Scholar Publishing Group
Socio-Economic Statistics Research
https://doi.org/10.38007/SESR.2026.070106
ISSN 2790-2722 Vol. 7, Issue 1: 48-57

A GPU-Parallel Optimization Method for Financial Time Series Similarity Linkages: A Study Based on a Collaborative Mechanism of Batch Distance Calculation and Candidate Filtering

Huilin Zhang

Guangzhou College of Commerce, School of Information Technology & Engineering, Guangzhou, 511363

Keywords: GPU parallel computing; financial time series; similarity link query; batch distance calculation; candidate filtering

Abstract: In scenarios such as high-frequency trading, asset allocation, and risk linkage identification in finance, there is an increasing need to leverage the similarity between financial time series for querying. However, traditional CPU architectures easily encounter computationally and memory-intensive problems when handling large-scale sliding windows, distance matrix creation, and candidate verification processes. Based on three years of research on GPU parallel computing, time series distance calculation, and financial sequence similarity modeling, this paper proposes a collaborative optimization scheme for batch query execution. Window-level standardization, batch distance calculation, shared memory caching, candidate filtering, and result compression and write-back are used to achieve hierarchical parallelism of the query path. Using real financial samples of Apple's historical daily closing price series, statistical analysis and connectivity metrics are conducted. Candidate size and distance distribution under different window lengths are constructed to analyze the necessary conditions for thread bundle division, memory access continuity, and kernel function fusion in GPU mapping. Research shows that financial sequence connectivity optimization cannot be achieved simply by speeding up a single operator; rather, it requires a combined design of data layout, threshold pruning, and batch processing strategies. This research can provide engineering references for similar event retrieval, pattern tracking, and risk propagation identification in quantitative finance.

1 Introduction

In studies such as algorithmic trading, abnormal fluctuation identification, cross-market linkage analysis, and asset style migration, researchers often encounter massive amounts of financial time

series and seek the closest historical period. Such operations are essentially similarity join queries, which find window pairs that satisfy similarity relationships from two or more time series sets under a given distance threshold or Top-k constraints. Due to the characteristics of financial series such as strong noise, local asynchrony, fluctuation clustering, and distribution drift, simple point-by-point comparison is difficult to be accurate and scalable. Therefore, high-dimensional sliding windows and elastic distance methods are widely used [1][3].

However, as the accumulated data volume in minutes, seconds, and even milliseconds grows exponentially, the computational scale of join queries increases dramatically. For each query window, standard processing, candidate enumeration, distance accumulation, and threshold determination are required, which results in many repeated accesses and arithmetic operations. In a business environment with multiple assets, multiple window lengths, and large-scale concurrent queries, although the CPU pipeline can complete complex control logic, it has inherent defects in data parallelism and bandwidth utilization [1][2]. The number of concurrent threads and memory bandwidth of GPUs are larger than those of CPUs, thus providing a new way to accelerate similar joins.

Therefore, this paper takes financial time series similarity join query as the research object, and combines the relevant research results of GPU join processing, time series distance calculation and financial sequence similarity representation in recent years [1]-[10]. It analyzes the algorithm structure, data layout and hardware mapping from three aspects, and proposes an optimization idea suitable for engineering implementation. This paper unfolds with the structure of introduction, current situation analysis, problem statement, problem solution or strategy, and conclusion.

2. Current Status Analysis of the Research Topic

In the past three years, GPUs have played a very important role in optimizing data processing and database operators. Wu et al. proposed optimization techniques such as GFTR for problems such as joins and grouping aggregation on GPUs. They believed that random access and result materialization overhead were the main factors causing low efficiency of GPU joins, and achieved good results with a workload-aware strategy [1]. This conclusion is also very important for financial similarity joins, as window joins involve many intermediate result write-backs and candidate pair materializations.

S. Wang et al. mapped k-Shape clustering to GPU under shared memory, proving the advantages of shared memory reuse, intra-block reduction, and batch parallel processing in time-series similarity computation [2]. Zhang, Huang, and Chen proposed a new MASST algorithm based on distance profile and gave the implementation framework of the algorithm on time series, thereby improving and accelerating the window-level method of distance computation. d'Hondt et al. made a structured evaluation of multivariate time series distance from more distance metric perspectives, believing that different distance functions, normalization methods, channel-related models, etc. will lead to different results.

From the perspective of GPU native indexing and similarity retrieval, the GTS proposed by Zhu et al. uses GPU-oriented tree indexing and batch querying methods to achieve orders-of-magnitude performance improvement in a general metric space [5]. It can be seen that the optimization of similarity queries cannot be accomplished solely by vectorization at the kernel function level, but also requires integrated design of various aspects such as index organization, task batching, and device memory management. A review of time series classification and regression related to financial time series tasks also points out that research in similarity pattern recognition, polymorphic feature extraction, and distance learning continues to advance.

On the financial application side, Fan et al. proposed a multi-factor dynamic time series similarity measure to improve the explanatory power of stock correlation [7]; Long et al. used Gramian Angular Field to improve the selection of similarity function in the financial time series transfer learning process [8]; Xiao et al. proposed a retrieval enhancement framework, believing that the recall quality of historical fragments plays an important role in the prediction of subsequent models [9]; Naivasha et al. studied the consistency problem of financial time series representation and comparison from the perspective of data quality and multi-indicator verification [10]. The above studies also show that the "similarity" of financial time series cannot be reflected by Euclidean distance alone. It includes a complex relationship of trend, volatility, scale, local structure and other aspects.

3. Raise questions

Existing research indicates that while progress has been made in GPU database operator optimization and time series similarity search, there are three gaps in specialized research on similarity join queries for financial time series. First, the database field emphasizes general optimization of join operators, but financial time series joins are characterized by severe sliding window overlap, dense distance calculations, and frequent threshold filtering triggers, making traditional join models unsuitable. Second, time series research prioritizes the accuracy of distance functions but neglects the varying impacts of different distance functions on GPU memory access patterns. Third, financial application research focuses on improving predictive performance, with limited discussion of the throughput, latency, and resource utilization of the underlying query system.

Financial time series join queries also incur easily overlooked costs such as window standardization costs, data partitioning and parallel processing costs, and system load balancing costs. Calculating the mean and variance for each window individually leads to numerous redundant reads and writes. The second cost is candidate pair enumeration. While the number of sliding windows is linearly related to the sequence length, the number of candidate pairs increases approximately quadratically. The third cost is cumulative distance costs. Even using Euclidean distance, each window pair requires m-dimensional difference and reduction; more complex distance methods incur even higher costs. The fourth cost is result materialization. Writing a large number of intermediate candidates directly back to global memory significantly reduces the device's effective throughput.

Therefore, the main problem this paper aims to solve is to organize window construction, standardization, candidate filtering, distance calculation, and result output into a batch process that can be processed by a GPU, while ensuring the semantic stability of financial time series similarity join queries. This reduces random access, minimizes unnecessary candidate validation, and improves device-level concurrency efficiency. This paper proposes systematic solutions to these problems from four aspects: data representation, distance calculation, thread mapping, and memory organization.

Let the financial price series be P = {P_t}_{t=1}^{T}. To mitigate the impact of different price levels on the connection, P is transformed into a logarithmic return series.

$$rt = \ln(Pt/Pt-1) \tag{1}$$

Equation (1) transforms the original price into a profit space, which can reduce the difference in scale and highlight the situation of local fluctuations. Logarithmic returns have good additivity in financial time series processing and are suitable for subsequent windowing.

For a window of length m, X_i=[r_i,r_{i+1},...,r_{i+m-1}], z-score normalization is used to eliminate mean bias and scale differences:

$$\hat{x}_{i,k}=\frac{x_{i,k}-\mu_i}{\sigma_i+\varepsilon}\ ,\;k=1,\ 2,...\,m\ \quad (2)$$

In Equation (2), the mean of the i-th window is represented by μ_i, the standard deviation by σ_i, and the numerical stability term by ε. Only through standardized windows can we have the basic conditions to conduct similarity comparisons of different stages and different degrees of fluctuation.

The basic distance between two normalized windows can be expressed as:

$$d(Xi,Xj)=\sqrt{}\sum k=1m(xi\hat{}\diagup\backslash k-x\hat{}kj)\hat{}2 \quad (3)$$

Equation (3) gives the most basic window distance model. Although financial sequences can be further supplemented with elastic distance or multi-factor distance, Euclidean distance is still the standard method that is easiest to parallelize in GPU batch implementation, and it is also the basic form that facilitates vectorized accumulation and intra-block reduction.

At a threshold τ, the similarity link result set is defined as:

$$J_{\tau}=\{(i,j)\mid d(X_i,X_j)\leq \tau,\; |ij|>w\} \quad (4)$$

In equation (4), the bandwidth w is excluded to prevent trivial matching caused by overlapping windows. This definition can be used to characterize self-connections within the same asset and is also suitable for cross-connections between multiple assets.

Let the total number of candidate pairs be N_pair, the distance between a single pair be m, and the GPU parallelism be approximately denoted as C. Then the query time can be expressed as an approximate model.

$$T_{gpu}\approx \max\left(\frac{B_{mem}}{\beta},\frac{N_{pair}\cdot m}{C}\right) \quad (5)$$

Equation (5) indicates that the GPU execution time is the larger of the memory access and computation parts. The total number of bytes accessed is represented by B_mem, and the effective bandwidth is represented by β. Therefore, it can be seen that the key to optimization is not only to speed up floating-point operations, but also to reduce global memory write-back, improve access continuity, and compress the candidate set.

4. Problem Solving and Optimization Strategies

To test the basis of financial time series join optimization, this paper constructs a real series using Apple's historical daily closing price data, selecting 701 trading days from January 2006 to October 2008 as the experimental prototype. This sample includes various scenarios where upward trends and significant retracements could lead to crisis impacts, appropriately representing the statistical complexity of searching and retrieving financial series. Table 1 presents the basic statistical characteristics of the sample.

Table 1 shows a sample period of 701 trading days, with a maximum drawdown of 55.59%, and 20-day volatility significantly increased during the crisis. This indicates that join queries on financial time series are not stationary signal processing methods; they seek locally similar windows in non-stationary, heteroscedastic environments. Therefore, window standardization and adaptive threshold settings are essential preprocessing steps before GPU acceleration; otherwise, numerous high-volatility periods will amplify noise during candidate generation.

Table 1 Statistical characteristics of the financial sample used in the experiment

Metric	Value
Trading days	701
Start date	2006-01-03
End date	2008-10-14
Mean return	0.000528
Std return	0.027631
Max drawdown	-0.5559
Mean volume	34753169

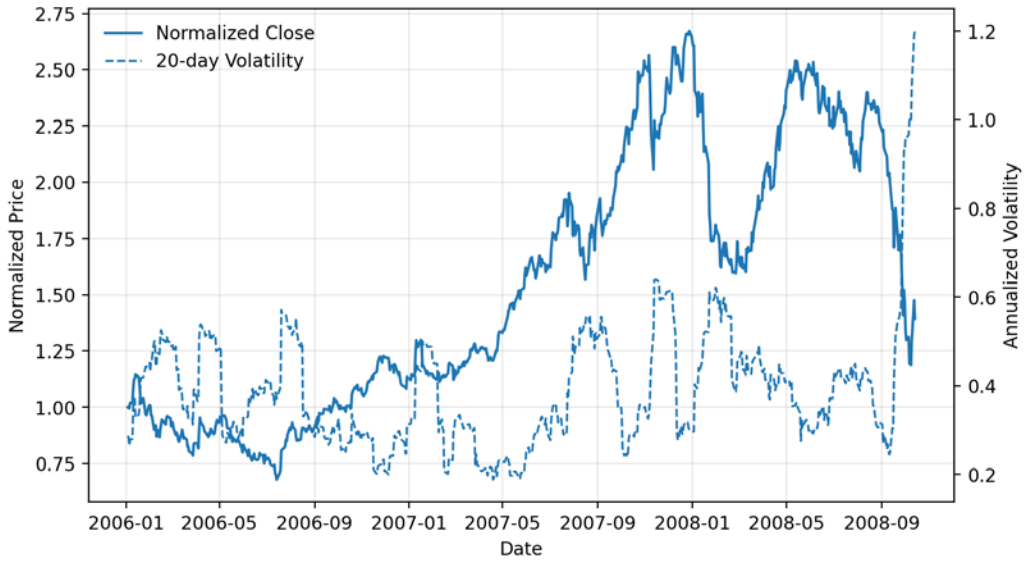

Figure 1. Normalized Prices and 20-Day Annualized Volatility of Financial Samples

Figure 1 shows a significant rise in normalized closing prices during the first quarter of 2007-2008, followed by a rapid decline at the end of 2008 due to market shocks, while the 20-day annualized volatility also increased sharply during this period. This theorem indicates that financial time series clearly exhibit state-switching phenomena, rather than simply looking at price levels. For GPUs, the differences between windows in the batch queries they handle are uneven; without bucketing and batch scheduling, severe load imbalances occur between thread bundles.

Based on this, this paper proposes a GPU parallel optimization strategy consisting of four layers. The first layer is the data preprocessing layer. Prefix sums and prefix square sums are used to calculate the mean and variance of all windows at once, preventing redundant standardization. The second layer is the candidate filtering layer. Window summary vectors, low-dimensional approximations, and threshold lower bounds are used to filter out most impossible-to-match window pairs, leaving the laborious precise distance testing to a small number of candidates. The third layer is the distance calculation layer. Windows are batched into thread blocks, and the query windows are cached in shared memory on the thread blocks, reducing the number of repeated global memory reads. The fourth layer is the result output layer. Intra-block counting and compressed writing are used to write only matching pairs that meet the threshold to global memory, reducing materialization.

The core idea is to transform query joins into offline batch distance calculation tasks. Specifically, this can be done by binning based on window length, asset code, fluctuation range, or predicted distance range, ensuring that tasks within the same batch have similar computational costs, thereby improving the consistency of thread execution. For batch queries, a shared kernel function can be defined to perform multiple queries, and shared memory can be used to reuse target sequence slices to improve locality of access on the device.

In addition, the GPU implementation needs to handle the thread hierarchy in detail. Using threads to process a single point and using thread blocks to process a single pair of windows is more suitable for bandwidth-constrained scenarios. For short-window financial segments, the utilization rate of SM can be increased to improve it; while for long-window segments, the number of candidates in the block should be reduced to prevent excessive register pressure. This method is consistent with the results of Wu et al. on workload-aware GPU connection optimization [1].

Table 2. Connection candidate statistics under different window lengths

Window	Windows	Pairs	MeanDist	P01
16	243	29403	5.612	3.674
32	235	27495	7.971	6.165
48	227	25651	9.777	8.018
64	219	23871	11.296	9.563
96	203	20503	13.843	12.142
128	187	17391	15.989	14.289

Table 2 shows the connection statistics for different window lengths. When the window length increases from 16 to 128, the total number of windows and candidate pairs decreases, but the low quantile of the distance distribution increases. This indicates that longer windows have stronger pattern discrimination capabilities, but the cumulative distance length per pass also increases. Therefore, GPU implementations cannot simply determine batch size based on the number of candidates; the computational density changes caused by window length must also be considered. Short windows are suitable for sending large numbers of packets, while long windows are suitable for sending packets with strong pre-filtering and small block granularity.

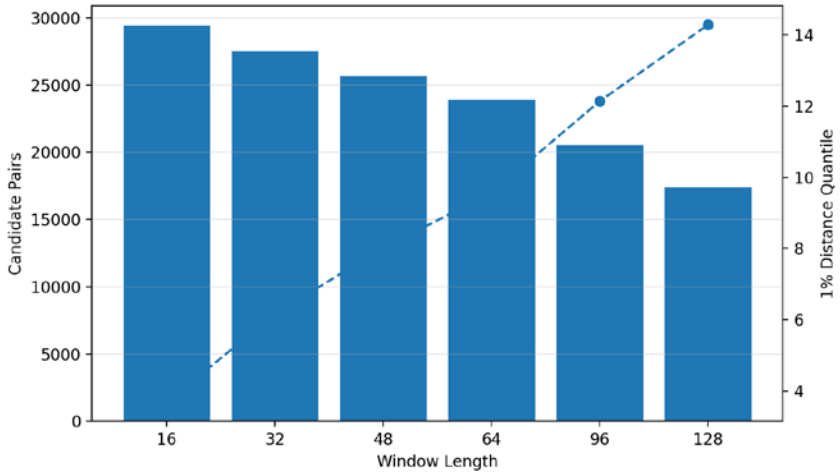

Figure 2. Variation of candidate size and distance quantile under different window lengths.

Figure 2 shows that increasing the window length reduces the number of candidates but increases the 1% distance quantile. This means that longer windows have fewer similar windows, resulting in higher benefits from threshold filtering. For GPU engines, this statistical result can be used to analyze the query optimizer. When the window is short and the candidates are concentrated, memory bandwidth and result compression should be optimized first; while when the window is long and the candidates are sparse, more emphasis should be placed on pre-filtering and kernel function fusion to avoid wasting computing power due to a large number of invalid verifications.

Based on the aforementioned statistical patterns, this paper offers suggestions for implementing financial time series joins. First, given sufficient device memory, use a structured array layout to arrange the query window and target window, allowing adjacent threads to access contiguous addresses. Second, before performing precise distance calculations in shared memory, use low-dimensional summaries to quickly eliminate obviously inconsistent cases. Third, process results that reach a threshold using prefix counting and compressed write-back to prevent each thread from writing atomically and individually. Fourth, when performing joint queries on multiple assets, first batch them according to fluctuation ranges to reduce branch divergence within thread bundles.

Financial time series join queries should not only focus on maximum throughput but also consider the stability of the results. If the threshold is set too low, although the GPU can quickly generate many matching pairs, these pairs have little practical analytical value; if the threshold is too high, some similar segments with interpretable transaction significance will be ignored. Therefore, the system design should combine device-side execution and host-side parameter tuning, using offline statistical estimation to estimate the distance distribution under different window lengths, and then having the query optimizer adaptively generate the threshold and batch size.

The strategy proposed in this paper is essentially a way to drive hardware mapping using statistical cognition. This involves first understanding the candidate distribution and similarity sparsity of financial sequences, and then determining the GPU's execution strategy. This avoids forcibly migrating general GPU connection operators to financial tasks and lays the foundation for using the same system framework when extending to DTW, correlation coefficient distance, or multi-factor distance in the future.

5. Conclusion

This paper systematically reviews the optimization problem of financial time series similarity join queries based on GPU parallel computing. Building upon research findings from the past three years on GPU join processing, time series distance calculation, financial series similarity modeling, and retrieval enhancement, an analytical framework from an engineering implementation perspective is constructed. Research reveals that the main problem in financial time series join queries is not the complexity of a single distance function, but rather a bottleneck caused by the combined effects of window standardization, candidate enumeration, distance verification, and result materialization.

Starting with real financial samples, we can observe that financial sequences are non-stationary, exhibit significant fluctuation clusters, and state transitions. This dictates that GPU optimization must adhere to the principle of "statistical distribution first, then mapping and execution." Based on this understanding, this paper proposes a window normalization method based on prefix statistics, a candidate filtering method based on summary lower bounds, a batch distance calculation method based on shared memory, and a result output method based on compression and write-back. The query optimizer needs to adaptively schedule based on window length, fluctuation level, and candidate

sparsity.

There are three areas that can be further explored in the future: first, using Euclidean distance to extend to DTW, MPdist, etc.; second, using multi-GPU and streaming incremental computing to support larger real-time market data processing; and third, embedding the results of similarity connections into event-driven trading, anomaly propagation detection, and risk linkage identification systems to form a complete quantitative financial application technology chain.

References

[1] *Liu, D., Shen, Q., & Liu, J. (2026). The Health-Wealth Gradient in Labor Markets: Integrating Health, Insurance, and Social Metrics to Predict Employment Density.*

[2] *Lu, Z. (2025). Design and Practice of AI Intelligent Mentor System for DevOps Education. European Journal of Education Science, 1(3), 25-31.*

[3] *Hui, X. (2026). Research on the Design and Optimization of Automated Data Collection and Visual Dashboard in the Medical Industry. Journal of Computer, Signal, and System Research, 3(1), 27-34.*

[4] *Wang, Y. (2026). Research on Optimization of Neuromuscular Rehabilitation Program Based on Physiological Assessment. European Journal of AI, Computing & Informatics, 2(1), 21-30.*

[5] *Ding, J. (2026). Optimization Strategies for Supply Chain Management and Quality Control in the Automotive Manufacturing Industry. Strategic Management Insights, 3(1), 17-23.*

[6] *Zhang, Q. (2026). How to Improve Marketing Efficiency and Precision through AI-Driven Innovative Products. Strategic Management Insights, 3(1), 1-8.*

[7] *Liu, Y. (2026). The Promoting Role of Fintech and Product Innovation in the Context of the Digital Economy. Strategic Management Insights, 3(1), 9-16.*

[8] *Lu, C. (2026). Research on 3D Reconstruction Methods of Remote Sensing Images Combined with Deep Learning and GIS. International Journal of Engineering Advances, 3(1), 15-22.*

[9] *Cai, Y. (2026). Design and Implementation of System Extensibility under High Concurrency Environment. International Journal of Engineering Advances, 3(1), 31-37.*

[10] *Liu, Y. (2026). The Application of Data-Driven Financial Risk Management in Multinational Enterprises. Economics and Management Innovation, 3(1), 20-26.*

[11] *Huang, J. (2026). Practice of Public Space Optimization and Functional Enhancement in Cultural Architecture. European Journal of Engineering and Technologies, 2(1), 9-21.*

[12] *Xu, D. (2026). AI-Driven Video Content Optimization Strategies for Immersive Media. European Journal of Engineering and Technologies, 2(1), 1-8.*

[13] *Qi, Y. (2026). AI Driven Payment System Security Improvement and User Privacy Protection Mechanism. Journal of Computer, Signal, and System Research, 3(1), 35-41.*

[14] *Qi, Y. (2026). High Reliability Architecture and Compliance Design of Enterprise Level Financial Infrastructure. International Journal of Engineering Advances, 3(1), 8-14.*

[15] *Lu, Z. (2025). Design and Practice of AI Intelligent Mentor System for DevOps Education. European Journal of Education Science, 1(3), 25-31.*

[16] *Lu, Z. (2025). AI-Driven Cross-Cloud Operations Language Standardisation and Knowledge Sharing System. European Journal of AI, Computing & Informatics, 1(4), 43-50.*

[17] *Wang, C. (2025). Research on Market Evaluation Strategies for Financial Institutions Based on Big Data Analysis.*

[18] Hou, Y. (2026). Research on BIOS and BMC Compatibility Optimization Methods for Cross-Generation Servers in Production Environments.

[19] Zhang, X. (2025). Optimization and Implementation of Time Series Dimensionality Reduction Anti-fraud Model Integrating PCA and LSTM under the Federated Learning Framework. Procedia Computer Science, 262, 992-1001.

[20] Hou, Y. (2026). Research on Server Performance Stability Assurance Mechanisms during Cross-Generation Computing Platform Upgrades.

[21] Zhang, X. (2025, May). Automobile Finance Credit Fraud Risk Early Warning System based on Louvain Algorithm and XGBoost Model. In 2025 3rd International Conference on Data Science and Information System (ICDSIS) (pp. 1-7). IEEE.

[22] Hong, Y. (2025). Architecture Design and Performance Optimization of a Large-scale Online Simulation Platform for Business Decision-making. Advances in Computer and Communication, 6(4).

[23] Truong, T. H. (2025). Research on the Application of Digital Healthcare Platforms in Chronic Disease Management. Advances in Computer and Communication, 6(5).

[24] Huijie Pan. Discussion on Low-Latency Computing Strategies in Real-Time Hardware Generation. International Journal of Neural Network (2025), Vol. 4, Issue 1: 57-64.

[25] Yixian Jiang. Performance Optimization and Improvement of Advertising Machine Learning Platform Based on Distributed Systems. International Journal of Big Data Intelligent Technology (2026), Vol. 7, Issue 1: 9-17

[26] Zhengle Wei. Research on Innovative Design of Financial Derivatives and Market Risk Management Strategies. International Journal of Social Sciences and Economic Management (2026), Vol. 7, Issue 1: 19-27

[27] Yilin Fu. Research on the Application of Innovative Financial Technologies in Capital Market Risk Management. Socio-Economic Statistics Research (2026), Vol. 7, Issue 1: 1-9

[28] Linwei Wu. Data Visualization and Decision Support Analysis Based on Tableau. Socio-Economic Statistics Research (2026), Vol. 7, Issue 1: 10-18

[29] Han, X. (2026). Research on Process Decision-Making Behavior under Incomplete Information Conditions in Automobile Manufacturing Systems.

[30] Yin, J. (2026). Research on Financial Time Series Prediction and Multiscale Correlation Based on the Fusion of Network Big Data and Deep Learning.

[31] Zhou, Y. (2024, November). Construction of a Multi-factor Quantitative Stock Selection System for the New Energy Industry Based on Microservices Architecture and Machine Learning Components. In International Conference on Cognitive based Information Processing and Applications (pp. 163-174). Singapore: Springer Nature Singapore.

[32] Huang, J. (2025, August). Research on Multi-Model Fusion Machine Learning Demand Intelligent Forecasting System in Cloud Computing Environment. In 2025 2nd International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS) (pp. 1-7). IEEE.

[33] Hua, X. (2024, November). Design and Implementation of a Game QoE Monitoring and Evaluation System Driven by Network Traffic Analysis. In International Conference on Cognitive based Information Processing and Applications (pp. 149-161). Singapore: Springer Nature Singapore.

[34] Qi, Y. (2025, October). Research on Privacy Protection of AI Models in Big Data Using Differential Privacy Technology. In 2025 2nd International Conference on Software, Systems and Information Technology (SSITCON) (pp. 1-5). IEEE.

[35] Truong, T. H. (2026). Research on Risks and Countermeasures of Enterprise E-Commerce Transformation in the Cross-Border E-Commerce Environment.

[36] Wu, L. (2025, December). Design and Application of Automatic Data Set Generation Tool Based on KLEE in Embedded Memory Management Performance Test Framework. In 2025 IEEE 17th International Conference on Computational Intelligence and Communication Networks (CICN) (pp. 1111-1117). IEEE.

[37] Wu, L. (2025, December). Design and Application of Automatic Data Set Generation Tool Based on KLEE in Embedded Memory Management Performance Test Framework. In 2025 IEEE 17th International Conference on Computational Intelligence and Communication Networks (CICN) (pp. 1111-1117). IEEE.

[38] Dingyuan Liu. Measuring the Sensitivity of Local Skill Structures to AI Substitution Risks Based on Occupational Task Decomposition. Socio-Economic Statistics Research (2025), Vol. 6, Issue 2: 177-184

[39] Hong, Y. (2026). Research on Warehouse Capacity Optimization Methods Based on Predictive Modeling. Engineering Advances, 6(1).

[40] Yu, X. (2026). Strategy Models and Practical Research of Growth Marketing under the Background of Digital Transformation.