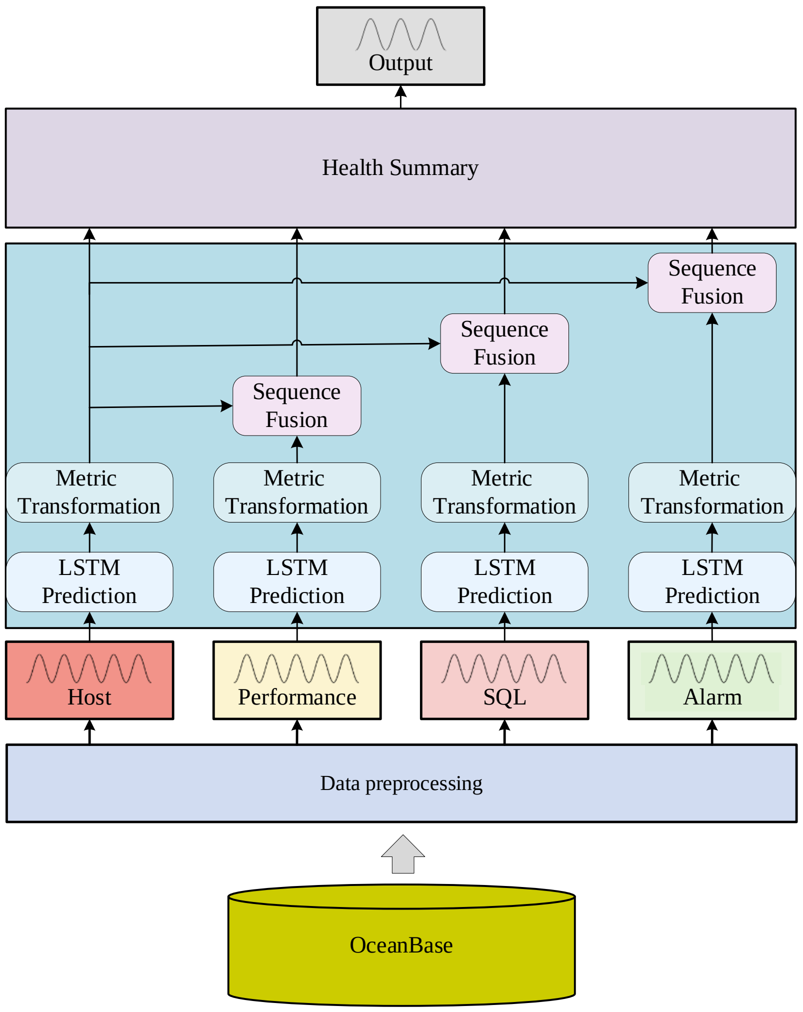

Figure 1: The framework of HPSA-LSTM.

2.2 Data Preprocessing

2.2.1 Data Acquisition

After multiple version iterations, OceanBase has accumulated a rich set of monitoring indicators in terms of performance and resources. However, to date, there remains a lack of a unified evaluation method that can comprehensively reflect the overall operational status.

Based on existing monitoring data and combined with our long-term practical experience in database operation and maintenance, we have designed a new database health assessment strategy. As shown in Table 1, this strategy collects multiple key monitoring indicators—including CPU utilization and disk utilization—from four dimensions (host, performance, SQL, and alerts) for two resource types: clusters and tenants.

In addition, we have configured performance and resource monitoring in the database monitoring platform, setting the sampling frequency of the key indicators mentioned in the table to once per minute. The system automatically synchronizes the collected data to the monitoring storage module to support subsequent analysis and evaluation.

Table 1: OceanBase indicator collection items.

Scoring categories	Resource type	Indicator
Host score	Cluster	CPU utilization
		Disk utilization
Performance score	Cluster	Data disk utilization
		Log disk utilization
	Tenant	Memory utilization
	Cluster / Tenant	Queries Per Second
		SQL response time
		Transactions Per Second
		Transaction response time
		Number of active sessions
SQL score	Cluster / Tenant	Number of slow SQL statements
		Number of suspicious SQL statements
		Number of high-risk SQL statements
Alert score	Cluster	Alert level

2.2.2 Outlier Detection and Treatment

Outliers tend to significantly alter the overall pattern of time series, thereby adversely affecting the accuracy of trend prediction. This paper adopts the classic box plot method to identify outliers in the data. Let the original time series be $\{x_t\}_{t=1}^{T}$, where the sequence $x_t \in \mathrm{R}$ represents the value at time t. We define the set of normal values as:

$$O = \{x_t \mid x_t \in [Q_1 - 1.5 \cdot IQR, Q_3 + 1.5 \cdot IQR]\} \tag{1}$$

Where Q_1 and Q_3 are the first and third quartiles of the data, respectively, and $IQR = Q_3 - Q_1$ is the interquartile range. If a sequence value falls outside this interval, it is considered an outlier.

After detecting outliers, these values are set as missing values, which are then interpolated to ensure data integrity and continuity. To address the issue of consecutive missing values in time series, this paper employs a segmented Lagrange interpolation method based on a sliding window for interpolating such missing values. Let the half-width of the window be w; for a missing segment$[a, b]$, valid data from w time points each in the forward and backward directions are selected to form a support set:

$$S = \{(t_j, x_{t_j}) \mid t_j \in [[a - w, a) \cup (b, b + w], x_{t_j} \in \mathrm{R}\} \tag{2}$$

Then, a unified interpolation polynomial $P(t)$ is constructed over the entire missing interval $[a, b]$ to fit the overall trend of this interval. This interpolation process can be expressed as:

$$P_i(t) = \sum_{j=1}^{n} x_{t_j} \prod_{1 \le k \le n, k \ne j} \frac{t - t_k}{t_j - t_k}, \quad t \in [a, b] \tag{3}$$

2.3 LSTM Prediction

Figure 2 illustrates the structure of an N-layer LSTM. Compared with traditional RNNs [5], LSTM, by introducing a gating mechanism, largely alleviates the problems of gradient vanishing and gradient exploding that may occur during training [6]. This enables the model to retain and transmit key information over a longer time horizon while remaining sensitive to temporal changes.

Although the hidden state finally output by the LSTM already contains the temporal information of the sequence, these features are usually high-dimensional, abstract, and unnormalized representations, which are not well-suited for the final prediction task. Therefore, we append a dense layer after the LSTM output to perform non-linear transformations on the outputs of the hidden units, thereby extracting higher-level feature combinations.

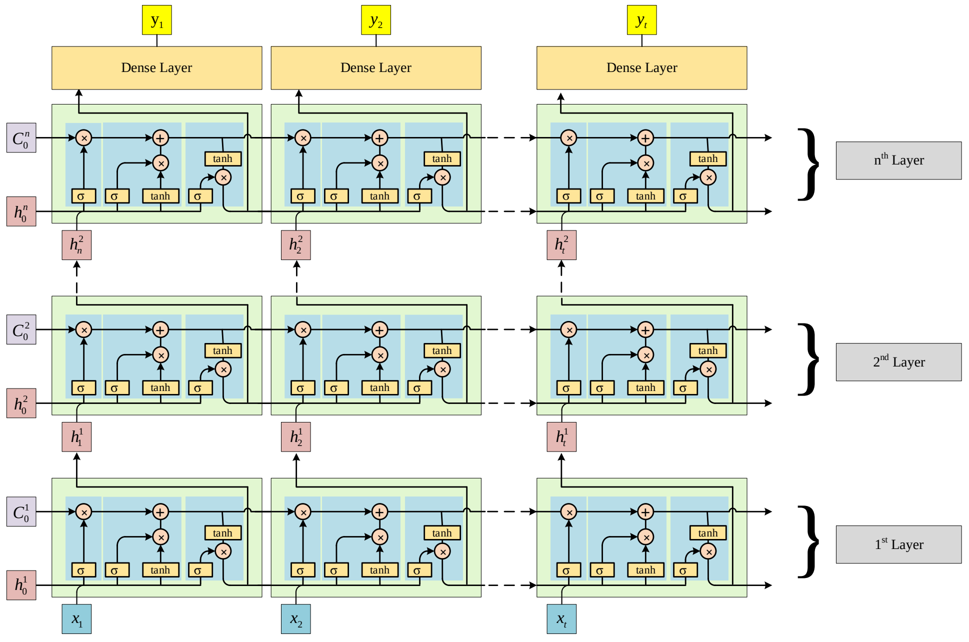

Figure 2: Structure of the K-layer LSTM recurrent neural network.

2.4 Monitoring Metric Transformation

After obtaining all the predicted monitoring metrics, we calculate the host score, performance score, SQL score, and alarm score respectively according to different rules.

2.4.1 Host Score

We have selected two key indicators: CPU usage and disk usage, which are used to measure the health status of the host. For the sampled CPU usage and disk usage, denoted as M_{cpu} and M_{disk} respectively, the scores for CPU usage and disk usage are defined as follows:

$$S_{cpu} = \begin{cases} 100, & if\ M_{cpu} < 70\% \\ 100 - \left[1 + \left(M_{cpu} - 70\%\right) \cdot 20\right], & if\ 80\% \geq M_{cpu} \geq 70\% \\ 100 - \left[10 + \left(M_{cpu} - 80\%\right) \cdot 30\right], & if\ M_{cpu} > 80\% \end{cases} \tag{4}$$

$$S_{disk} = \begin{cases} 100, & if\ M_{disk} < 80\% \\ 100 - [1 + (M_{disk} - 80\%) \cdot 30], & if\ 90\% \geq M_{disk} \geq 80\% \\ 100 - [15 + (M_{disk} - 90\%) \cdot 50], & if\ M_{disk} > 90\% \end{cases} \tag{5}$$

The final host score can be calculated as follows:

$$S_H = S_{cpu} \times 50\% + S_{disk} \times 50\% \tag{6}$$

2.4.2 Performance Score

Table 2: The threshold coefficients and scoring functions of different indicators.

Indicator	Normal threshold	Abnormal threshold	Function
QPS	0.5	0.7	f_1
TPS	0.5	0.7	f_1
Number of active sessions	0.5	0.7	f_1
SQL response time	1.5	2	f_2
Transaction response time	1.5	2	f_2

The performance score mainly includes 8 indicators: data disk usage, log disk usage, memory usage, QPS, SQL response time, TPS, transaction response time, and number of active sessions. Among them, the calculation rules for data disk usage, log disk usage, and memory usage are consistent with Equation (5). The remaining indicators are calculated for two types of resources: clusters and tenants. Due to differences in the characteristics of the indicators, for example, a higher QPS often indicates better performance, while the opposite is true for SQL response time, we define two types of calculation functions:

$$f_1(x, x_h, t_1, t_2) = \begin{cases} 100, & if\ x < x_h \cdot t_1 \\ 90, & if\ x_h \cdot t_1 \le x \le x_h \cdot t_2 \\ 80, & if\ x > x_h \cdot t_2 \end{cases} \tag{7}$$

$$f_2(x, x_h, t_1, t_2) = \begin{cases} 80, & if\ x < x_h \cdot t_1 \\ 90, & if\ x_h \cdot t_1 \le x \le x_h \cdot t_2 \\ 100, & if\ x > x_h \cdot t_2 \end{cases} \tag{8}$$

Where x represents the current sampled value of the current indicator, x_h represents the mean value of the historical values of the current indicator, and t_1 and t_2 represent the normal threshold coefficient and abnormal threshold coefficient respectively. The threshold coefficients of each indicator are shown in Table 2.

According to the above evaluation scheme, the scores of various performance evaluation indicators listed in Table 1 can be calculated. Finally, the arithmetic mean of all indicator scores is taken as the overall system performance score S_p, which will be used in the subsequent model training process.

2.4.3 SQL Score

We collect three monitoring indicators, namely the number of slow SQLs, the number of suspicious SQLs, and the number of high-risk SQLs, on two resource types: clusters and tenants. Assuming the number of a certain type of SQL is S_c, the score value of a single monitoring item is defined as:

$$S_{sql} = \begin{cases} 50,\ if\ \ S_c > 100 \\ 70,\ if\ \ 100 \ge S_c \ge 50 \\ 90,\ if\ \ S_c < 50 \end{cases} \tag{9}$$

According to Equation (9), the slow SQL score S_{slow}, suspicious SQL score S_{susp}, and high-risk SQL score S_{high} can be calculated. Finally, the comprehensive SQL score is calculated as follows:

$$S_S = \frac{S_{slow}^{culster} + S_{susp}^{culster} + S_{slow}^{culster}}{3} \times 50\% + \frac{S_{slow}^{tenant} + S_{susp}^{tenant} + S_{slow}^{tenant}}{3} \times 50\% \tag{10}$$

2.4.4 Alarm Score

In our database system, alarm information is divided into five levels: Critical, Severe, Warning, Low, and Hint, represented by C, S, W, L, and H respectively. Suppose there are n alarm items in the system, and each alarm item $i \in \{1,2,\dots,n\}$ has a corresponding alarm level r_i. The deduction weights corresponding to each alarm level are defined as follows:

$$S_{r_i} = \begin{cases} 1, & if\ \ r_i = H \\ 2, & if\ \ r_i = L \\ 4, & if\ \ r_i = W \\ 8, & if\ \ r_i = S \\ 10, & if\ \ r_i = C \end{cases} \tag{11}$$

Next, define the total risk level:

$$R_{total} = \max(r_1, r_2, \dots, r_n) \tag{12}$$

Where the priority relationship among the risk levels of each alarm in the max operation is:

$$C > S > W > L > H \tag{13}$$

The score range to which the score belongs can be determined according to the total risk level R_{total}. The score ranges corresponding to the 5 risk levels are defined as shown in Table 3. The calculation of the alarm score must meet the minimum score limit, that is, it shall not be lower than the lower limit of the score range corresponding to the current total risk level.

Let $[L_R, U_R]$ denote the score range corresponding to the total risk level R, where L_R is the lower limit and U_R is the upper limit. The final alarm score can be calculated as follows:

$$S_A = \max(L_R, U_R - \sum_{i=1}^{n} w(r_i, R) \cdot S_{r_i}) \tag{14}$$

Where $w(r_i, R)$ represents the deduction of the i-th alarm under the condition of risk level R:

$$w(r_i, R) = \begin{cases} S_{r_i}, & if\ r_i = R \\ 0, & otherwise \end{cases} \tag{15}$$

2.5 Sequence Fusion

We evaluate database health through various indicators involved in four types of scores: host, performance, SQL, and alarm. Considering that the host score sequence contains two key pieces of information in the OceanBase database, namely CPU and disk, which may affect other score sequences, we propose a sequence fusion module to fuse the host score sequence with other sequences.

For input time sequences $S_1 = \{\alpha_1, \alpha_2, \dots \alpha_T,\}$ and $S_2 = \{\beta_1, \beta_2, \dots \beta_T,\}$ with length T, we design a dynamic fusion module based on learnable weights. This module can adaptively extract useful information from the two input sequences and perform weighted combination at each time step. For each time step $t \in [1, T]$, the normalized weights of S_1 and S_2 are calculated respectively:

$$W_t^1 = \sigma(W \cdot [S_1 || S_2] + b) \tag{16}$$

$$W_t^2 = 1 - W_t^1 \tag{17}$$

Where σ is the sigmoid activation function, $W \in \mathrm{R}^{2\mathrm{T}}$ and $b \in \mathrm{R}$ are learnable parameters. The symbol $||$ represents the vector concatenation operation. The computed W_t^1 and W_t^2 are regarded as the importance assignments to the time sequences S_1 and S_2 at time step t. According to Equation (16) and Equation (17), the two time sequence can be fused using W_t^1 and W_t^2 as weights:

$$\mathrm{Y} = \mathrm{W}_\mathrm{t}^1 \odot \mathrm{S}_1 + \mathrm{W}_\mathrm{t}^2 \odot \mathrm{S}_2 \tag{18}$$

Table 3: Alarm risk levels and corresponding score ranges.

Risk level	Score range
Critical	[0,40)
Severe	[40,60)
Warning	[60,80)
Low	[80,90)
Hint	[90,100]

2.6 Health Summary

After obtaining the four categories of scoring results, we perform a weighted summation to calculate the final comprehensive score. We define the set of scoring weights as $W = \{W_H, W_P, W_S, W_A\}$, where W_H, W_P, W_S, and W_A are used to weight the host score S_H, performance score S_P, SQL score S_S, and availability score S_A, respectively.

It is important to note that the weights defined above are learnable parameters, initialized uniformly to 0.25. During the training process, these weights are continuously optimized and updated based on feedback from the loss function, enabling the model to automatically learn the varying impacts of each score type on the final result.

To ensure that the weighted comprehensive score falls within a reasonable output range, we apply the Sigmoid function in the final stage to perform a nonlinear mapping on the output. This generates a health score that aligns with human cognitive habits. This process can be expressed as:

$$health = Sigmoid(W_H \cdot S_H + W_P \cdot S_P + W_S \cdot S_S + W_A \cdot S_A) \tag{19}$$

3. Experimental Results and Analysis

3.1 Experimental Setup

In this study, we designed a three-layer LSTM architecture with 60 recurrent neurons per layer, incorporating a dense output layer for prediction. Data was collected from five database instances at 1-minute intervals, preprocessed, and split into 80% training and 20% testing sets. A dense layer appended to the hidden layer performed nonlinear transformations, while Dropout (0.3 probability) was applied to mitigate overfitting. The HPSA-LSTM model's performance was evaluated using Mean Squared Error (MSE), which quantifies prediction accuracy by measuring deviations between predicted and actual values.

During the model training process, we set the number of training epochs to 400 and the initial learning rate to 1e-3. To enhance training effectiveness, we employed the cosine annealing strategy [7] to dynamically adjust the learning rate. As for the optimizer, we selected the Adam algorithm [8], proposed by Kingma et al. This algorithm is an adaptive learning rate optimization method that demonstrates superior convergence performance and stability compared to traditional stochastic gradient descent in practical applications.

3.2 Model Prediction Performance

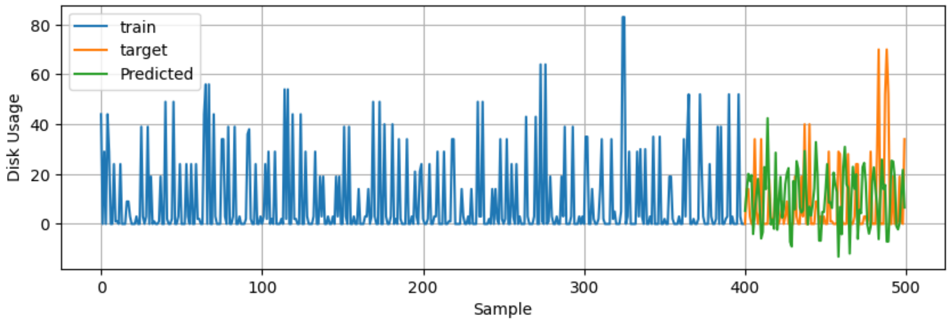

Figure 3: Disk Usage Sequence Prediction.

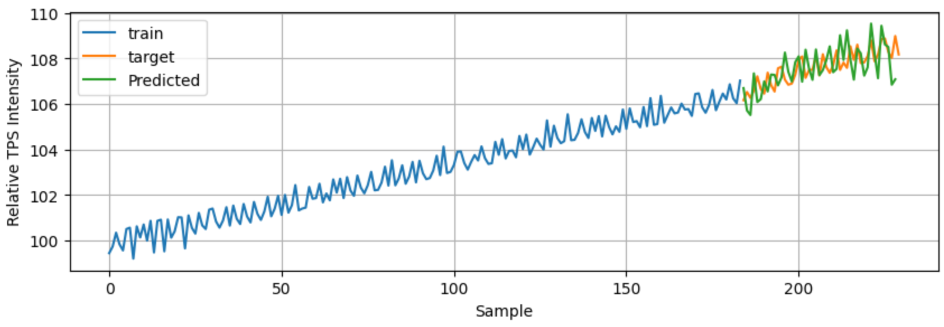

Figure 4: TPS Sequence Prediction.

Figure 3 and 4 illustrate the model's prediction performance using disk usage rate curves and TPS curves as examples, respectively. In these figures, the blue curve represents the training data sequence, the orange curve denotes the prediction target, and the green curve indicates the model's predicted results. As can be seen from the figures, the model demonstrates a good fit to the overall data trends. However, it is noteworthy that when predicting TPS, the model produced negative output values. Such a phenomenon is unreasonable in real-world scenarios. Therefore, in subsequent processing, we truncated the negative values in the model's predicted results, setting them uniformly to 0 to ensure the normal operation of subsequent processes and the rationality of the results.

3.3 Comparison of Sampling Intervals and Model Performance

The original data were sampled at 1-minute intervals, but the high sampling frequency obscured the temporal trends. Therefore, as shown in Figure 5 (a), the data were resampled to construct training sets with intervals of 15, 30, 45, and 60 minutes, in order to evaluate the impact of different sampling periods on modeling performance. Experimental results indicate that the HPSA-LSTM model exhibits stable performance across various database instances; however, when the sampling interval increases to 60 minutes, the prediction error rises significantly, suggesting that excessively long sampling periods may lead to the loss of critical temporal information. In contrast, the model achieves lower and more stable errors with a 45-minute interval. Hence, a 45-minute sampling interval is adopted as the standard for subsequent experiments.

Based on data from five database instances, this paper selects three typical time series forecasting models—SVR [9], Prophet [10] and ARIMA [11]—for comparative analysis. As illustrated in Figure 5(b), HPSA-LSTM outperforms all three models in prediction accuracy across all datasets, with average

MSE reductions of 10.80%, 2.20%, and 9.40% compared to SVR, Prophet, and ARIMA, respectively, fully demonstrating the effectiveness of HPSA-LSTM.

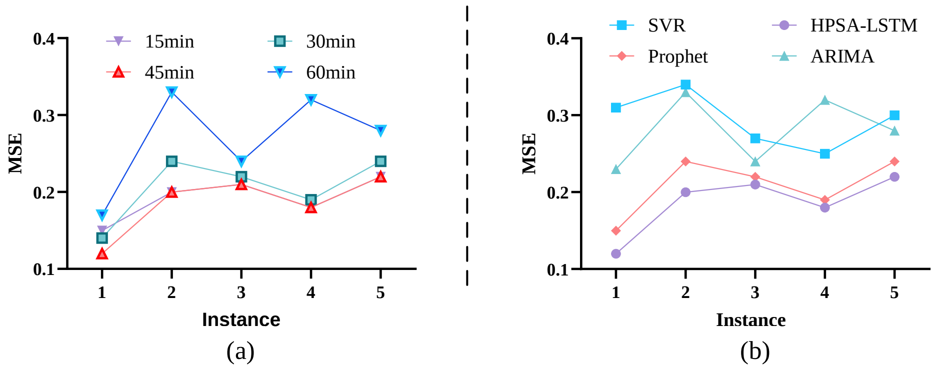

Figure 5: Comparison of different sampling intervals (a) and performance comparison of different models (b).

4. Conclusions

This paper proposes an HPSA-LSTM-based method for database health assessment and prediction, applied to OceanBase. The model integrates historical data with real-time metrics to predict operational trends, proactively identify performance bottlenecks and risks, and provide timely warnings before anomalies occur, significantly improving system stability. This approach has promising applications in database O&M management, enhancing enterprise efficiency and decision-making.

Future work will explore integrating large language models into health management. We aim to enable automated risk mitigation recommendations through precise risk identification, creating an intelligent mechanism to advance database O&M toward autonomous operations. This will achieve higher automation levels and intelligent decision-making capabilities in database systems.

References

[1] P. Fantini, "Memory technology enabling future computing systems," APL Mach. Learn., vol. 3, no. 2, 2025.
[2] Q. Xu, C. Yang, and A. Zhou, "Native Distributed Databases: Problems, Challenges and Opportunities," Proc. VLDB Endow., vol. 17, no. 12, pp. 4217–4220, 2024.
[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997.
[4] Z. Yang et al., "OceanBase: a 707 million tpmC distributed relational database system," Proc. VLDB Endow., vol. 15, no. 12, pp. 3385–3397, 2022.
[5] J. L. Elman, "Finding structure in time," Cogn. Sci., vol. 14, no. 2, pp. 179–211, 1990.
[6] C. Qin, L. Chen, Z. Cai, M. Liu, and L. Jin, "Long short-term memory with activation on gradient," Neural Netw., vol. 164, pp. 135–145, 2023.
[7] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," ArXiv Prepr. ArXiv160803983, 2016.
[8] D. P. Kingma, "Adam: A method for stochastic optimization," ArXiv Prepr. ArXiv14126980, 2014.
[9] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," IEEE Intell. Syst. Their Appl., vol. 13, no. 4, pp. 18–28, 1998.
[10] S. J. Taylor and B. Letham, "Forecasting at scale," Am. Stat., vol. 72, no. 1, pp. 37–45, 2018.
[11] D. K. Yadav, K. Soumya, and L. Goswami, "Autoregressive integrated moving average model for time series analysis," in 2024 International Conference on Optimization Computing and Wireless Communication (ICOCWC), IEEE, 2024, pp. 1–6.