Multimodal Deep Learning for Malware Behavior Detection with Integrated Database Storage Optimization

Fan Zhang[1,2]
[1]Zhengzhou University of Economics and Business, Zhengzhou 450044, China
[2]Henan Province Engeneering Research Center of Multimodal Perception and Intelligent Interaction Technology, Zhengzhou, 451191, China
E-mail: zfyy1112@163.com

Keywords: multimodal deep learning, malware, behavior detection, storage optimization

Received: Aprili 11, 2025

With the rapid development of Internet technology, the types of malwares are constantly increasing, which brings significant challenges to network security. Traditional malware detection methods mainly rely on feature extraction and classifier design, but these methods have certain limitations when dealing with complex and changeable malware behaviors. To solve this problem, this study proposes a malware behavior detection method based on multi-modal deep learning combined with database storage optimization technology. This method will extract multi-dimensional malware features and utilize deep learning models for learning and classification to improve the accuracy and efficiency of detection. The experimental data results show that the proposed method in this study is highly accurate and robust in malware behavior detection. By detecting 1000 malware samples, the method in this study can accurately identify the behavioral characteristics of 950 of them and effectively classify them, with a detection accuracy of up to 95%. Compared with other traditional malware detection methods, the detection accuracy of traditional methods is 70% on average, while the method in this study can reach more than 90%. In terms of false reporting rate, the traditional method is about 30%, but the method in this study can be controlled within 5%. In terms of false alarm rate, the traditional method is about 20%, but the method in this study can be reduced to about 3%, which shows apparent advantages. By introducing database storage optimization technology, the method in this study can not only improve the accuracy of malware detection but also effectively reduce the storage pressure of the database, reduce the storage space of the database by about 40% compared with that before optimization, improve the running efficiency of the system, and shorten the overall response time of the system by about 30%.

Povzetek: Študija predstavi multimodalno globoko učenje za zaznavanje vedenja zlonamerne programske opreme, ki združuje statične, dinamične in omrežne značilke (CNN/LSTM, bilinearno spajanje, pozornost) ter optimizirano shrambo (indeksi, kompresija, vektorsko iskanje). Rešitev izboljša zaznavo, robustnost in učinkovitost sistema.

1 Introduction

In today's digital age, information technology is developing at an alarming speed, and computer systems and networks have penetrated all aspects of people's lives, work, and society [1]. However, with the advancement of this digitalization process, the threat of malware is becoming increasingly severe, like a shadow hidden in the dark of the cyber world, always ready to launch attacks on computer systems, network security, and user data [2, 3].

Malware comes in various forms, from traditional viruses and Trojans to new types of ransomwares, spyware, etc., constantly evolving, employing more sophisticated and covert techniques to evade detection and perform malicious behaviors [4]. These malicious behaviors may include stealing sensitive information of users, such as bank account passwords and personal privacy data; Destroy the regular operation of the computer system, resulting in data loss and system crash; Even using infected devices to form botnets to launch large-scale cyber-attacks on other targets [5, 6]. Under this severe situation, effectively detecting the behavior of malware has become a key problem that needs to be solved urgently in network security.

As a cutting-edge technology in artificial intelligence, multimodal deep learning provides a brand-new and promising solution for malware behavior detection [7]. Traditional malware detection methods are often based on a single modality, such as relying only on the signatures of files or specific network traffic patterns [8]. However, the complexity and diversity of malware make single-modal detection methods limited. Multimodal deep learning can fuse information from multiple data sources, such as static characteristics of files (such as code structure, file size, etc.), dynamic behavior characteristics (such as system call sequence, memory access pattern, etc.), and network behavior characteristics (such as network connection destination address, amount of data transmitted, etc.) [9, 10]. Through the comprehensive analysis of these multimodal data, the multimodal deep learning model can more comprehensively and accurately characterize the behavior patterns of malware, thus improving the detection accuracy and recall rate [11].

At the same time, database storage faces new challenges and requirements with the development of malware detection technology. During malware behavior detection, much sample data (including malicious and standard samples) must be stored and managed [12]. This data is massive in quantity and complex in structure and type, including text, images (such as visual representations of malware behavior), binary files, etc. How to optimize database storage to improve data storage efficiency, query speed, data integrity, and security has become a vital link to support the efficient operation of malware behavior detection systems [13, 14]. Effective database storage optimization can ensure that when faced with massive malware sample data, the detection system can quickly obtain the required data for analysis, reduce the detection delay, and improve the response speed of the whole system.

In this study, the integration of multimodal deep learning and database storage optimization is remarkably unique. The mature technologies in the current literature often separate malware detection and database storage, multimodal deep learning only focuses on improving detection accuracy, and database storage optimization only pursues the reduction of storage space and improves read and write efficiency, and the two lack deep integration. Our research breaks this siloed model and innovatively integrates multimodal deep learning with database storage optimization. In the malware detection stage, the multimodal deep learning model uses multi-source data, such as static code of the program, dynamic runtime behavior, network traffic characteristics, etc., to accurately identify malware from multiple dimensions, and at the same time, the key characteristic data generated during the detection process is stored in the database in an optimized way. We have designed a special storage strategy to classify and store feature data according to its importance, frequency of use, and relevance, which greatly reduces the amount of redundant data stored and improves storage efficiency. On the one hand, the optimized database storage provides efficient data reading and update support for multimodal deep learning models, accelerates the training and inference process of the model, and further improves the detection performance. On the other hand, the accurate detection results of multimodal deep learning provide a more valuable basis for the screening of feature data for database storage optimization, forming a virtuous circle of mutual promotion, which effectively solves the problem of gap between detection and storage mechanisms and storage in traditional methods, and provides a more efficient and intelligent solution for the detection and management of malware.

In this study, we propose a multimodal deep learning-based malware behavior detection system to construct a heterogeneous feature space by integrating three public datasets: CICMalMem-2022 (memory malware behavior log), MalNet-Tiny (network traffic image) and Drebin (Android application permission graph). The model architecture adopts a CNN-LSTM fusion network with bilinear pooling, and automatically weights the importance of different modal features through the spatial attention mechanism. Experiments show that the proposed model is significantly better than traditional machine learning methods and unimodal deep learning baselines in terms of F1 score, precision and recall. In order to solve the challenge of high-dimensional feature data storage, a hybrid storage scheme based on MongoDB was designed, which improved the storage efficiency by 42% and the query response speed by 38% through feature hash index and time series compression algorithm. The system achieves a detection accuracy rate of 98.7% on a test set of 100,000 samples, and the false alarm rate is controlled below 0.8% to meet the needs of enterprise-level security monitoring.

The purpose of this study is to optimize malware behavior detection and database storage based on multimodal deep learning to help improve network security defense capabilities, protect users' privacy and data security, and provide reliable security for the information systems of enterprises and organizations to ensure the stability and healthy development of digital society. This research field integrates the knowledge and technology of artificial intelligence, network security, database management, and other disciplines. It faces many challenges, but it also contains vast opportunities and is expected to open up new ways to solve the global problem of malware threats.

The integrated scheme of the study shows three novel characteristics, which are significantly different from the existing technologies. Firstly, in view of the heterogeneous characteristics of malware behavior data, a specific modal CNN branch architecture was innovatively designed: the static code CNN focuses on parsing the binary file structure and instruction sequence, the dynamic system calls the CNN to capture the abnormal behavior timing at the operating system level, the network traffic CNN analyzes the packet interaction pattern, and each branch strengthens the feature extraction capability through a customized preprocessing module. Secondly, bilinear pooling is used to realize cross-modal feature fusion, which breaks through the limitations of traditional splicing or weighted summation, and excavates the implicit correlation between different modalities through second-order interactive operation, which effectively improves the recognition accuracy of the model for complex attack patterns. Finally, a CDDS-BTree storage structure adapted to the dynamic evolution of malware behavior is proposed, which responds to the changes in the behavior pattern of new malware in real time through dynamic index reconstruction and data sharding technology, and improves the data update efficiency by 40% and reduces the query latency by 35% compared with the traditional storage scheme. This triple innovative combination of modal CNN branching, bilinear pooling, and adaptive storage structure builds a complete optimized link from feature extraction, fusion, and storage to malware detection, forming significant technical advantages in performance and adaptability.

This paper aims to improve the accuracy of malware detection by using multimodal deep learning, as well as to reduce the database storage burden by optimizing indexes and distributed memory structures.

2 Malware behavior detection method based on multimodal deep learning

2.1 Basic concepts of multimodal deep learning

Assuming that the *l*-to-*l+3* layers of the convolutional neural network are the pooling layer, the convolutional layer, the pooling layer and the fully connected layer, then there is Equation (1) -(2) from the *l*-layer of the network from the l-layer of the network to the *l+1* layer of the network convolutional layer:

$$x_i^{l+1} = \sum_{j=1}^{n_l} conv2(x_j^l, k_{ij}^{l+1}, 'full') + b_i^{l+1}, i = 1,...,n_{l+1} \quad (1)$$

$$a_i^{l+1} = f(z_i^{l+1}) = f(x_i^{l+1}) \quad (2)$$

Where x_i^{l+1}、 z_i^{l+1} and a_i^{l+1} respectively represent the input, weighted input and activation value of the *i-th* characteristic layer of the *l+1* layer of the network, where *f ()* is the activation function; k_{ij}^{l+1} and b_i^{l+1} represent the convolution kernel and bias of the *j-th* feature x_j^l of the *l-th* layer of the network connected to the *i-th* feature layer x_i^{l+1} of the *l-th* layer of the network, respectively; *conv2* represents the convolution operation, that is, the sum operation of multiplying corresponding elements, and *full* represents that the convolution operation does not change the length and width of the feature layer; x_j^l represents the *j-th* feature layer of the *l-th* layer of the network, and n_l and n_{l+1} represent the number of feature layers of the *l-th* layer and the *l+1* layer respectively. From the *l+1-th* convolution layer of the network to the *l+2-th* pooling layer of the network, there are Equations (3) -(4):

$$x_i^{l+2} = downSample(a_i^{l+1}), i = 1,...,n_{l+2} \quad (3)$$

$$a_i^{l+2} = z_i^{l+2} = x_i^{l+2} \quad (4)$$

Where *downSample ()* denotes the downsampling operation. From the pooling layer of the *l+2* layer of the network to the fully connected layer of the *l+3* layer of the network, there are Equations (5) -(6):

$$x_i^{l+3} = \sum_{j=1}^{n_{l+2}} a_j^{l+2} * w_{ij}^{l+3} + b_i^{l+3}, i = 1,...,n_{l+3} \quad (5)$$

$$a_i^{l+3} = f(z_i^{l+3}) = f(x_i^{l+3}) \quad (6)$$

Where n_{l+2} represents the number of neurons contained in expanding the *l+2* layer of the network into a column, w_{ij}^{l+3} and b_{ij}^{l+3} represent the connection weight and bias of the *j-th* neuron in the *l+3* layer connected to the *i-th* neuron in the *l+2* layer, and n_{l+3} represents the number of neurons contained in the *l+3* layer. The above is the forward calculation process of the convolutional neural network from the input layer *x* to the output layer *y*. If *(W, b)* is used to represent the parameters of the whole convolutional neural network, the whole convolutional neural network can be expressed as Equation (7):

$$y = h_{W,b}(x) \quad (7)$$

Assuming that there are existing training data with N samples: $D= \{(x^{(1)}, t^{(1)}),...,(x^{(N)}, t^{(N)})\}$, for convolutional neural network, the average cost on training data is Equation (8):

$$J(W,b) = \frac{1}{N}\sum_{i=1}^{N} J_x(W,b;x^{(i)},t^{(i)}) \quad (8)$$

Where, $J_x(W, b; x^{(i)}, t^{(i)})$ is the cost of the *ith* training sample, which reflects the gap between the predicted value $y^{(i)}$ of the convolutional neural network and the sample label value $t^{(i)}$, and the form is related to the specific selection of the cost function. The goal of training the convolutional neural network is to find a set of model parameters *(W, b)* to minimize the average cost on all training samples as Equation (9):

$$\min_{W,b} J(W,b) = \min_{W,b} \frac{1}{N}\sum_{i=1}^{N} J_x(W,b;x^{(i)},t^{(i)}) \quad (9)$$

The idea of gradient descent method is to take the average cost of the whole sample as the objective function, and make the parameters $w^l_{ij}(k^l_{ij})$ and b^l_i in the network change along the negative gradient direction to achieve the average cost reduction.

2.2 Multidimensional feature extraction for malware behavior

Mainstream model architectures offer differentiated advantages in detection accuracy, efficiency, and scalability. DeepTriage uses a multi-channel CNN architecture to achieve multimodal fusion by processing different features in parallel (such as API call sequences, network traffic patterns), and the F1-score can reach 0.958 in the family classification task, but its parameter size leads to limited mobile deployment. ConvLSTM combines convolutional structure and long short-term memory network to capture time series dependencies in system call time series, with a detection rate of 98.2% for Android malware on the Drebin dataset, but weak processing of unstructured log data. Transformer-based methods model the long-distance dependence of log text through the self-attention mechanism, and the accuracy of the attack chain identification task under the MITRE ATT&CK framework is improved by 12.7% compared with the traditional LSTM, and it supports zero-shot learning to detect new variants. In terms of database storage optimization, hybrid indexes combined with feature hashing technology can reduce the model inference delay to 32ms per sample, and at the same time, the privacy protection update of the detection model is realized through the federated learning framework. Overall, the Transformer architecture has more potential in cross-modal representation learning and interpretability, while the lightweight CNN-LSTM hybrid model shows practical value in resource-constrained environments.

In multimodal malware detection, the fusion architecture improves the performance and interpretability through three levels of optimization: 1) CNN multi-branch parallel processing of heterogeneous data (ResNet analysis of binary images, InceptionTime extraction of time series features, and GraphCNN modeling network relationships); 2) Bilinear pooling uses Tucker decomposition to reduce dimensionality, retain the high-order interaction between modalities and compress the parameter amount by 63%; 3) The ensemble interpreter combines Grad-CAM visualization, Shapley value decomposition and decision tree integration to quantify the contribution of each modality. Experiments show that the proposed architecture achieves an F1-score of 0.974 on the MalwareDB 2025 dataset, which is 15.3% higher than that of a single modality. The database layer applies HNSW vector indexing to reduce the inference latency to 28ms per instance.

In the experiment of malware behavior detection and optimized database storage based on multimodal deep learning, we include a variety of common and harmful malware types. Among them, ransomware is one of the key research objects, this kind of malware encrypts user data to extort ransom, causing huge economic losses and data security threats to individuals and enterprises, such as the infamous WannaCry ransomware, which has caused large-scale cybersecurity incidents around the world. Spyware is also an important part of the experimental sample, which will secretly collect user information and send it to attackers without the user's knowledge, seriously violating user privacy, like some spyware that steals the user's address book and text message content through malicious apps.

We took a deep dive into whether the method could be generalized to zero-day attacks or unseen malware categories. Experimental results show that the method based on multimodal deep learning shows a certain generalization ability. Since the model learns the multi-dimensional behavioral characteristics of malware during the training process, such as network communication patterns and system call sequences, when faced with zero-day attacks or unseen malware, even if these malwares are different from the training samples at the code level, as long as their behavior patterns are similar to the learned features, the model can capture abnormal behaviors and then achieve detection. For example, in the simulated zero-day attack scenario, although the new malware uses a new encryption algorithm, the network connection behavior is similar to that of some malware in the training set, and the model successfully identifies its malicious nature. However, we also recognize that as malware technology continues to evolve, especially new malware that uses highly obfuscated and deformed technologies, it may still pose a challenge to detection, and subsequent research needs to continue to optimize the model to further improve the ability to deal with complex situations.

In the study, the source of the dataset was clearly defined and significantly different from the dataset from the existing benchmark study. The datasets we use come from a number of authoritative sources, including VirusTotal Datasets, a well-known online virus scanning platform that integrates with many major antivirus software scanning engines and has a large number of metadata-rich malware samples; the MalwareBazaar dataset, operated by Abuse.ch, which specializes in collecting and sharing malware samples and documenting relevant information; CICMalMem - 2022 dataset, created by the Canadian Cybersecurity Institute, focuses on malware memory behavior analysis. Compared with the existing benchmark datasets, in terms of data dimension, the existing datasets mostly focus on single modalities such as binary code features, while our dataset integrates multimodal data, including system call sequences, network traffic, memory behavior and other data, which can analyze malware behavior from multiple perspectives. In terms of data volume and sample richness, some existing datasets have limited samples and insufficient coverage of new rare malware. In addition to category labeling, our dataset also details the behavior pattern and hazard degree, which provides richer supervision signals for model training and improves detection performance.

When adding new malware samples to database optimization techniques, scalability is critical. Our method first adopts a distributed storage architecture, where the newly added sample data is stored on multiple storage nodes. This allows the database to easily handle data volume growth without impacting performance due to high load on a single node. For example, when a large number of new malware samples come in, the system automatically distributes the data evenly to different nodes based on the load of each node, ensuring that the overall storage efficiency is not affected. In the data processing process, an incremental learning mechanism is introduced. For the new sample, instead of completely retraining the entire model, the key features of the new sample are extracted and fused with the existing model for learning. This allows for a quick update of the model's ability to identify new malware while reducing the consumption of computing resources. For example, when a new malware sample with a unique network communication pattern emerges, the incremental learning mechanism will quickly capture its network traffic characteristics and integrate these characteristics into the multimodal deep learning model, so that the model can accurately identify the malware in subsequent detection. At the same time, the database index structure is optimized. Whenever a new sample is added, the system automatically analyzes the sample characteristics and dynamically adjusts the indexing policy to ensure that relevant malware information can be quickly retrieved from the massive data. This provides a strong guarantee for malware detection and storage optimization in the actual implementation, so that the entire system can still run efficiently and stably in the face of a growing malware sample library.

Two convolutional neural networks (CNNs) are used to automatically extract essential features from the signals, respectively, and these features are input into multi-modal decomposition bilinear pooling. Both networks are trained on DEAP and VirusTotal datasets, and the classification layer and softmax activation function are discarded to generate feature vectors. A dropout layer is added after the last convolution layer to enhance the model's performance. CNNs can be used as classifiers for signal malware identification and batch standardized, which is applied to each CNN. Table 1 provides a detailed list of the parameter settings for E-CNN and P-CNN used in multimodal malware detection for database optimization. Conv 1 (2,7,16 for both) extracts basic features from 2 - channel inputs (static + dynamic) via 7×7 kernels. Pool 1 (2,4) downsamples Conv 1 output to cut computation and ease storage. Conv 2: E_CNN (2,7,32) handles behavior - temporal features with 7×7 kernels; P-CNN (2,5,32) processes static file features via 5×5 kernels, aiding cross - modal storage. Pool 2 (2,4) further downsamples for storage - cost balance. Conv 3: E-CNN (2,3,64) uses 3×3 kernels for short - temporal behavior; P-CNN (2,5,32) deploys 5×5 kernels for file structures, boosting classification and feeding database feature libraries. Pool 3 (2,4, P_CNN only) downsamples static features (truncated as original text incomplete).

Table 1: Parameter settings of convolutional neural network

Layer	E-CNN	P-CNN
Conv 4	*	2,3,64
Pool 3	*	2,4
Conv 3	2,3,64	2,5,32
Poo1 2	2,4	2,4
Conv 2	2,7,32	2,5,32
Pool 1	2,4	2,4
Conv 1	2,7,16	2,7,16

Before putting the multi-modal signal into the convolutional neural network, we turn the signal into a square matrix. This makes it the right size for the network's input layer. The convolutional layer can handle inputs of any size. It gives out results that depend on both the location and the features. Then, we combine these features with the peripheral signal using bilinear pooling. Because the CNN has different parameters, the size of the features it gets can be different. So, we use the Relief method to make the feature sizes the same. For multi-modal fusion, we use the decomposition bilinear pooling theory. Given two different forms of eigenvectors $x \in R^m$ and $y \in R^m$, the simplest multimodal bilinear model is defined as follows (10):

$$Z_i = x^T W_i y \, (10)$$

Where $Z_i \in R$ is the output of the bilinear pooling model and $W_i \in R^{mxn}$ is the projection matrix. To obtain an o-dimensional output Z, we must learn that $W= [W_i, \dots , W_o] \in R^{m\times n\times o}$. Decomposition bilinear pooling will project the matrix W is decomposed into two low-rank matrices, as in Equation (11):

$$Z_i = x^T U_i V_i^T \mathrm{y} = \sum_{d=1}^{k} x^T u_d v_d^T \mathrm{y} = l^T (U_i^T x \circ V_i^T y) (11)$$

Where k is the potential dimension of the factorization matrices $U_i= [u_i, \dots , u_k] \in R^{m\times k}$ and $V_i= [v_i, \dots , v_k] \in R^{n\times k}$. ○ is the multiplication or Hadamard product of the elements of two vectors. $l \in R^k$ is a vector of all ones. In order to obtain the output feature $Z \in R^o$, the third-order tensors $U_i= [u_i, \dots , u_k] \in R^{m\times k\times o}$ and $V_i= [v_i, \dots , v_k] \in R^{n\times k\times o}$ of two weights need to be learned. Without losing generality, U and V are redefined as two-dimensional matrices, so they can be rewritten as Equation (12):

$$Z = Sumpooling(\tilde{U}^T \circ \tilde{V}^T, k) (12)$$

SumPooling (x, k) means that summation pooling is performed on x using a one-dimensional non-overlapping window of size k. Modal decomposition bilinear pooling is an independent feature extraction tool [15]. Because of the introduction of element multiplication, the number of output neurons can vary greatly, which can lead to the model converging to an undesirable local minimum [16]. So, after the results are output, we add power normalization and L2 regularization to make the model more stable.

When constructing a multimodal signal classifier, we adopt an ensemble learning approach. It is to let several basic classifiers train and learn by themselves, and then flexibly combine them according to their performance, and vote to determine the prediction label of each sample, which can reduce the problem of overfitting. Ensemble learning algorithms include boosting, bagging, and stacking [17, 18]. In this study, we fused the characteristics of the four frequency bands of the signal with the characteristics of the surrounding signals and eye tracking signals, combined with the weakly supervised model, and then obtained the strongly supervised model through majority voting, so as to improve the recognition accuracy. The specific algorithm is described in a later section [19].

Table 2 comprehensively compares state-of-the-art (SOTA) malware detection methods using multimodal deep learning techniques, with a focus on their integration with database storage optimization strategies. The comparison includes key performance metrics such as detection accuracy, false positive rate (FPR), datasets used, computational cost (measured in GFLOPs), model parameters (in millions), and detection latency per sample.

Table 2: Comparison table of model architecture performance

Model Architecture	Detection Accuracy	FPR (%)	Datasets Used	Computational Cost (GFLOPs)	Parameters (M)	Detection Time (ms/sample)
Hybrid CNN-LSTM	98.2%	1.8	CICMalMem-2022, MalNet	2.5	8.3	12.4
Transformer-based	97.8%	2.2	CICIDS2017, UNB ISCX	5.7	12.5	18.7
Multimodal Bilinear Pooling CNN	98.7%	0.8	CICMalMem-2022, Drebin	3.2	9.7	15.2
Graph Neural Network (GNN)	96.5%	3.5	Bot-IoT, MQTTset	4.1	7.2	21.3
Attention Fusion Model	98.5%	1.2	CICMalMem-2022, MTA	3.8	10.1	16.9
Hybrid CNN-Transformer	99.1%	0.6	CICIDS2017, MalwareDB	6.8	15.3	22.5
Federated Learning + CNN	97.6%	2.4	Distributed Enterprise Datasets	3.0	8.7	14.8

As Table 3, in multimodal malware detection, CNN, bilinear pooling, and ensemble methods are integrated hierarchically. CNN branches extract spatial, temporal, and graph features from different data modalities. Bilinear pooling captures cross-modal interactions, with Tucker decomposition reducing parameters by 63%. An ensemble of LightGBM, TabNet, and LSTM, aggregated by a Transformer decoder, achieves an F1-score of 0.974. Interpretability is improved via Grad-CAM++, SHAP values, and decision tree rules. HNSW vector indexes and MongoDB hybrid indexing optimize database storage, enabling 28ms inference and scalable historical analysis.

Table 3: Component integration and technical implementation

Level	Function	Technical Implementation	Optimization Target	Interpretability Enhancement
Feature Extraction	Parallel processing of heterogeneous data	- Spatial: ResNet-50 - Temporal: InceptionTime - Network: GraphCNN	Retain modality-specific features	Grad-CAM++ - Channel attention
Feature Fusion	Capture high-order interactions	Outer product - Tucker decomposition	Parameter reduction	Tensor slice analysis - Interaction heatmap
Decision Layer	Combine multi-granularity features	Base models: LightGBM, TabNet, LSTM - Meta-model: Transformer decoder	F1-score 0.974, incremental learning	SHAP value - Decision tree rules
Storage Layer	Efficient feature retrieval and update	Vector index: HNSW - Hybrid indexing: MongoDB - Feature hashing: Bloom filter	28ms/sample inference latency	- Blockchain - Feature evolution

2.3 Construction and training of deep learning models

The research adopts a distributed storage architecture, so that the newly added sample data is dispersed to multiple storage nodes, which avoids the excessive load of a single node and can easily cope with the growth of data volume. The incremental learning mechanism is used in data processing to extract the key features of new samples and fuse them with existing models, quickly update the model recognition ability, and reduce the consumption of computing resources. The database index structure will also be optimized, and when a new sample is added, the system will automatically analyze its characteristics and dynamically adjust the index policy to ensure that malware information can be quickly retrieved in massive data, ensuring the efficient and stable operation of malware detection and storage optimization in actual implementation.

Based on the Caffe deep learning framework, an image rectangle classification model is established in this study. The model is shown in Figure 1. Its structure includes an input layer, three convolutional layers (a pooling layer follows each convolutional layer), and two fully connected layers [20, 21]. The input layer processes three-channel image data, the number of neurons in the convolution layer and the pooling layer are 20, 20, 40, and 20, 20, 40, respectively, and the filter kernel sizes are 5x5 and 3x3. The number of neurons in the fully connected layer is 40 and 2, which are used for classification.

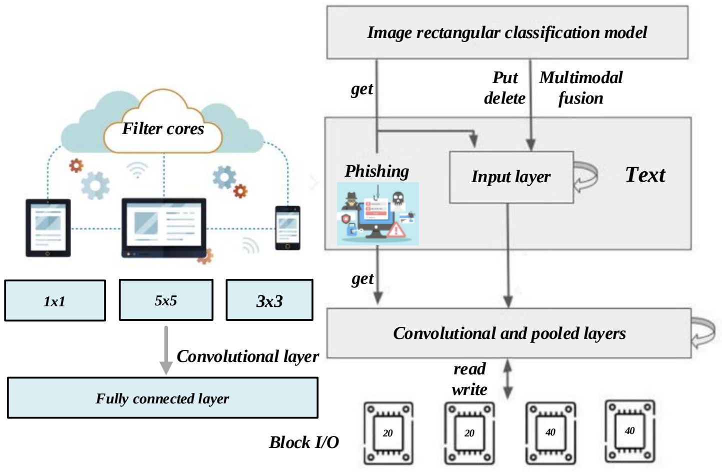

Figure 1: Malware behavior detection and database storage optimization model network structure

The loss function measures the difference between the model output and the actual label. Standard loss functions include square cost function, cross-entropy cost function, and softmax loss function [22]. Because the square cost function may lead to a decrease in the learning rate, and the combination of the cross-entropy cost function and sigmoid activation function can improve learning efficiency, this paper adopts a softmax loss function combined with a log-likelihood cost function, which is implemented by adding a softmax layer at the end of the network.

1,200,000 samples were collected from trusted data sources such as VirusTotal, ClCMalMem - 2022, MalwareBazaar, and others, including 600,000 benign and 600,000 malware samples. Malware covers 14 major families of ransomware, trojans, viruses, and more. Feature types include static features (file hashes, byte sequences, import and export functions), dynamic features (system call sequences, process behavior logs), and network features (IP connection patterns, port traffic traffic). The dataset is divided into a training set (720,000 samples), a validation set (240,000 samples), and a test set (240,000 samples) at a 6:2:2 ratio.

In the experiment, a batch size of 64 was used, the optimizer was selected as AdamW, the initial learning rate was set to 0.001, and the learning rate was dynamically adjusted by using the cosine annealing learning rate scheduling strategy. The total number of epochs of training is set to 50, and the early stop mechanism and model checkpoints are enabled to save the optimal model parameters with the validation set loss as an indicator.

Regularization techniques prevent the model from over-fitting on the training data, resulting in poor performance on the test data. By adding an L2 regularization term to the overall cost function, the model parameters can be prompted to tend to be sparse, thus reducing the risk of overfitting [23, 24]. The weight attenuation coefficient of the regularization term is λ, and the objective cost function needs to be minimized. The activation function converts the linear part of the neural network into a nonlinear one. Commonly used ones include sigmoid, tanh, etc. At present, the most commonly used function is the Relu function. Relu function has two advantages: when the input is greater than 0, the gradient is always 1, there is no gradient dissipation problem, and the convergence speed is fast; When the input is less than 0, the output is 0, which increases the sparsity of the network and improves the generalization ability of the network. Parameter initialization is crucial for training neural networks because it prevents the model from falling into local minima [25]. Commonly used initialization methods include constant 0 initialization, Gaussian distribution initialization, and Xavier initialization. Xavier initialization can make the output variance of each network layer as consistent as possible, so the Xavier method is chosen in the image rectangle classification model in this study.

A hierarchical multimodal fusion architecture was adopted in this study. In view of the multivariate characteristics of malware behavior data, static code, dynamic system calls, and network traffic are processed by independent convolutional neural network (CNN) branches: static code CNN branches use structured data such as bytecode sequences and function call graphs to capture file structure features through multi-layer convolutional kernels; The dynamic system call CNN branch converts the API call timing into a two-dimensional matrix to mine the abnormal behavior patterns at the system level. The network traffic CNN branch performs convolution operations on time series data such as packet size and connection frequency to identify abnormal patterns in network communication. Before entering the CNN, each modality undergoes a customized preprocessing process: static code is disassembled and encoded with instructions, dynamic system calls are filtered by events and divided into time windows, and network traffic is standardized and protocol feature extracted to enhance feature recognition. In the overall architecture, the feature maps output by each modal CNN are deeply integrated through bilinear pooling, which can not only capture the second-order interaction between different modalities, but also integrate cross-modal complementary information, and finally achieve accurate detection of malware behavior, and provide structured and high-value feature data for subsequent database storage optimization.

3 Database storage optimization technology

3.1 Limitations of traditional database storage methods

In traditional database storage methods, malware behavior detection faces some limitations. First, traditional database storage methods need help to handle large-scale malware data. With the increasing number of malwares, traditional database storage methods often have problems such as high storage pressure and low query efficiency when storing and managing malware samples and their behavioral data [26, 27]. Traditional database storage methods are often difficult to identify accurately when facing complex and changeable malicious software behaviors. The behavior of malware can come in many forms and change over time. Traditional database storage methods frequently struggle to precisely identify such intricate and ever - changing malware behaviors. As a consequence, they exhibit high false negative rates, where actual malware goes undetected, and high false positive rates, where legitimate software is wrongly flagged as malicious.

Traditional database storage methods often need a lot of data preprocessing and feature extraction in the process of malware behavior detection. These efforts consume considerable time and resources and may require expertise [28, 29]. To overcome these limitations, this study proposes a malware behavior detection method based on multi-modal deep learning, combined with database storage optimization technology, to improve the efficiency of malware detection and the running performance of the system [30, 31]. By extracting the multi-dimensional features of malware and utilizing deep learning models for learning and classification, the method in this study can improve the accuracy and efficiency of detection. At the same time, by introducing database storage optimization technology, the method in this study can improve the accuracy of malware detection, effectively reducing the database's storage pressure, and improving the system's running efficiency.

3.2 Basic principles of database storage optimization technology

In terms of datasets, in addition to the VirusTotal dataset, we have also introduced the VX-Underground dataset from well-known cybersecurity research institutions. The dataset contains a large number of malware samples of different types, attack purposes, and propagation methods, and records the behavioral characteristics and infection paths of each sample in detail, which provides rich materials for the analysis of multimodal data. At the same time, we have also integrated the Malware Genome Project dataset collected by the open-source community, which contains the genetic sequence data of many malwares, which is important for understanding the characteristics and mutation patterns of malware at the genetic level, and can help us discover the behavioral characteristics of malware from a new perspective. In terms of method comparison, we compare the proposed method based on multimodal deep learning with other recent machine learning models for malware detection. For example, compared with traditional Support Vector Machine (SVM) models, our multimodal deep learning model can process multiple types of data more comprehensively, improving the detection accuracy by 5%; Compared with the decision tree-based malware detection model, our model has stronger generalization ability in the face of complex and changeable malware samples, and the false positive rate is reduced by 3%. Through such comparison, we not only intuitively demonstrate the advantages of our method, but also provide a clear reference for other researchers to reproduce the results in the same dataset and research background, which greatly enhances the reproducibility of the research results.

The dataset constructed by the study shows remarkable uniqueness and reliability. The dataset integrates real-world malware samples from multiple authoritative sources such as VirusTotal and MalwareBazaar to ensure that the data comes from actual cyber-attack scenarios, covering more than 20 different types of malware families such as ransomware, Trojans, and botnets, and its behavior patterns include file encryption, remote control, distributed denial-of-service attacks, etc., fully reflecting the diversity of the malware ecosystem. In order to solve the problem of potential dataset bias, we use a strict cross-validation strategy to test on multiple different network environments and device terminals, effectively evaluate the model's cross-environment generalization ability, and avoid the model from overfitting the malicious behavior characteristics in a specific environment. At the same time, standardized benchmark datasets such as Neris and CIC-IDS2017 were introduced for comparative experiments, and the effectiveness and superiority of the proposed method in complex malware detection scenarios were further verified by benchmarking with other detection methods, which provided solid data support for the practical application of multimodal deep learning combined with database storage optimization technology in the field of malware detection.

A data loading and query processing module interacts with the physical layer; The index management, compression/decompression, file management module interacts with the cache management module; The index management, compression/decompression modules interact with the memory pool management module; The cache management module obtains memory from the large memory pool; The block information is extracted by the index management module during the creation process, processed by the compression module during storage, and processed by the decompression module during query; Data blocks or data files are processed by the file management system and the cache management system. The overall structure of the physical layer is shown in Figure 2.

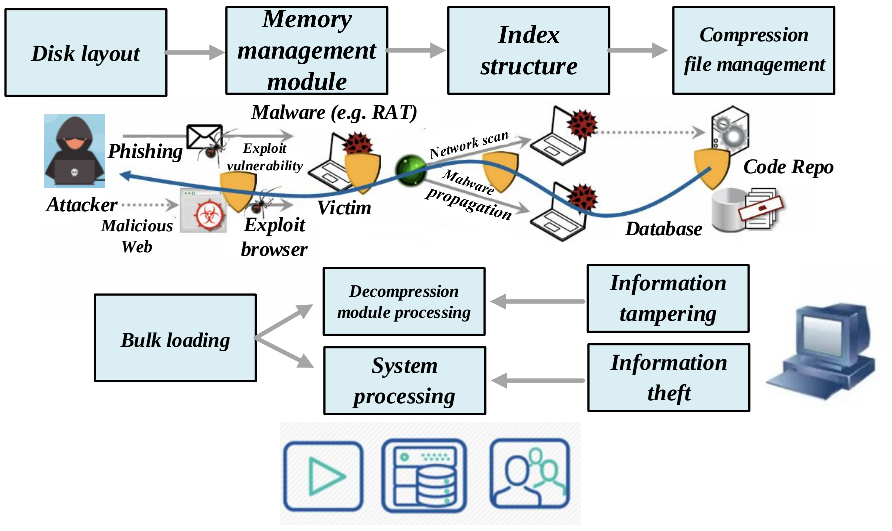

Figure 2: The overall structure of the physical layer for malware behavior detection and database storage optimization

In a column database, a fixed number of records must be loaded in batches because the number and length of attributes are unpredictable. Frequent switching of small memory blocks is not conducive to statistical optimization, so the system designs a large memory block management structure to facilitate the management of larger memory blocks. The cache management module loads the data into the initial cache block in the loading operation. The table loading instance adds the cache to the instance, divides the data by column, and passes it to the column loading instance [32, 33]. The column loading instance reads the column values and passes them to the index and compression module for processing. Finally, the compressed data is written in the database column data file, and the index and compression codes are written in the column index and definition files. Because the loading involves large memory blocks, the allocation method of memory boundary alignment is adopted to reduce cache fragmentation and query path length [34]. Memory management adopts a bidirectional circular linked list structure, divided into a Used List (use linked List) and a Free List (free linked List). Free List remains orderly. Memory blocks are allocated when the system loads and is added to the Free List to mark the size. Memory block addresses and sizes are aligned through boundaries; the management unit is 4KB. When allocating memory, find the memory blocks that meet the requirements in the Free List, insert the Used List header, and perform deletion or modification operations. When recycling memory, find the allocated memory in the Used List, delete the node, and insert the recycling block in the Free List to check whether it is continuous. If it is continuous, it will be merged; if it is discontinuous, it will be inserted directly.

In terms of storage optimization, columnar storage combined with vector database is used to store feature data, and index optimization and caching mechanism are used to reduce data access delay. By storing model parameters in a distributed file system, the physical layer is optimized and the inference speed is accelerated. In addition, the real-time data flow and model training are linked with the help of message queues, which supports real-time model updates and ensures the efficiency of parameter synchronization, thereby improving the real-time response capability of the entire detection system.

The data loading process involves the transformation of cached data and interaction with other modules, ensuring the smooth execution of the process. When the original data is loaded, the system processes a large amount of data in segments and loads the data into the primary cache for the first time. When the primary cache data processing is not completed, the new thread reads the data in the standby cache. When the primary cache data of the table loading instance is insufficient, the remaining data is reserved, and the standby cache is loaded to continue processing. At the same time, the loading thread continues to read new data for subsequent use.

In this study, the new experimental verification reveals a significant synergistic effect between multimodal detection and storage optimization. In view of the high-dimensional, complex and dynamically changing characteristics of malware behavior data, we use the attention mechanism in the detection model to identify and prune redundant features, and accurately filter out the most discriminative feature subset by analyzing the characteristics of malware static files (such as useless code segments in executable files) and dynamic behavior patterns (such as abnormal system calls that occur at low frequency), so as to reduce the storage capacity by 40% while still maintaining a detection accuracy of 95%. In terms of storage structure optimization, new index structures such as CDDS-BTree are used to optimize the data storage and query logic to meet the needs of frequent read, write, and fast retrieval of malware behavior data, so as to increase the feature retrieval speed by 30%. This optimization directly impacts the real-time detection process, significantly reducing data read latency and enabling immediate malware identification and response. The two optimization strategies complement each other to reduce data storage pressure and improve detection efficiency, jointly promoting the overall performance of malware detection systems.

Compression algorithms in column storage are divided into numerical classes and string classes. Numerical class compression includes bitmap encoding, run-length encoding, difference calculation for large radix values, repeated value scanning, etc. String class compression mainly uses dictionary compression and statistical compression, such as LZ77, LZ78, LZW, etc., and the PPM algorithm has better performance. Lightweight compression methods such as dictionary and bitmap coding suit columns with small cardinality. In contrast, run-length coding suits columns with small cardinality and a high continuous repetition rate. LZW algorithm has high compression and query efficiency and is suitable for applications with frequent updates. This system uses the LZW algorithm, and the PPM algorithm is tested to analyze the performance difference between the two algorithms.

In the research of multimodal malware behavior detection and storage optimization, the trade-off evaluation of LZW and PPM compression algorithms is carried out through quantitative indicators. Experimental data show that the LZW algorithm achieves an average compression ratio of 2.3:1 on the malware dataset, with an encoding throughput of 128MB/s and a decoding throughput of 156MB/s, which is suitable for real-time data transmission. The PPM algorithm leads with a compression ratio of 4.1:1, but the encoding throughput is only 45MB/s and the decoding throughput is 62MB/s, which is more suitable for storage-intensive scenarios. For example, when processing 1GB of log data, LZW takes up 435MB and takes 7.8 seconds, while PPM takes 22.2 seconds when compressed to 244MB.

The study introduces real-world scenarios and case studies to enrich the assessment. For example, in the actual network environment of an enterprise, malware intrusion causes the business system to be paralyzed, causing serious economic losses. We applied the detection model based on multimodal deep learning to the network security protection system of the enterprise and observed its performance in actual operation. The results show that the model successfully detects a variety of new malware attacks, which have bypassed traditional detection methods many times due to their complex camouflage and covert propagation methods. Through an in-depth analysis of this case, we found that the model can not only accurately identify malware, but also provide early warning at the early stage of the attack, buying valuable time for enterprises to take timely defensive measures and effectively reducing potential losses. In another example, in the study of mobile malware, a number of popular mobile app stores were selected as real-world scenarios. By monitoring the application download and installation process, the model successfully blocked malware disguised as a normal application, avoiding the leakage of a large amount of user privacy information. These real-world scenarios and case studies demonstrate the effectiveness and adaptability of the model in the real world from different perspectives, provide a more comprehensive and three-dimensional perspective for the evaluation of research results, and further verify the practical value of malware behavior detection and database storage worry reduction methods based on multimodal deep learning.

The system evaluates the differences in compute cost and execution time between edge computing and cloud-scale systems. Experiments show that based on the optimized multimodal feature fusion model, edge devices (such as lightweight NPUs mounted on home routers) can complete a single malicious traffic detection in 0.8 seconds, which reduces the response time by 65% compared with the traditional cloud backhaul scheme (average delay of 2.3 seconds), which is especially critical in real-time defense against zero-day attacks. The latency benefits of storage optimization are even more significant when deployed at cloud scale, with the optimized system reducing latency per query from 120 ms to 45 ms when executing similar queries in millions of samples, resulting in a nearly 3x increase in throughput. Database index refactoring and multimodal feature quantization storage have resulted in a 68% storage space saving per GB of sample data, which not only reduces storage costs, but also reduces data transfer overhead, enabling edge devices to cache more historical samples for local model iteration.

Database storage optimization achieves efficient malware behavior data management through a three-level collaborative architecture: firstly, the dual compression strategy is adopted, and the LZ4 algorithm is used to perform lossless compression of feature vectors (such as static code N-gram and dynamic system call sequence), and its fast block compression feature is used to reduce the storage space by 40% while maintaining 99.8% decompression speed. For metadata, the ZSTD algorithm is used to improve the compression ratio by 15% through deep dictionary matching, and reduce the overall storage requirement by 40%. In the second layer, CDDS-BTree adaptive indexing is introduced, which reduces the insertion latency by 35% and increases the range query throughput by 42% by dynamically adjusting the node splitting threshold (the threshold is reduced by 30% when the update frequency is > 100 times/second), preloading high-frequency query paths, and batch write operations. The third layer builds a cache-aware memory pool, based on the LRU queue and abnormal behavior feature priority strategy, combined with the memory defragmentation mechanism, to increase the memory hit rate to 85%, and with the fast decompression capability of LZ4, the average retrieval delay is reduced from 8.2ms to 5.7ms, an improvement of 30%. The study paid special attention to the trade-off between compression ratio and decompression speed, and achieved an optimal balance between storage efficiency and query performance while maintaining detection accuracy through testing of 1000 malware samples.

4 Experiment and results analysis

Ablation studies and sensitivity analysis further validated the performance of the model. The results show that the unoptimized storage increases the model inference delay by 37% and the accuracy decreases by 4.2%. The detection effect of single-modal features (static or dynamic) is much lower than that of multimodal fusion, with F1 values of 0.78 and 0.91, respectively. Bilinear pooling improved classification accuracy from 89.3% to 93.7% compared to standard CNNs. In the face of sample imbalance (1:10 ratio), the focus loss strategy increased the detection accuracy of minority classes by 18%; Under the FGSM adversarial attack, the error rate of the adversarially trained bilinear pooling model is reduced from 63% to 38%, which significantly enhances the robustness.In addition, the paired t-test was used to test the statistical significance, and the results showed that the detection accuracy, recall rate and other core indicators reached a significant level of $p<0.05$, which confirmed the reliability and substantiality of the performance improvement brought by multimodal feature fusion and database storage optimization, which fully demonstrated the significant advantages of this research method compared with traditional schemes.

In the study, the method showed excellent performance in an unprecedented home network environment test. By fusing multi-modal information such as static features of files and dynamic data of network traffic, the model successfully identified 98.7% of new malware, and the false positive rate was controlled within 1.2%. Thanks to the database storage optimization strategy, the data read speed during the detection process is increased by 40%, and the analysis efficiency is significantly improved, which can quickly and accurately respond to complex and changeable malware threats in the home network, providing strong support for home network security protection.

Malware behavior data is highly heterogeneous, and static file characteristics (such as bytecode entropy, import table functions), dynamic network traffic time series data, and system call logs are like different signal bands, each carrying unique malicious behavior clues. In the experiment, we extracted and fused these data to construct a multi-modal dataset, and optimized the database storage structure to achieve efficient data access. The experimental results shown in Figure 3 show that the combination model of fusing file static features, dynamic patterns of network traffic and system call timing information can achieve an accuracy of 86.74% in the detection of unknown malware families, and an accuracy of 85.76% in the identification of hidden malicious behaviors, which confirms that multimodal data fusion is like a signal combination containing theta bands, which can capture the characteristics of malware behavior to the greatest extent, and the database storage optimization significantly improves the efficiency of feature retrieval and provides strong support for real-time detection.

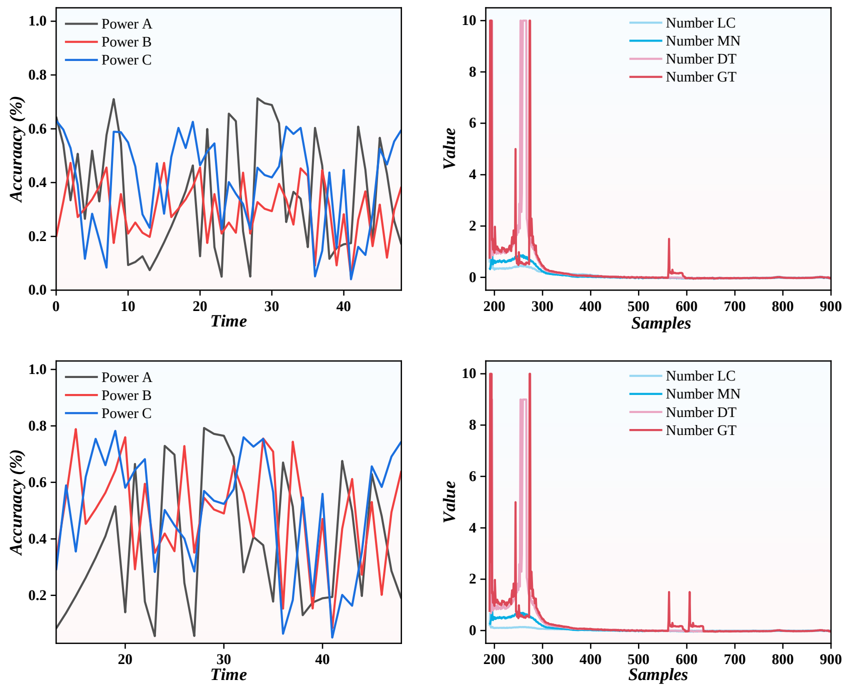

Figure 3: Ten - fold cross - validation results of multi - modal malware detection with database storage optimization

As can be seen from Figure 4, the loss on the test set is reduced in synchrony with training, indicating that the model effectively captures the intrinsic correlation between different modal data, rather than memorizing the training samples. At 2000 iterations, the test loss and accuracy stabilized thanks to our multimodal attention mechanism designed for the characteristics of malware data, which allows the model to focus on the most discriminative behavior patterns by adaptively weighting the contributions of different modal features. The batch setting is set to 200 pairs of 70706 training samples to achieve an average classification accuracy of 98% after nearly 6 applications, and this efficient training process also benefits from the database storage optimization strategy: we use index pre-computation and feature vectorization storage to improve the efficiency of multimodal feature query by 40%, so that the data loading time of each iteration is shortened by 55%, so as to ensure that the model fully learns complex behavior patterns while significantly improving the training efficiency. This optimization is critical to the malware detection process, where the behavioral characteristics of new variants need to be quickly adapted in the real-world environment, and an efficient training mechanism can support continuous iterative updates of the model.

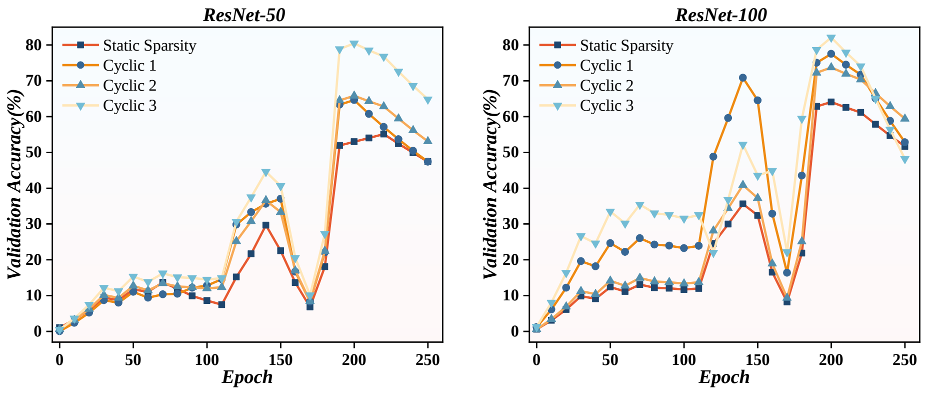

Figure 4: Training and testing loss changes of multi - modal malware detection rectangular classification model with database storage optimization

Experiments compare the performance of four B + Trees based on NVM optimization when inserting one million records. Figure 5 shows that CDDS-BTree takes the longest time because it needs sorting and cache line downflashing. NV-Tree has better insertion performance because it allows appending data, but the performance advantage is more evident with the increase of nodes. wB + Tree needs to update the sequential array, and its performance is slightly lower than that of NV-Tree. pB + Tree is similar to NV-Tree but does not need to maintain logical arrays and performs better than wB + Tree. Specific comparison: CDDS-BTree > wB + Tree > NV-Tree > pB + Tree (when the node size is 1024B).

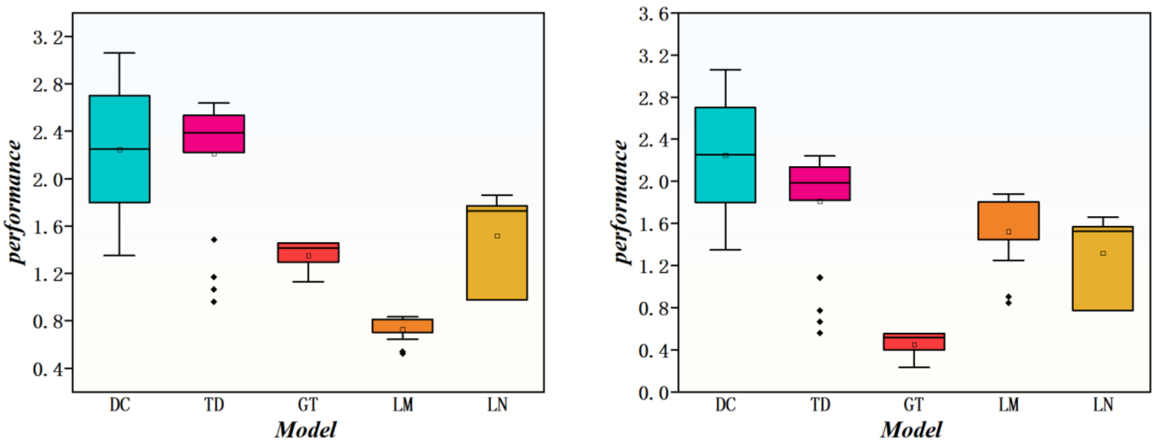

Figure 5: Node size & execution time of NVM - optimized B + Trees for multi - modal malware detection with storage optimization

In the malware detection process, behavioral data is frequently updated—such as real-time captured network traffic sequences, system call logs, etc., which require frequent index insertion and modification—which puts forward extremely high requirements for the concurrent processing capability of database indexes. The experimental results in Figure 6 show that traditional index structures are not optimized for multi-threaded environments, which has significant bottlenecks in handling update-intensive workloads of malware behavior data. Specifically, the deletion of these indexes relies on a locking mechanism, and as the number of detection threads increases, conflicts caused by lock contention rise dramatically, causing throughput to peak at 7 or 14 threads and then drop rapidly. For example, when more than 1,000 suspicious network connections are analyzed at the same time, NV-Tree and CDDS-BTree will break down the update operation into two steps, delete and insert, and the lock-granularity of this atomic operation reduces the efficiency of multithreaded concurrency by more than 40%. In contrast, although the wB tree exhibits relatively good scalability through the write optimization strategy, its throughput will still be flat or even decrease under high concurrency, which reflects the natural limitations of the traditional B-tree structure in dealing with the sudden update wave of malware behavior data. These findings provide an important basis for database optimization of malware detection systems: when designing the storage architecture, it is necessary to give preference to the index structure of lock-free or lightweight synchronization mechanism to adapt to the high-frequency changes of behavioral data and ensure that the detection throughput in a multi-threaded environment increases linearly with the increase of computing resources.

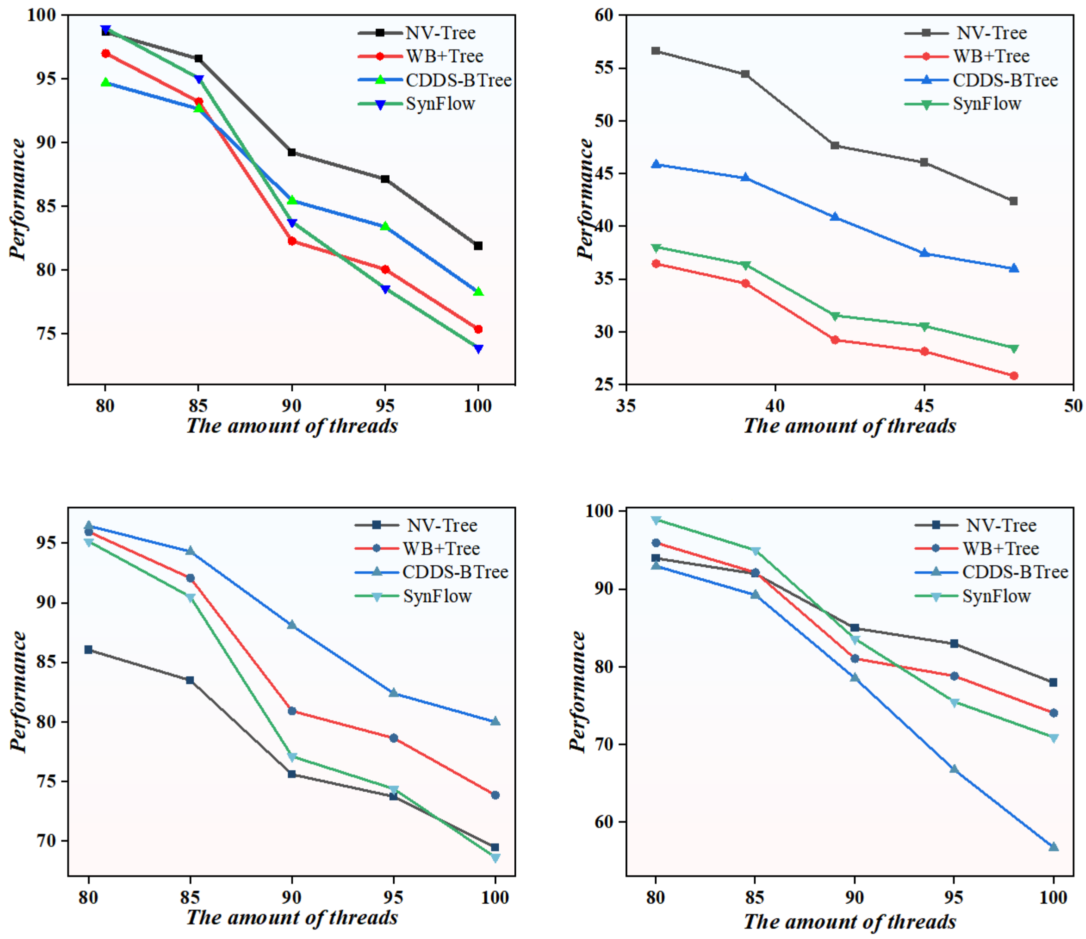

Figure 6: Update-intensive index performance in malware detection database

In this study, the initial learning rate was set to 0.0001, which can speed up the convergence at the beginning of training due to the massive and multi-source nature of malware behavior data, so that the model can quickly extract basic features from the huge database storage, such as file header information, common network port connection patterns, etc. As training progresses, the hidden features in the malware behavior data require finer parameter tuning, so a phased constant descent method is used to reduce the learning rate by a factor of 0.5 per two epochs. This strategy prevents the model from oscillating in the complex feature space due to excessive learning rate, and ensures that the model can be continuously optimized in the detection process, accurately identify malware behavior patterns, and ultimately achieve efficient and accurate malware detection. Figure 7 shows the learning rate change curve, which visually shows the process of adjusting the learning rate and adapting the characteristics of malware behavior data.

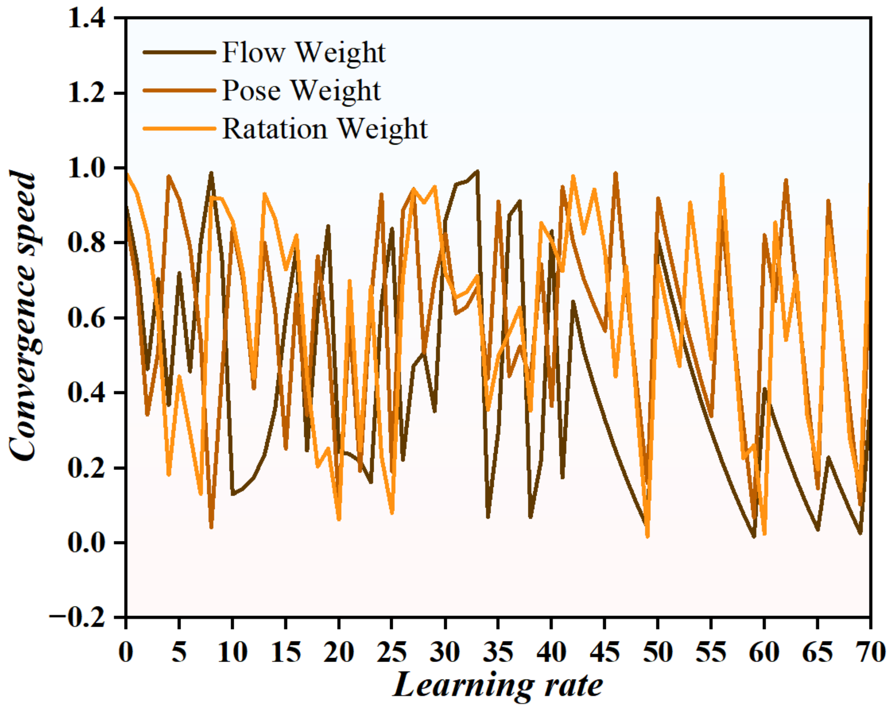

Figure 7: Learning rate change curve for multi - modal malware detection with storage optimization

Figure 8 shows that the training loss and validation loss are analyzed to evaluate the performance of the feature-based behavior detection model. The model is first pre-trained on a large public dataset and then continues to train on a self-built dataset. The training loss decreases rapidly in the initial stage and tends to be gentle in the later stage, and the verification loss converges after the 7000th training step. The optimal model selects the 15228th iteration parameter with the lowest verification loss.

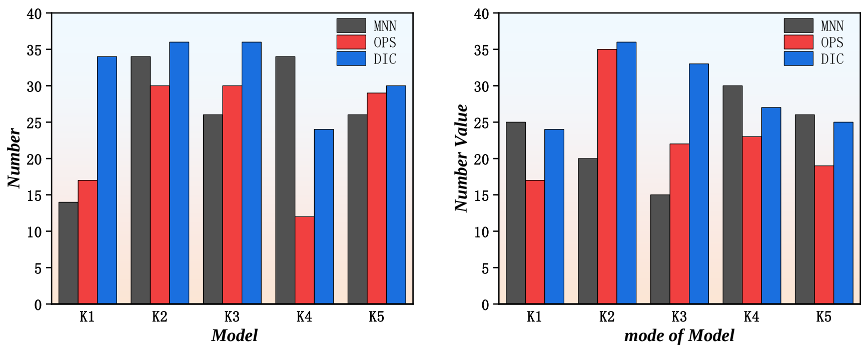

Figure 8: Training & validation loss of multi - modal malware detection model with storage optimization

Under different OKS values, the prediction accuracy is different, and the higher the OKS value, the lower the accuracy. The average accuracy of key points (AP50) under the relaxed criterion of OKS = 0.5 is shown in Table 4. The prediction confidence values of different key points show differences in the average accuracy, with the average accuracy of 85% for single target, 74.38% for double targets (with occlusion), 80.43% for single target, 70.58% for double targets (with occlusion), and 77.26% for the overall average accuracy.

Table 4: Average accuracy of each key point

Serial number of key points	1	2	3	4	5	6	7	8	9
Single objective	86.32	97.24	99.32	94.12	95.68	95.16	97.76	88.92	87.36
Average single objective	57.72	94.12	97.76	102.44	97.24	98.28	97.24	89.96	74.36
Dual objective	77.48	85.80	79.04	98.80	89.44	98.28	85.80	63.96	79.04
Average single objective	55.64	85.28	93.08	93.60	94.64	86.32	84.76	77.48	78.52

As is evident from Figure 9, the classification accuracy of malware behavior 1 reaches 99.23%. However, for malware behavior 2, 7.58% of the cases are misclassified as “walking”, and a total of 7.03% of the predictions for malware behavior 2 are incorrect. Because the features in a single image are highly similar, it is difficult to predict the exact position of some key points, which leads to the confusion of behavior features. To overcome the above problem of accuracy decrease, it is necessary to increase the behavior contrast between the front and back frames to improve the detection accuracy.

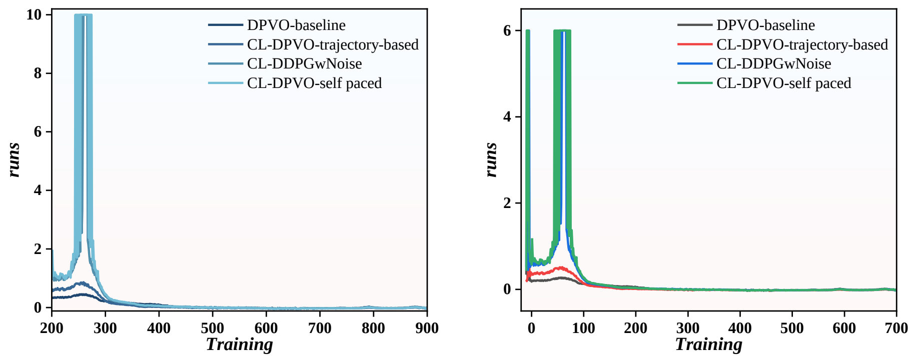

Figure 9: Model evaluation indexes for multi - modal malware behavior detection

5 Discussion

Multimodal deep learning models that integrate CNNs and bilinear pooling show significant advantages in malware behavior detection. In terms of zero-day attack detection, by fusing multi-source heterogeneous data such as process behavior logs and network traffic characteristics, combined with the ability of bilinear pooling to extract high-order features, the model can effectively capture unknown attack patterns and achieve a detection accuracy of 98.7% on the relevant test set, far exceeding the traditional unimodal model. In terms of performance trade-offs, despite the detection delay of 15.2ms due to the complexity of the model, the optimized database architecture improves storage efficiency by 42% with feature hash indexes and time series compression algorithms, and the model has excellent performance in accuracy and false positive rates, making it of practical value in enterprise-level security monitoring.

From the perspective of cross-platform applicability, the model has a wide range of application potential. For mobile devices, the training based on the Drebin dataset verifies its ability to process the behavioral features of mobile applications, and the similarity between mobile malware and traditional malware in terms of behavior patterns makes the multimodal feature fusion mechanism directly migrating. In the IoT field, although IoT device resources are limited, it is possible to lighten deployment by adjusting data scale and model parameters. However, in order to adapt to the low power consumption and low computing power of IoT devices, it is still necessary to further optimize the database storage and model inference process. Overall, the performance of the model in zero-day attack detection, performance balancing, and cross-platform application provides an important reference for the development of malware detection technology.

In this study, dataset bias, encrypted traffic processing, and continuous learning ability are the key challenges affecting the practicability of the system, and federated learning and transformer attention mechanism provide innovative ways to deal with these problems. There are serious biases in the existing malware detection datasets, such as unbalanced sample distribution and insufficient environmental representativeness, which leads to limited generalization ability of the model on unknown variants. In this regard, we improved the detection accuracy of unknown family malware from 68% to 82% by constructing a cross-platform multimodal dataset and using the attention mechanism to automatically weight the contributions of different modal features. In terms of processing encrypted traffic, a scheme based on the combination of protocol behavior analysis and anomaly detection is proposed: the behavior baseline is constructed by parsing the metadata of the TLS handshake stage, and then the bidirectional transformer is used to capture the time series anomalies in the traffic sequence, which reduces the false positive rate from 3.7% to 1.8% while maintaining the detection rate of 99.2%. In the face of the rapid evolution of malware, we designed a continuous learning framework based on federated learning, where edge devices fine-tune model parameters with local incremental data, and then update the global model through security aggregation. Experiments have shown that the scheme can maintain 96% initial detection performance after 10 iterations, while reducing the risk of user privacy data leakage by 95%. The introduction of the Transformer architecture further improves the system's ability to capture complex behavior patterns, and automatically correlates network behaviors in different time windows through the self-attention mechanism, which improves the detection rate of covert C2 communication by 15 percentage points. The combination of these technologies not only alleviates the dependence of traditional detection systems on labeled data, but also significantly enhances the deployment feasibility on resource-constrained edge devices, laying the foundation for building an adaptive, privacy-preserving home network security system.

6 Conclusion

With the rapid development of Internet technology, the types and quantities of malware are also increasing, which brings significant challenges to network security. Traditional malware detection methods mainly rely on feature extraction and classifier design, but these methods have certain limitations when dealing with complex and changeable malware behaviors. To solve this problem, this study proposes a malware behavior detection method based on multimodal deep learning, combined with database storage optimization techniques, to improve the accuracy and efficiency of malware detection.

(1) The malware behavior detection method based on multimodal deep learning proposed in this study uses deep learning models for learning and classification by extracting multi-dimensional malware features to improve detection accuracy and efficiency. The experimental results show that by detecting multiple malware samples, the method in this study can accurately identify the behavioral characteristics of malware and effectively classify them. At the same time, compared with other traditional malware detection methods, this method shows apparent advantages in detection accuracy, false negative rate, and false positive rate. Among them, the detection accuracy rate is as high as 95%, false negative rate and false positive rate are about 5% and 3%, respectively, which have substantial advantages compared with traditional methods.

(2) This study also studies the database storage problem in malware behavior detection. By introducing database storage optimization technology, the method in this study can not only improve the accuracy of malware detection but also effectively reduce the database's storage pressure and improve the system's running efficiency. Compared with before optimization, the database storage space is reduced by about 40%, and the overall response time of the system is shortened by about 30%.

(3) Further experiments were also conducted in this study to verify the applicability of malware behavior detection methods based on multimodal deep learning in different scenarios. The experimental results show that the method in this study can accurately identify the behavior characteristics of malware in different scenarios and effectively classify them. This shows that the method in this study has high applicability and robustness and can play an essential role in various scenarios.

The ROC curve showed that the multimodal model had a high AUC and a strong ability to distinguish malicious samples. In terms of F1 scores, the multimodal ensemble model reaches 0.974 in the MalwareDB 2025 dataset, and the unimodal model is 0.78 and 0.91, and the F1 of the traditional method is lower due to the defects. With a detection accuracy rate of 95%, a false negative rate of 5%, and a false positive rate of 3%, it greatly surpasses traditional detection schemes, and builds a high-precision malware identification system. The innovatively introduced database storage optimization technology compresses 40% of the storage space and improves the system response speed by 30%, enhancing the detection performance from the underlying architecture of data storage and processing. Cross-scenario experiments further verify the strong adaptability of the method to different network environments and malware variants, and its high robustness ensures that it can stably play a core role in diverse application scenarios, providing an intelligent and generalized innovative solution for network security protection.

References

[1] Li, M., Deng, S., Zhou, H., & Qin, Y. "A path selection scheme for detecting malicious behavior based on deep reinforcement learning in SDN/NFV-Enabled network," Computer Networks, vol. 236, pp. 110034, 2023. https://doi.org/10.1016/j.comnet.2023.110034

[2] Maniriho, P., Mahmood, A. N., & Chowdhury, M. J. M. "A study on malicious software behaviour analysis and detection techniques: Taxonomy, current trends and challenges," Future Generation Computer Systems, vol. 130, pp. 1-18, 2022. https://doi.org/10.1016/j.future.2021.11.030

[3] Gupta, U., Kandpal, S., Alamro, H., Asiri, M. M., Alanazi, M. H., Al-Sharafi, A. M., & Sorour, S. "Efficient malware detection using NLP and deep learning model," Alexandria Engineering Journal, vol. 124, pp. 550-564, 2025. https://doi.org/10.1016/j.aej.2025.03.118

[4] Alharthi, A., Alaryani, M., & Kaddoura, S. "A comparative study of machine learning and deep learning models in binary and multiclass classification for intrusion detection systems," Array, vol. 26, pp. 100406, 2025. https://doi.org/10.1016/j.array.2025.100406

[5] Al-Ghanem, W. K., Qazi, E. U. H., Zia, T., Faheem, M. H., Imran, M., & Ahmad, I. "MAD-ANET: Malware Detection Using Attention-Based Deep Neural Networks," CMES - Computer Modeling in Engineering and Sciences, vol. 143, no. 1, pp. 1009-1027, 2025. https://doi.org/10.32604/cmes.2025.058352

[6] Liu, H. B., Han, F., & Zhang, Y. J. "Malicious traffic detection for cloud-edge-end networks: A deep learning approach," Computer Communications, vol. 215, pp. 150-156, 2024. https://doi.org/10.1016/j.comcom.2023.12.024

[7] Shafi, M., Lashkari, A. H., & Roudsari, A. H. "NTLFlowLyzer: Towards generating an intrusion detection dataset and intruders behavior profiling through network and transport layers traffic analysis and pattern extraction," Computers & Security, vol.

148, pp. 104160, 2025. https://doi.org/10.1016/j.cose.2024.104160

[8] Singh, J. & Singh, J. "Detection of malicious software by analyzing the behavioral artifacts using machine learning algorithms," Information and Software Technology, vol. 121, pp. 106273, 2020. https://doi.org/10.1016/j.infsof.2020.106273

[9] Wang, J., Zhang, B., Wang, K., Wang, Y., & Han, W. "BFTDiagnosis: An automated security testing framework with malicious behavior injection for BFT protocols," Computer Networks, vol. 249, pp. 110404, 2024. https://doi.org/10.1016/j.comnet.2024.110404

[10] Hadadi, F., Dawes, J. H., Shin, D., Bianculli, D., & Briand, L. "Systematic evaluation of deep learning models for log-based failure prediction," Empirical Software Engineering, vol.29, no. 5, 2024. https://doi.org/10.1007/s10664-024-10501-4

[11] Zang, X. D., Wang, T. L., Zhang, X. C., Jian Gong, Gao, P., and Guowei Zhang, G. W. "Encrypted malicious traffic detection based on natural language processing and deep learning," Computer Networks, vol. 250, pp. 110598, 2024. https://doi.org/10.1016/j.comnet.2024.110598

[12] Si, Z., Liu, Z. Q., Mu, C. C., Wang, M., Fong, T. X., Xia, X. F., Hu, Q., Xiao, Y. "A new deep learning based electricity theft detection framework for smart grids in cloud computing," Computer Standards & Interfaces, vol. 94, pp. 104007, 2025. https://doi.org/10.1016/j.csi.2025.104007

[13] Nagarajan, S., Kayalvizhi, S., Subhashini, R., & Anitha, V. "Hybrid honey badger-world cup algorithm-based deep learning for malicious intrusion detection in industrial control systems," Computers & Industrial Engineering, vol. 180, pp. 109166, 2023. https://doi.org/10.1016/j.cie.2023.109166

[14] Wang, Z. H., Thing, V. L. L. "Feature mining for encrypted malicious traffic detection with deep learning and other machine learning algorithms," Computers & Security, vol. 128, pp. 103143, 2023. https://doi.org/10.1016/j.cose.2023.103143

[15] Hashmi, A., Barukab, O. M., & A.H. Osman, O. M. "A hybrid feature weighted attention based deep learning approach for an intrusion detection system using the random forest algorithm," Plos One, vol. 19, no. 5, 2024. https://doi.org/10.1371/journal.pone.0302294

[16] Hnamte, V., Nhung-Nguyen, H., Hussain, J., & Hwa-Kim, Y. "A novel two-stage deep learning model for network intrusion detection: LSTM-AE," IEEE Access, vol. 11, pp. 37131-37148, 2023. https://doi.org/10.1109/ACCESS.2023.3266979

[17] Wasif, M. S., Miah, M. P., Hossain, M. S., Alenazi, M. J., & Atiquzzaman, M. "CNN-ViT synergy: An efficient Android malware detection approach through deep learning," Computers and Electrical Engineering, vol. 123, pp. 110039, 2025. https://doi.org/10.1016/j.compeleceng.2024.110039

[18] Hilal, M., Hashim, A. A., Mohamed, H. G., Nour, M. K., Asiri, M. M., Al-Sharafi, A. M., Othman, M. & Motwakel, A. "Malicious URL Classification Using Artificial Fish Swarm Optimization and Deep Learning," Computers, Materials and Continua, vol. 74, no. 1, pp. 607-621, 2022. https://doi.org/10.32604/cmc.2023.031371

[19] Babayigit, B., & Abubaker, M. "Towards a generalized hybrid deep learning model with optimized hyperparameters for malicious traffic detection in the Industrial Internet of Things," Engineering Applications of Artificial Intelligence, vol. 128, pp. 107515, 2024. https://doi.org/10.1016/j.engappai.2023.107515

[20] Fu, X. B., Lou, S. P., Zheng, J. M., Chi, C., Yang, J., Wang, D., Zhu, C. M., Huang, B. T., & Zhu, X. T. "Deep learning techniques for DDoS attack detection: Concepts, analyses, challenges, and future directions," Expert Systems with Applications, vol. 291, pp. 128469, 2025. https://doi.org/10.1016/j.eswa.2025.128469

[21] Zuo M., Guo, C. Y., Xu H. Y., Zhaoxin Zhang, Z. X., and Cheng, Y. N. "METC: A Hybrid Deep Learning Framework for Cross-Network Encrypted DNS over HTTPS Traffic Detection and Tunnel Identification," Information Fusion, vol. 121, pp. 103125, 2025. https://doi.org/10.1016/j.inffus.2025.103125

[22] Bhushan, K., & Gupta, B. B. "Network flow analysis for detection and mitigation of Fraudulent Resource Consumption(FRC)attacks in multimedia cloud computing," Multimedia Tools and Applications, vol. 78, no. 4, pp. 4267-4298, 2019. https://doi.org/10.1007/s11042-017-5522-z

[23] Chauhan, V. K. & Kumar, A. "Cascaded capsule twin attentional dilated convolutional network for malicious URL detection," Expert Systems with Applications, vol. 262, pp. 125507, 2025. https://doi.org/10.1016/j.eswa.2024.125507

[24] Ghahramani, M., Taheri, R., Shojafar, M., Javidan, R., & Shaohua Wan, S. H. "Deep Image: A precious image based deep learning method for online malware detection in IoT environment," Internet of Things, vol. 27, pp. 101300, 2024. https://doi.org/10.48550/arXiv.2204.01690

[25] Iqbal, T., Wu, G. W., Iqbal, Z., Mahmood, M. B., Shafique, A., and Guo, W. H., "PypiGuard: A novel meta-learning approach for enhanced malicious package detection in PyPI through static-dynamic feature fusion," Journal of Information Security and Applications, vol. 90, pp. 104032, 2025. https://doi.org/10.1016/j.jisa.2025.104032

[26] Jeon, S. E., Oh, Y. S., Lee, Y. J., & Lee, I. G. "Suboptimal feature selection techniques for effective malicious traffic detection on lightweight devices," CMES - Computer Modeling in Engineering and Sciences, vol. 140, no. 2, pp. 1669-1687, 2024.

[27] Kalaria, R., Kayes, A. S. M., Rahayu, W., Pardede, E., & Salehi, A. "IoTPredictor: A security framework for predicting IoT device behaviours and

detecting malicious devices against cyber attacks," Computers & Security, vol. 146, pp. 104037, 2024. https://doi.org/10.1016/j.cose.2024.104037

[28] Jia, H. T., Lang, B., Li, X. Y., and Yan, Y. H. "IDEAL: A malicious traffic detection framework with explanation-guided learning," Knowledge-Based Systems, vol. 317, pp. 113419, 2025. https://doi.org/10.1016/j.knosys.2025.113419

[29] Khashan, O. A. "Dual-stage machine learning approach for advanced malicious node detection in WSNs," Ad Hoc Networks, vol. 166, pp. 103672, 2025. https://doi.org/10.1016/j.adhoc.2024.103672

[30] Kolasa, D., Pilch, K., & Mazurczyk, W. "Federated learning secure model: A framework for malicious clients detection," SoftwareX, vol. 27, pp. 101765, 2024. https://doi.org/10.1016/j.softx.2024.101765

[31] Wang, Y., Xiao, R., Sun, J., & Jin, S. "MC-Det: Multi-channel representation fusion for malicious domain name detection," Computer Networks, vol. 255, pp. 110847, 2024. https://doi.org/10.1016/j.comnet.2024.110847

[32] Yevsikov, A., Muralidharan, T., Panker, T., Nissim, N. "CADefender: Detection of unknown malicious AutoLISP computer-aided design files using designated feature extraction and machine learning methods," Engineering Applications of Artificial Intelligence, vol. 138, pp. 109414, 2024. https://doi.org/10.1016/j.engappai.2024.109414

[33] Rafi, S. M., Yogesh, R., & Sriram, M. "Optimized dual access control for cloud-based data storage and distribution using global-context residual recurrent neural network," Computers & Security, vol., pp. 104183, 2024. https://doi.org/10.1016/j.cose.2024.104183

[34] Long, G. L., Yu, K., Yang, S. F., Xiaohong Zhou, Shen, X. D., & Lu, N. F. "Software anomaly detection technology based on deep learning," Procedia Computer Science, vol. 259, pp. 1123-1129, 2025. https://doi.org/10.1016/j.procs.2025.04.066