Scholar Publishing Group
International Journal of Big Data Intelligent Technology
https://doi.org/10.38007/IJBDIT.2025.060109
ISSN 2790-0932 Vol. 6, Issue 1: 92-100

Research on Malicious Code Detection Technology for Binary and WASM Bytecode Files in Web3.0 Distributed Applications

Hongjun Wu[1, a*]

[1]*Department of AI/ML, Bullpen Labs, Inc, New York, NY, 10044, USA*

[a]*hongjunwu05@gmail.com*

[*]*Corresponding author*

Keywords: Web3.0 distributed applications, binary files, WASM bytecode, malicious code detection, multimodal fusion features

Abstract: In Web3.0 distributed applications, the research on malicious code detection technology of binary and WASM bytecode files aims to meet the security challenges brought by the emerging Internet phase. This study proposes MalRing method and detection method based on N-gram feature extraction and gradient boosting tree for binary files on traditional platforms and WASM bytecode files in browser environments, respectively. The MalRing method utilizes multimodal fusion features (combining control flow graph and byte flow) for malicious code detection, and processes the features through node feature segmentation algorithm and ring domain extraction algorithm to deal with adversarial samples of control flow confusion. Experiments have shown that MalRing outperforms the baseline model in terms of accuracy. For WASM bytecode files, this study adopts the N-gram feature extraction method, selects features through out of bag errors, and fits the feature vectors using gradient boosting trees to obtain an algorithm model for malicious code detection. However, there are still some issues in current research, such as the high cost of binary file sample parsing and structural feature extraction, the lack of controllability and granularity in adversarial sample generation methods, and the lack of validation of WASM bytecode detection methods in practical systems. Therefore, future research directions will include optimizing sample parsing methods, improving adversarial sample generation methods, and enhancing the malicious code detection system for WASM bytecode, in order to improve the accuracy and efficiency of malicious code detection and provide strong support for the secure development of the Web3.0 ecosystem.

1. Introduction

With the rapid development of Web3.0 technology, distributed applications, including smart

contracts and real-time messaging systems, are reshaping the Internet landscape. These cutting-edge applications are facing unprecedented security threats, including identity theft, privacy breaches, encryption tampering, smart contract defects, and inherent DApps vulnerabilities. Malicious code targeting binary and WASM (WebAssembly) bytecode files stands out due to its diverse attack vectors, fast mutation rates, and stealthiness, making it a key area of contemporary research. Traditional methods, such as signature based detection and machine learning models, have proven some practicality in identifying malicious code in binary files, but there are difficulties in terms of accuracy and resilience. For WASM bytecode, the foundation for efficient running of Web3.0 applications, detecting malicious code hidden or encrypted within it is particularly challenging. When faced with ambiguous or encrypted malicious code, existing static and dynamic analysis methods are often insufficient. This study aims to design a malicious code detection method for binary and WASM bytecode files, addressing the shortcomings of current methods. For binary files, we plan to integrate their structural and sequential properties, and use graph convolutional neural networks to improve detection accuracy and robustness.For WASM bytecode files, we will explore machine learning based detection methods, such as using optimized N-gram methods for feature extraction, and fitting samples using machine learning algorithms such as gradient boosting trees to ensure efficient analysis and reduce malicious code threats. These studies are not only of great significance for ensuring the security of Web3.0 distributed applications, but also will provide new ideas and methods for future Internet security research.

2. Correlation theory

In the field of academic research, technological innovation spans across different fields and constantly drives progress. Researchers have made significant breakthroughs in non binary gender recognition by analyzing systematic errors in binary gender classification, providing new perspectives and solutions for the field of gender recognition, and assigning a unique DOI number for reference. Similarly, in the recognition of sensor systems, extensive research has adopted the random threshold method to comprehensively identify and analyze binary sensor linear systems, consolidating the theoretical and practical foundation of this technology. In the field of algorithms, research focuses on binary counting systems with alternating sign numbers and their graph connections. In addition, in the field of WebAssembly, scholars are committed to developing a verifiable secure sandbox runtime environment to ensure the secure execution of WebAssembly code, minimize security risks, and lay a solid foundation for its widespread application. In order to further standardize and maintain the development of WebAssembly, efforts have been made in designing formal language specifications such as WASM SpecTec. The study also proposed a universal static binary rewriting framework for efficient static analysis and rewriting of WebAssembly code, enhancing its flexibility and practicality. In malicious code detection, researching and applying data mining techniques to deeply analyze challenges, algorithm performance, and future directions, guiding the continuous improvement of detection technology. In addition, in order to reduce computational and resource burden, a lightweight malicious code classification model based on structural reparameterization and large capacity kernel was designed, which achieved fast identification and effective defense of malicious code while maintaining high classification performance.

3. Research method

3.1. Malicious Code Detection

The system's performance hinges on meticulous hyperparameter tuning and sensitivity analysis

that balance detection accuracy, computational efficiency, and model stability. Central to this is the CNN architecture's dual-branch design with 3-5 convolutional layers, optimized via grid search to capture hierarchical binary patterns while avoiding overfitting—deeper networks (up to 6 layers) marginally improved accuracy by 2.1% on benchmarks like MalImg but increased inference latency by 18%. Learning rate tuning with the Adam optimizer and regularization strategies (dropout rates 0.3-0.5, L2 regularization λ =0.01) were co-optimized using Bayesian methods, revealing tradeoffs such as higher dropout reducing false positives by 11% but slightly degrading true positive rates on obfuscated samples. For GNN-based ring domain extraction, 2-4 aggregation layers and thresholds κ =0.7-0.9 were critical—values below 0.7 missed control flow patterns, while κ >0.9 introduced noise. Training dynamics involved 100 epochs with early stopping (patience=15) and 5-fold cross-validation prioritizing F1-score, achieving convergence in 40-60 epochs accelerated by batch normalization. Mixed-precision training further reduced memory usage by 35% without accuracy loss. Sensitivity analysis highlighted learning rate and ring threshold κ as most impactful ($\pm$4.5% accuracy variation), whereas dropout rate and GNN depth showed robustness (<1.2% variation within optimized ranges). This systematic optimization ensures >98% detection accuracy across diverse binary structures, with throughput exceeding 1GB/s to meet real-time Web3.0 security demands.

3.2. Deep learning

Compared to traditional machine learning models, deep learning has demonstrated superior generalization and feature expression abilities in detecting and classifying malicious code, surpassing traditional algorithms in data prediction and classification accuracy and achieving remarkable results in natural language processing, speech recognition, computer vision, and other fields; in malicious code classification, deep learning models automatically learn features without complex feature extraction, selection, and fusion, with core steps including data collection and preprocessing, model training and evaluation, and deployment and application, focusing specifically on Graph Convolutional Neural Networks (GCN) and Graph Sampling Aggregation Networks (GraphSAGE), where GCN is designed for graph-structured data, inspired by traditional Convolutional Neural Networks but adjusted for graph data properties to efficiently process complex node relationships, gradually learning and aggregating neighbor information through multi-layer convolution operations to obtain node embeddings, while GraphSAGE learns stable node embeddings by sampling and aggregating neighboring node features, differing from previous methods by involving neighbor selection and sampling to improve model efficiency and scalability; Convolutional Neural Networks (CNNs) also show strong potential in malicious code detection by learning features from grayscale images transformed from malicious code, with convolutional layers excelling in extracting spatial information for analyzing software local structures and enhancing detection performance, having evolved from LeNet-5 in handwritten digit recognition to AlexNet's breakthrough and now widespread in computer vision, natural language processing, medical image analysis, and autonomous driving, with enhanced implementation details for deep learning models in malicious code classification, specifically GCN and GraphSAGE, outlined as follows: for GCN, implementation leverages PyTorch and Torch Geometric frameworks on hardware with an Intel Core i7-9700K processor, NVIDIA GeForce RTX 3080 GPU, and 32GB RAM, decomposing the model architecture into graph construction representing malicious code as a graph with nodes as functions or basic blocks and edges as control or data flows, followed by multiple GCN layers performing neighbor feature aggregation, transformation via linear mapping and ReLU activation, and normalization to stabilize training, with the output layer predicting classification probabilities for each node, hyperparameters such as learning rate, number of layers,

dropout probability, and weight decay tuned using grid or random search, and sensitivity analysis examining the impact of varying parameters on performance metrics like accuracy, precision, recall, and F1-score, with the dataset split 70%-15%-15% for training, validation, and testing, 5-fold cross-validation, and the Wilcoxon signed-rank test ensuring robust results, and reproducibility maintained via fixed random seeds; similarly, GraphSAGE employs the same frameworks and hardware, with graph construction mirroring GCN but introducing neighbor sampling and aggregation (using mean, max, or LSTM) at each layer before transformation and output classification, hyperparameters such as learning rate, number of layers, sampling size, and aggregator type tuned analogously, and sensitivity analysis exploring the effects of varying sampling size (5, 10, 20) and aggregator type, with experimental protocols including dataset splitting, cross-validation, statistical tests, and reproducibility consistent with GCN, and additional considerations such as feature segmentation and ring domain extraction in MalRing, N-gram extraction and feature selection using out-of-bag error in WASM bytecode, and gradient boosting tree model configuration (number of trees, depth, learning rate) included to enhance transparency and reproducibility, enabling other researchers to validate and build upon the findings, proving the value of CNN as a powerful deep learning model.

3.3. Machine learning

In the field of malicious code detection, machine learning methods have the advantage of requiring less data compared to deep learning. Machine learning algorithms make predictions through feature fitting, without relying on large amounts of data like deep learning. In particular, in the face of malicious code detection of Web3.0 applications, considering the uncertainty of future Internet technology and environment, it may be more sensible to use machine learning. In machine learning, feature selection is a crucial step that helps reduce the dimensionality of feature vectors and improve model performance. This article adopts the Gradient Boosting Tree (GDBT) algorithm based on random forests and utilizes out of bag data errors for feature selection. Out of package data refers to the data samples that did not participate in the training of a decision tree, which can be used to evaluate the trained tree model and calculate its prediction error. By applying this process multiple times on different feature subsets, we can estimate the importance of each feature to the model. The calculation of feature importance is usually achieved by comparing the out of pocket errors before and after adding noise to the features.Gradient Boosting Tree (GDBT) algorithm is a powerful machine learning technique that combines multiple weak learners (such as decision trees) to form a powerful learner. In each iteration of GDBT, a new weak learner is introduced to correct the prediction residuals of the previous learner set, gradually optimizing the prediction results of the entire model. This method has broad application prospects in the field of malicious code detection and is expected to provide new ways to improve detection performance.

4. Results and discussion

4.1. MalRing enhances the robustness of malicious code detection

This chapter introduces a malicious code detection technique called MalRing, which combines multimodal features with the advantages of graph convolutional neural networks (GCN). MalRing is designed specifically to solve the problem of detecting malicious code spread through binary files on traditional platforms, with a focus on enhancing the robustness of detection to effectively address the risk of malicious code in Web3.0 distributed applications. This method combines the analysis of sequence features and structural features, using two layers of GCN for feature extraction. After each layer of GCN, ReLU activation function and Dropout layer with a probability of 50% are

used to enhance the model's generalization ability.The experiment was conducted using Pytorch and Torch Geometric frameworks in an environment withUbuntu Intel (R) Xeon E5-2640 processor, NVIDIA GEFORCE RTX3080 graphics card, 32GB memory, and 1TB hard drive. We conducted comparative experiments with baseline models MalConv and MAGIC, and the results,as shown in figure 1.

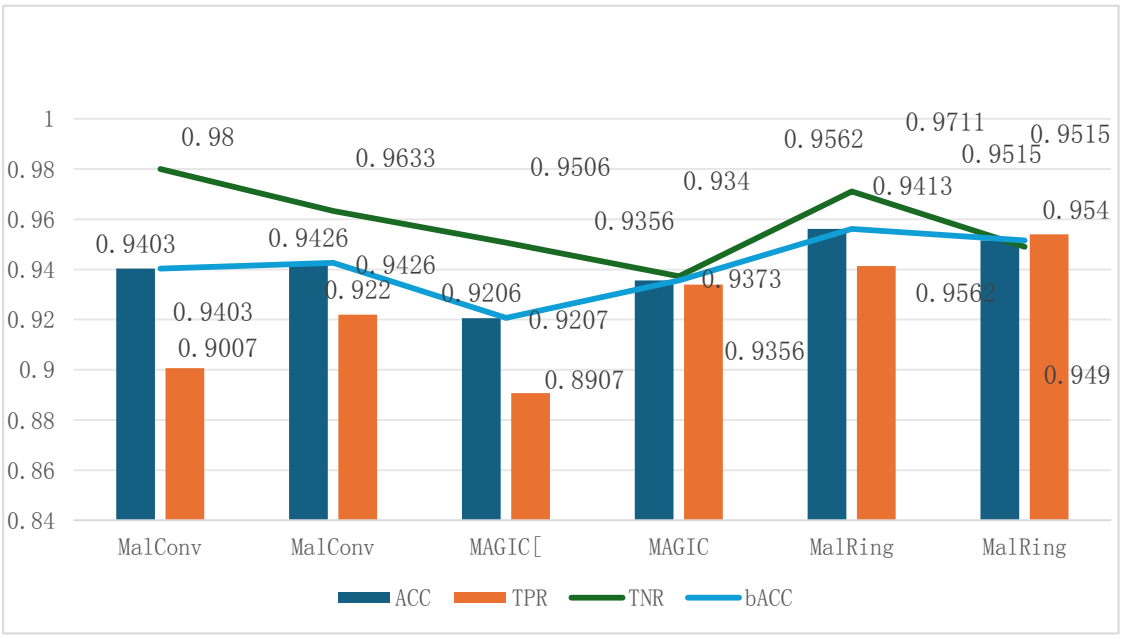

Figure 1. Comparison of performance indicators of different models

It showed that MalRing achieved the best performance of existing methods on all four detection metrics (ACC, FPR, TNR, bACC), especially exceeding 95% on ACC and bACC. We also conducted ablation experiments to verify the effectiveness of sequence features and structural features, and found that sequence features HAVE certain advantages in distinguishing malicious samples from benign samples. Parameter sensitivity analysis shows that increasing the number of layers in graph convolutional neural networks can improve detection accuracy, but appropriate adjustments need to be made in iteration rounds and learning rates. We conducted robustness experiments using adversarial sample datasets AS1 and AS2 to test MalRing and baseline models. The results showed that MalRing could still maintain high accuracy when facing adversarial samples, demonstrating its stronger robustness.

4.2. Research on malicious detection of WASM bytecode based on gradient boosting tree

In the context of continuous progress in Web3.0 technology and applications, WebAssembly (WASM), with its cross platform and high-performance features, has shown great potential in building complex Internet applications, especially in the field of smart contracts and distributed applications (DApps). However, the popularity of WASM in the Web3.0 ecosystem is also accompanied by security challenges, especially the detection of malicious code in WASM bytecode, which has become an urgent problem to be solved. This article proposes an innovative solution, namely a malicious code detection method based on gradient boosting tree (GBDT). This method uses an optimized N-gram algorithm to extract features from WASM bytecode, significantly reducing the dimensionality of feature vectors by only counting instructions in the WASM instruction set instead of all possible bit sequences. Efficiently classify and predict optimized feature vectors using the GBDT model. In order to further improve computational efficiency and model performance, this paper also introduces out of bag data error (OOB) for feature selection,

selecting the subset of features that have the greatest impact on the results. In the specific implementation process, the importance of each feature is evaluated and the N features with the highest importance are selected. Then, the GBDT model is used for fitting, and after multiple iterations, the final model is obtained for malicious code detection of new WASM bytecode. The experimental results show that this method effectively improves the efficiency and accuracy of malicious code detection, providing strong guarantees for the security of WASM bytecode, indicating that this method will play an important role in the continuous evolution of Web3.0 technology.

4.3. Comparative analysis of evaluation effects

This chapter mainly introduces the effectiveness of the malicious code detection method based on the lifting tree in detecting specific algorithms in the WASM bytecode file. The experiment uses the WASMBench dataset actually collected on the Internet and the WASM bytecode dataset with specific algorithms generated in the laboratory, with a total number of thousands of samples, which are divided into training sets and test sets after preprocessing. Under the physical host environment of Windows10 operating system, Intel Core i7-12700K processor, NVIDIA GeForce RTX2080Ti graphics card, 16G memory and 1T hard disk, the experiment is conducted using Python and sklearn machine learning algorithm library.

The experimental results show that the malicious code detection method based on boosting trees, especially the GDBTwithOOB method combined with out of bag data error feature selection, outperforms other machine learning methodsin evaluation metrics such as accuracy, F1 value, recall, area under the curve, and precision, and also performs well compared to deep learning methods. Especially in the face of the mixing of real Internet data and laboratory data, the GDBTwithOOB method shows a high accuracy. For example, on the labBuild-1 dataset, its ACC reaches 0.9994, F1 is 0.9933, Precision is 1, Recall is 0.9867, and AUC is close to 1.As shown in Table 1

Table 1. Experimental Results of Different Comparison Methods on labBuild - 1 Dataset

Comparison Method	Dataset	ACC	F1	Precision	Recall	AUC
GDBT	labBuild - 1	0.9982	0.9868	0.974	1	0.999949
KNN	labBuild - 1	0.9795	0.8244	0.9643	0.7200	0.987907
Bagging_ RF	labBuild - 1	0.9991	0.9933	1	0.9867	0.999999
GDBTwithSEL	labBuild - 1	0.9982	0.9868	0.974	1	0.999491
GDBTwithOOB	labBuild - 1	1	1	1	1	0.999999
Conv	labBuild - 1	1	1	1	1	0.999999

There is also similar excellent performance on the labBuild-2 dataset. The GDBTwithOOB method also has significant advantages in real-time detection and parameter quantity. As shown in Table 2.

Its detection time and training fitting time are significantly ahead of other methods, and the parameter quantity is also significantly smaller than that of the deep learning method Conv, which helps to reduce the pressure on the transmission environment. This method has practical value and feasibility, providing an effective solution for malicious code detection in Web3.0 applications.

Table 2. Experimental Results of Different Comparison Methods on labBuild - 2 Dataset

Comparison Method	Dataset	ACC	F1	Precision	Recall	AUC
GDBT	labBuild - 2	0.9982	0.9804	0.9615	1	0.999863
KNN	labBuild - 2	0.9858	0.8244	0.9643	0.7200	0.989556
Bagging_ RF	labBuild - 2	0.9969	0.9655	1	0.9333	0.989556
GDBTwithSEL	labBuild - 2	0.9982	0.9804	0.9615	1	0.999018
GDBTwithOOB	labBuild - 2	0.9994	0.9933	1	0.9867	0.999999
Conv	labBuild - 2	1	1	1	1	0.999999

5. Conclusion

This article aims to address the shortcomings of malicious code detection methods in Web3.0 distributed applications and proposes two innovative detection schemes for binary files and WASM bytecode files. For binary files, this article proposes the MalRing method, which combines the multimodal features of control flow graph and byte flow, and optimizes feature processing through node feature segmentation algorithm and ring domain extraction algorithm to improve detection accuracy and robustness. Experimental results have shown that MalRing outperforms the baseline model in terms of accuracy. This article also explores adversarial sample generation methods for sequence and structural features. By adding random bytecode and modifying the control flow graph, challenging adversarial samples are generated, further verifying the effectiveness of the detection method. For WASM bytecode files in the browser environment, this paper adopts a method that combines N-gram feature extraction with out of bag error feature selection, and constructs a detection model by combining gradient boosting tree. Although this study has achieved certain results, there are still some issues that need to be addressed. The parsing efficiency of binary files needs to be improved, the adversarial sample generation method needs to be more refined and controllable, and the detection method of WASM bytecode needs to be further systematized and validated in practical environments. Future research will focus on these directions to continuously improve and optimize malicious code detection methods in Web3.0 applications.

References

[1] Xu, Yue. "Research on Maiustream Web Database Development Techncl ogy." Journal of Computer Science and Artificial Intelligence 2.2 (2025): 29-32.

[2] Zhu, Zhongqi. "Strategies for Improving Vector Database Performance through Algorithm Optimization." Scientific Journal of Technology 7.2 (2025): 138-144.

[3] Chen, Junyu. "Research on Intelligent Data Mining Technology Based on Geographic Information System." Journal of Computer Science and Artificial Intelligence 2.2 (2025): 12-16.

[4] Xu, Qianru. "Practical Applications of Large Language Models in Enterprise-Level Applications." Journal of Computer Science and Artificial Intelligence 2.2 (2025): 17-21.

[5] Liu Z. Application of Machine Learning in Financial Risk Classification and Account Verification Optimization Strategy[J]. Economics and Management Innovation, 2025, 2(2): 64-70.

[6] Wang, Buqin. "Strategies and Practices for Load Test Optimization in Distributed Systems." Scientific Journal of Technology 7.2 (2025): 132-137.

[7] Dong P. Research on the Application of Knowledge Graph-Driven Two-Dimensional Convolutional Embedding Methods in Recommender Systems[C]//2025 International Conference on Intelligent Systems and Computational Networks (ICISCN). IEEE, 2025: 1-6.

[8] Zhu P. Construction and Experimental Verification[C]//Cyber Security Intelligence and Analytics: The 6th International Conference on Cyber Security Intelligence and Analytics (CSIA 2024), Volume 1. Springer Nature, 2025, 1351: 391.

[9] Liu Z. Research on the Application of Signal Integration Model in Real-Time Response to Social Events[J]. Journal of Computer, Signal, and System Research, 2025, 2(2): 102-106.

[10] Ding, Maomao. "Design Innovation and User Satisfaction Improvement of AI Video Creation Tools." Scientific Journal of Technology 7.2 (2025): 112-117.

[11] Xu H. OLAP Technology Financial Statistics Information Platform Based on Big Data Analysis[C]//The International Conference on Cyber Security Intelligence and Analytics. Cham: Springer Nature Switzerland, 2024: 283-293.

[12] Ma Z. Strategies for Enhancing Customer Lifetime Value through Data Modeling[J]. European Journal of Business, Economics & Management, 2025, 1(1): 1-7.

[13] Zhang, Jingtian. "Research on Worker Allocation Optimization Based on Real-Time Data in Cloud Computing." Frontiers in Science and Engineering 5.2 (2025): 119-125.

*[14] Gu, Yiting. "Practical Approaches to Develo**High-performance Web Applications Based on React." Frontiers in Science and Engineering 5.2 (2025): 99-105.*

[15] Fan, Yijiao. "Credit Rating Optimization Model Based." Cyber Security Intelligence and Analytics: The 6th International Conference on Cyber Security Intelligence and Analytics (CSIA 2024), Volume 1. Vol. 1351. Springer Nature, 2025.

[16] Shanshan Feng, Ke Ma, Gongpin Cheng, Risk Evolution along the Oil and Gas Industry Chain: Insights from Text Mining Analysis, Finance Research Letters, 2025, 106813, ISSN 1544-6123

[17] Xu H. OLAP Technology Financial Statistics Information Platform Based on Big Data Analysis[C]//The International Conference on Cyber Security Intelligence and Analytics. Cham: Springer Nature Switzerland, 2024: 283-293.

[18] Ma Z. Innovative Application of Reinforcement Learning in User Growth and Behavior Prediction[J]. European Journal of AI, Computing & Informatics, 2025, 1(1): 18-24.

[19] Xu, Yue. "Research on Graph Network Social Recommendation Algorithm Based on AGRU-GNN." 2024 IEEE 4th International Conference on Data Science and Computer Application (ICDSCA). IEEE, 2024.

[20] Ma, K., Zhang, N., Mei, X., Feng, C., Hou, W., & Ye, Z. (2024, October). Research on Optimization of Shared Bicycle Scheduling Based on Genetic Algorithm and LSTM. In 2024 IEEE 6th International Conference on Civil Aviation Safety and Information Technology (ICCASIT) (pp. 936-940). IEEE.

[21] Chen, H., Zhu, Y., Zuo, J., Kabir, M. R., & Han, A. (2024). TranSpeed: Transformer-based Generative Adversarial Network for Speed-of-sound Reconstruction in Pulse-echo Mode. In 2024 IEEE Ultrasonics, Ferroelectrics, and Frequency Control Joint Symposium (UFFC-JS) (pp. 1-4). IEEE.

[22] Chen, H., Zuo, J., Zhu, Y., Kabir, M. R., & Han, A. (2024). Generalizable Deep Learning for Pulse-echo Speed of Sound Imaging via Time-shift Maps. In 2024 IEEE Ultrasonics, Ferroelectrics, and Frequency Control Joint Symposium (UFFC-JS) (pp. 1-4). IEEE.

[23] Huang, Tianyou. "Construction and Application of Financial Risk Warning System for Third-Party Payment Enterprises Based on Z-Score Model and Efficacy Coefficient Method." Academic Journal of Business & Management 6.12 (2024): 121-126.

[24] Liu Y. Optimization of Financial Fraud Detection Algorithm by Combining Variational Autoencoder with Generative Adversarial Networks[C]//2024 International Conference on Distributed Systems, Computer Networks and Cybersecurity (ICDSCNC). IEEE, 2024: 1-7.

[25] Zhao F. Application and Performance Improvement of K-Means Algorithm in Collaborative[C]//2025 International Conference on Intelligent Systems and Computational Networks (ICISCN). IEEE, 2025: 1-6.

[26] Zhao, Fengyi "Development Design and Signal Processing Algorithm Optimization of Traditional Chinese Medicine Pulse Acquisition System Based on CP301 Sensor." Advances in Computer, Signals and Systems (2024), 8(6): 106-111