Hindawi
Security and Communication Networks
Volume 2022, Article ID 5692820, 18 pages
https://doi.org/10.1155/2022/5692820

Research Article

A DDoS-Attack Detection Method Oriented to the Blockchain Network Layer

Qian-yi Dai,[1,2] Bin Zhang,[1,2] and Shu-qin Dong[1,2]

[1]*Zhengzhou Information Science and Technology Institute, Zhengzhou, Henan Province 450001, China*
[2]*Henan Key Laboratory of Information Security, Zhengzhou, Henan Province 450001, China*

Correspondence should be addressed to Qian-yi Dai; qianyi.dai@outlook.com

Received 24 January 2022; Revised 1 March 2022; Accepted 26 March 2022; Published 4 May 2022

Academic Editor: AnMin Fu

By nature, a traditional attack method, denial-of-service (DDoS) attack poses a considerable threat to the security of the blockchain network layer. This paper proposes a distributed DDoS-attack traffic detection method based on a cross multilayer convolutional neural network model in the blockchain network layer. The method resolves the low generalisation, high misreporting rate, and low detection efficiency problems of the existing detection methods, which are caused by nondistinctive core features and the high complexity of robust features when detecting DDoS attacks transmitted by mixed protocols on a blockchain network layer. First, the model performs a convolution operation on preprocessed traffic on the blockchain network layer using a cross-layer method based on L_2 regularisation. After this operation, the model can perceive the detailed features of attack traffic from multiple levels while enhancing the representational performance of key features; specifically, the parameters with high-variance terms are penalised to limit changes in the model's weight parameters. The highly robust abstract features of attack traffic are extracted, thereby increasing the generalisation ability and reducing the misreporting rate of the model. Second, parametric encoding of the abstract features is performed by a stacked sparse autoencoder based on Kullback–Leibler divergence, and the sparsity of the model is adjusted to reduce the redundant data and the coupling between abstract features. The outputs of the encoded features are then effectively categorised. Finally, the global optimisation of parameters is performed by an improved random gradient-descent algorithm, which prevents oscillation of the training parameters and accelerates the model convergence. In an experimental evaluation, the proposed method achieved satisfactory binary- and multiclass detection of DDoS-attack traffic on both CSE-CIC-IDS 2018 on the AWS dataset and on the real mixed data of a blockchain network layer.

1. Introduction

The network layer of blockchain is the core supporting technology structure that can ensure legal addition and effective communication of blockchain nodes and includes multiple network communication technologies, e.g., networking mode, communication mode among nodes, extended network, and anonymous network. The communication process is independent of the trusted third party because the network layer of blockchain now mainly uses P2P (peer-to-peer) to build connection of network nodes. The blockchain system must follow P2P connection for nodes and the mechanism of freely publishing block content so that the network layer of blockchain uses the distributed and self-organized connection mode and lacks identity authentication, data verification, and network security management mechanism. The attacker may freely publish illegal content, spread the Trojan and virus, and implement DDoS (Distributed denial of service) and routing attack. Therefore, it spreads fast and cannot be detected easily. With their nonstructured network systems, high complexity of mixed data traffic, difficulty of control, and nonuniform network protocols, blockchain-based networks are major targets of distributed denial-of-service (DDoS) attacks [1]. In recent years, proliferating DDoS attacks have increasingly threatened blockchain security. Machine and mining shutdown incidents resulting from DDoS attacks have affected multiple exchanges and mines, such as Mt.

Gox, Fomo 3D, LooksRare, and Youbite. Clearly, conventional cyberattack methods continue to pose a huge security threat to the network layers of blockchains [2, 3]. Leveraging the high corruptibility, unevenly distributed computational power, and unsophisticated authorisation mechanisms of blockchain-consensus protocols [4, 5], attackers may launch redundancy- and rumour-based DDoS traffic attacks against the core computing nodes within the blockchain network layer. These attacks exhaust the resources of the attacked miner nodes and damage the computational-power balance of the blockchain system, granting attackers' computational power advantages and the preconditions for launching consensus-based attacks such as hard fork or 51% attacks [6, 7].

There are two modes of DDoS attack on the network layer of blockchain: (1) DDoS attack on the trading platform: For the network bandwidth, there is a high demand for the trading platform. Once DDoS attack occurs, the trading platform cannot work normally during downtime of the trading system due to resource depletion of the network host, which will directly influence trading volume and uprise of the blockchain currency and result in loss of platform capital and users. (2) DDoS attack on the mining pool: The attacker can directly start DDoS attack on the host running mining operation in each mining pool, stop trading verification and recording, and destroy activation of the consensus mechanism. After starting specific DDoS attack, the attacker can increase network load of the blockchain and occupy hard disk space of the mining pool through sending invalid block data and redundant flow, resulting in network delay and delayed trading verification. As a result, the attacker can further start DDoS attack on other normal trading on the network, which will influence consensus efficiency and throughput of the system. Particularly, 51% attacks can be implemented based on DDoS attack among attackers, resulting in on-chain delay of the block generated by the passive attacker. As a result, the trading will be rejected, which will waste the hash rate and influence on-chain result. To prevent attackers from unbalancing and destabilising the computational power in blockchain systems, methods that detect DDoS traffic attacks against the blockchain network layer are essential [8, 9].

Because the dataset distribution and traffic features differ between normal and DDoS traffic on the blockchain network layer, most studies collect and process the traffic flow through the blockchain network layer and transform it into datasets for separate detection and research [10]. Presently, DDoS attacks against blockchain network layers are detected by two types of methods: statistical and machine-learning methods. Statistical detection methods based on node-side traffic classify the traffic datasets in the leaf routers of a blockchain network within appropriate time intervals [11]. The classification is based on the threshold value of a statistical indicator such as the network congestion degree [12], which separates DDoS-attack traffic flows from normal traffic through the blockchain network layer [13]. However, these methods have several drawbacks. First, attackers can exploit the redundant data transmitted by the blockchain-consensus mechanism. For instance, DDoS-attack traffic can be camouflaged among redundant P2P traffic via a flooding mechanism using protocol obfuscation or variation [14]. As the core statistical features of this attack-traffic type are similar to those of normal traffic, statistical detection methods often have a low accuracy rate [15]. Second, the blockchain-consensus mechanism indiscriminately processes the traffic-load information sent from all miner nodes; therefore, attackers may encapsulate and transmit rumour-based DDoS-attack traffic containing camouflaged load information. As the data classification and traffic-load features of this attack-traffic type are very similar to those of Flash Crowd traffic, the attacks are commonly misrecognised by statistical detection methods, leading to a high misreporting rate [16].

Machine-learning detection methods capture the traffic data at endpoint nodes. A machine-learning algorithm then mines and extracts the core features of DDoS-attack traffic on the blockchain network layer. After training, the machine-learning algorithm can classify the traffic of the blockchain network layer to be detected and thereby check whether the input contains DDoS-attack traffic [17]. These methods also have several shortcomings. First is the coexistence of multiple consensus mechanisms (e.g., proof of work (PoW), proof of stake (PoS), delegated PoS (DPoS), and Byzantine fault tolerance) in the environment of the blockchain network layer [18, 19]. Such multiplicity confounds the underlying traffic protocols (e.g., RLPx, GHOST, Gossip, DEVp2p, Stratum, and Gnutella) and causes variable encapsulation formats of the underlying traffic across different running platforms (e.g., Ethereum, Hyperledger Fabric, and IOTA). These ramifications increase the dimensions of the nonlinear features of the traffic data and the complexity of the classified outputs of DDoS detection models, thereby weakening their generalisation capability for DDoS traffic in the blockchain network layer [20]. Second, as rumour-based attack traffic has very similar features to normal network traffic on the blockchain network layer, the core features of mixed DDoS-attack traffic are not easily distinguished by machine-learning methods, leading to a low accuracy rate of detection by these methods [17].

In terms of flow structure, the redundant P2P flow and rumour DDoS attack in the blockchain network are legal. Compared to the traditional DDoS attack, the above attack has a larger invisible flow. Compared to the traditional DDoS flow, the above flow has insignificant core features and a larger difficulty in detection. In summary, although the underlying blockchain network layer is built upon the P2P network structure, DDoS-attack traffic through this layer cannot be adequately detected by the existing P2P DDoS detection methods. All traffic information is indiscriminately and reciprocally processed by the blockchain-consensus mechanisms, and redundant network traffic is necessarily transmitted across the blockchain. Therefore, attackers may falsify illegitimate traffic as the traffic of the P2P network layer or rumour-based traffic for transmission. The existing suppression strategies based on malicious redundant P2P traffic cannot be directly applied to blockchain networks [21]. This type of DDoS attack can evade detection and screening by traditional P2P networks and downgrade the security of traffic on the blockchain network layer [22, 23].

As mentioned above, machine-learning algorithms have limited perceptibility and, therefore, low detection efficiency for traffic in the complex environments of blockchain network layers, whereas statistical methods have a low accuracy rate and high misreporting rate. To resolve these problems, the present study proposes a comprehensive multilayer convolutional neural network (CMCNN) deep-learning model based on machine-learning detection methods and designs some relevant detection methods. The paper detects DDoS-attack flow in the mining pool, redundant P2P flow, and rumour DDoS-attack flow to improve security of the blockchain network layer.

The method perceives the distinctive core features of DDoS-attack traffic at the blockchain network layer and supports the efficient detection and separation of DDoS-attack traffic. The contributions of this research are described below,

(1) Machine-learning methods cannot easily extract the distinctive core features when trained on mixed DDoS-attack traffic on blockchain network layers. We therefore designed an L_2-based regularised multilayer convolutional method that supports the separation of core feature vectors of the attack traffic from the mixed features. The features of DDoS-attack traffic on the blockchain network layer are extracted and mined by a multilayer convolutional process that probes the multiple channels of traffic datasets and merges the outputs to avoid repetitive output representations of the extracted core features. The L_2-based regularisation method in the multilayer convolutional process penalises the invalid parameters with high-variance terms and inhibits their outputs, incentivising the model to extract and separate the highly robust features of DDoS-attack traffic on the blockchain network layer. The regularisation improves the accuracy and recall rates of the model in classification-based detection.

(2) During training, the model can be overfitted due to an excessively high and tight coupling degree of the abstract features output by multilayer convolutions. To avoid this problem, we propose a Kullback–Leibler (KL) divergence stacked sparse autoencoder (SSAE) module for sparse coding of the abstract features. This module supports the adaptive reconstructive representation of the coupling relationship among the features and limits the overfitting of complex redundant features during training. By adaptively adjusting the sparsity between neurons in the SSAE module, the model constructs a simple autoencoder-layer network structure, updates the neurons corresponding to abstract features, and reconstructs the combinatorial relationships among the neurons. These actions reduce the coupling degree of the traffic features and the correlations among features in the model. The SSAE module also supports redundant data screening and reduces the overfitting degree in the model training, thus suppressing the effect of noise and other invalid data on the training results. Therefore, it improves the stability and detection efficiency of the model.

(3) Owing to its highly complex structure, the CMCNN model can develop parametric oscillations and fail to converge during multilayer deep-learning classification-based training. To avoid this problem, we propose a rapid convergence-based momentum descending approach for the training of different momentum terms during backpropagation and parameter adjustment of the model. Furthermore, a method that dynamically updates the stride is designed to avoid the missing of the lowest point in the parameter optimisation process. This method also reduces the model training time and improves the generalisation ability, ensuring an effective global-parameter optimisation and rapid convergence of the model.

2. The CMCNN Model and Detection Method

2.1. Design of the CMCNN Model. In the CMCNN-based blockchain network-layer method, DDoS traffic is detected by a deep-learning classification method. The classification method (binary- or multiclass) depends on the types of outputs of different classes. Therefore, a large number of key parameters such as weight, bias, and number of neurons must be set during model training. When deep-learning parameters are manually set, important feature information is easily lost during training and the workload of parametric adjustment increases. CNNs are globally trainable multistage artificial neural network architectures, enabling the design of training models for specific problems. CNNs can mine and perceive the detailed features of attack traffic and repetitively characterise the key features of attack traffic. To leverage these advantages, the CMCNN model includes a multilayer multistage CNN module that perceives and mines the abstract, highly robust core features of DDoS-attack traffic from the traffic data of raw multitype blockchain-consensus protocols, thus improving the classification-based detection ability of the model.

Noise and other data in the traffic samples of blockchain network layers can generate an overtight hidden relationship among the neuron weights during multilayer multistage CNN training, causing overfitting of the trained model. Meanwhile, multilayer convolutional processing increases the degree of internal association among the abstract features. Such highly coupled complex features increase the cost of model training and reduce the perception of detailed features in classification detection. The developed SSAE module reduces the coupling degree and complexity of the output abstract features by convolution and captures the detailed features of attack traffic amid the classified output of SoftMax. The SSAE module adaptively obtains and rerepresents the output features through multistage encoders. During training, it also introduces *KL* divergence to control the sparsity of the intermediate layer, thereby improving the detection efficiency while reducing the model complexity and preventing overfitting.

2.2. Structure and Training Process of the CMCNN Model. As shown in Figure 1, the CMCNN model includes five convolutional layers (Conv1–Conv5), two pooling layers (MaxPooling1 and MaxPooling2), eight SSAE layers (SSAE1–SSAE8), two full connection layers (FullConnection1 and FullConnection2), and a SoftMax layer. The model is sequentially trained by forward-propagation feature extraction and backpropagation iterative optimisation.

The CMCNN structure combines the advantage of CNN (perceiving the core features of forwarded feature extraction) with that of SSAE (reducing the complexity of tightly coupling the features). First, the CMCNN model provides a coarse-grained description through a singular convolution and outputs the DDoS-attack traffic, supporting attack-feature perception. Next, the perception of convolutional features is refined through two channels, forming a multilayer aggregate CNN deep-learning network architecture that outputs same-dimensional multilayer feature vectors. Starting from the second convolutional computation, the model reveals and describes the hidden dependencies of the traffic features of multilayer channels, saves the computing results of layer-by-layer convolution, and separately implements activation and max-pooling layer operations. Using adaptive sparse coding, the SSAE module reconstructs the core features in the output feature vectors of the pooling layer while preventing overfitting. Furthermore, SSAE enables the merging of multilayer SSAE output values into output vectors and supports the multilevel representation of DDoS-attack traffic on the blockchain network layer. In this way, it represents the details of the attack-traffic features from multiple perspectives. During the model-training stage, the training loss value of the relevant model is computed based on the classified output of SoftMax and is applied to gradient-descent training. The weight, bias, and other model parameters are optimised through multiple iterative training until the model converges. In the model output stage, the SoftMax classifier outputs the binary- or multiclass detection results.

2.2.1. Forward-Propagation Feature Extraction Process. The CMCNN model extracts the forward-propagation features through the minibatch method, which randomly selects a fixed-sized batch of samples from the preprocessed training dataset D_{train} and inputs it as a vector to the model during each round of training.

First, the CMCNN model implements a 1×1 convolution kernel (Conv1) on the input data, which encourages the representation of nonlinear features and appropriately increases the depth of the deep-learning model and the feature dimensions. A two-channel, multilevel fine-grained convolutional operation on the feature maps is then executed through four 3×3 convolutional layers (Conv2 and Conv3, Conv4, and Conv5) in a cross-channel convolutional mode. After this execution, the model perceives the combined spatial information of the feature data and the local nonlinear abstract features and can extract the detailed feature maps of DDoS-attack traffic on the blockchain network layer from multiple perspectives [24]. The vectors of both new reconstructed feature maps are joined into a dual-channel feature vector. This differentially represented multichannel feature vector serves the following purposes: (1) Because it reveals the hidden dependency in each cross-channel from the multiple perspectives of different convolution kernels outputting anisotropic results, it preserves the details and hence improves the detection accuracy; (2) it rerepresents the core features between two groups of cross-layer channels and reinforces the classified presentation of the crucial features amid the merged output. Therefore, the classifier can better perceive the differences in the core features between normal and DDoS-attack traffic on the blockchain network layer. The convolution kernels adopted in the cross-layer convolution process are shown in Figure 2.

Each convolution kernel outputs a feature map x_j^l denoted as

$$x_j^l = \sigma\left(\sum_{i \in M_j} x_j^{l-1} * w_{ij}^l + b_j^l\right), \tag{1}$$

where l is the current layer; M_j is the convolution window corresponding to the jth convolution kernel; x_{ij}^l is the feature map in the set; w_{ij}^l and b_j^l are the weight and bias, respectively, of the convolution kernel; $*$ denotes the convolution operation; and $s(\cdot)$ is the activation function of the convolutional layer. The convolution outputs are passed through a leaky rectified linear (ReLU) unit (or activation function) that appropriately increases the sparsity of the network model parameters. The representative traffic features are then integratively extracted, and the generalisation ability of the trained model is reinforced. The formulation is as follows:

$$\text{LeakyReLU}(x) = \begin{cases} x, & \text{if } x > 0, \\ \mu x, & \text{if } x \leq 0. \end{cases} \tag{2}$$

where m is the leakage coefficient. The leaky ReLu function retains the input values within the negative semiaxis and sets a small gradient that prevents the loss of neuron information during the model training. This function actively limits the influence of negative input values in the model.

The Max_pooling1 and Max_pooling2 layers implement the downsampling computations of the feature maps output by the cross-layer convolutional kernels Conv3 and Conv5, respectively. The dimensionality of the features is reduced while maintaining translation, rotation, and scaling invariance of the feature vector space. The local optimal features are retained, and overfitting of the trained model is prevented by the following expression:

$$y_j^{l+1} = w_j^l \cdot \text{down}\left(y_j^l\right) + b_j^l, \tag{3}$$

where down() is the downsampling function, w_j^l is the weight value of the pooling layer, b_j^l is the bias term, the pooling window is sized 2×2, and the stride size is 2.

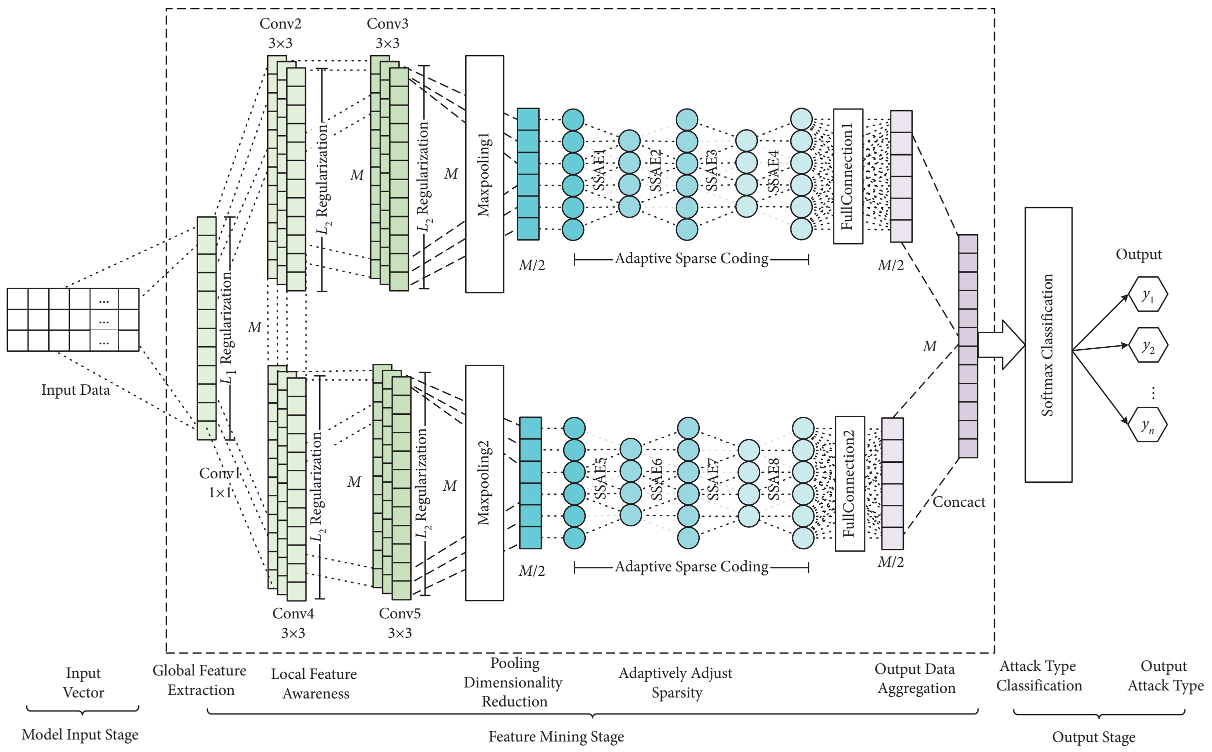
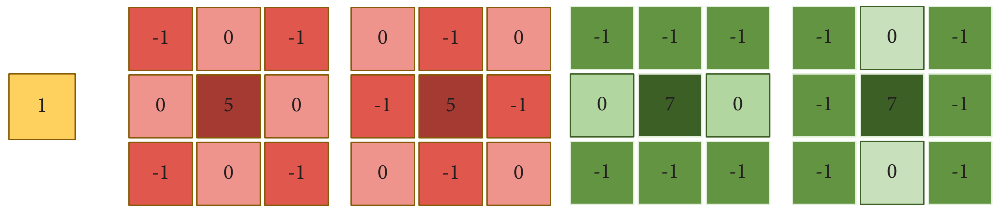


FIGURE 1: Overall structure of the CMCNN model.



(a) Conv1 Convolution Kernel (b) Conv2 Convolution Kernel (c) Conv3 Convolution Kernel (d) Conv4 Convolution Kernel (e) Conv5 Convolution Kernel

FIGURE 2: Convolution kernel used by each convolutional module in the CMCNN model: (a) Conv1 convolution kernel, (b) Conv2 convolution kernel, (c) Conv3 convolution kernel, (d) Conv4 convolution kernel, and (e) Conv5 convolution kernel.

Maxpooling1 and MaxPooling2 output $M/2$-dimensional dataset results, which are input to the autoencoder layers SSAE1 and SSAE2.

The eight sparse autoencoder layers in the SSAE module are stacked in a front-to-rear manner. Specifically, four contractive autoencoders compress the abstract models' data, which are output by the multilayer CNN, to improve the model's resistance to disturbance by high-dimensional input features and to improve the performances of the core weight and bias parameters. By extracting the features from high to low dimensions, these encoders can prevent models from overfitting and being trapped in local minima. The four decoders reconstruct the abstract data features and decode the compressed features into sparse output vectors. The CMCNN model can then perceive the sparsity features of the input data and prevent local minimisation of the training results caused by the overcompression of abstract features. The autoencoder composed of stacked cross-layer encoders and decoders can denoise the redundant input data and reconstruct the useful data. The integratively described abstract data features contain the local information with differentiated output classes, thus fulfilling adaptive extraction of the core abstract features [25].

The number of neurons in the intermediate layer of SSAE significantly influences the output results. For example, if the number of intermediate-layer neurons is excessively small, the model cannot effectively represent the reconstruction features; in contrast, a very large number of neurons can cause redundant model parameters that reduce the compression efficiency. The number of intermediate-layer neurons can be adaptively adjusted using the sparse regularisation method, which changes the output results of part of the units to zero. This action reduces the complex coadaptive coupling relationship between pairs of neurons, increasing the model's ability to learn the robust features while relieving the overfitting during training. The input feature data can then be reproduced with a reduced number of neurons. z^l, w_j, and b_m are the input value, weight, and bias of the autoencoder layer, respectively; $g_e(\cdot)$ is the tanh activation function; ρ' m denotes the average activation level of the intermediate-layer units; and N' is the number of neurons in the SSAE module;

$$z_j^{l+1} = g_e\left(w_j \cdot z_j^l + b_m\right),$$
$$\rho_m' = \frac{1}{N'}\sum_{n=1}^{N'} g_e\left(w_j z_n + b_m\right), g_e(x) = \tanh(x) = \left(\frac{e^x - e^{-x}}{e^x + e^{-x}}\right). \quad (4)$$

To reduce the effect of the feature location on the classified output results, the full connection operation is executed on the feature maps output by SSAE4, and SSAE8 and the feature spaces are mapped to the labelled sample spaces by a linear transformation. The derived $M/2$-dimensional feature maps are described by $z_o = (z_o^1, z_o^2, \ldots, z_o^{M/2})$ and $z_{o'} = (z_{o'}^1, z_{o'}^2, \ldots, z_{o'}^{M/2})$, respectively, where z_n denotes the nth output feature. The z_o and $z_{o'}$ feature maps are then concatenated into a comprehensive M-dimensional vector map z', which is input to the SoftMax function,

$$z' = (z_o, z_{o'}) = \left(z_o^1, z_o^2, \ldots, z_o^{M/2}, z_{o'}^1, z_{o'}^2, \ldots, z_{o'}^{M/2}\right). \quad (5)$$

The SoftMax classifier executes a normalisation operation on z', outputting k types of traffic classification results. The settings assigned to k are variable and depend on the types of detection outputs. When $k = 2$, the model is a binary classification model that determines whether the traffic is DDoS-attack traffic or normal traffic. Otherwise, k can denote multiple classes corresponding to the settings of the types of sample labels. The model then detects a specific type of DDoS-attack traffic on the blockchain network layer. The SoftMax classifier maps the outputs of multiple neurons to the range [0,1) based on probabilistic forms, and its outputs must sum to 1.0. The function is defined as

$$S(z') = \frac{e^{z_i}}{\sum_k e^{z_i}}. \quad (6)$$

2.2.2. Backpropagation Iterative Optimization. In the backpropagation iterative optimisation process, the output loss includes the error between the model output value and the target value and the penalty value of the model parameter regularisation. The backpropagation algorithm allocates this loss value to the neurons of each layer and obtains the errors in each layer. It then corrects the parameter values of each neuron. The main functions of the CMCNN backpropagation process are as follows: computing the output loss of the forwarding propagation, computing the backpropagation error based on the improved gradient-descent method, obtaining the rates of change in weight and bias, and updating the loss function.

The overall loss function is defined, and all parametric changes in the model are penalised to achieve model convergence. The error loss of the model, denoted by J_s, is calculated from the sample classification results of the training set on the SoftMax layer, which are expressed by

$$J_S = \frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{K} H\left(x_{mk}, z_{mk}\right). \quad (7)$$

Here, $H(x, z)$ is the cross-entropy loss function and x and z denote the expected and observed values of the feature attributes of the input data, respectively. M is the number of samples in the batch training, N is the number of neurons in the convolution layer, and K is the dimension of input features in the training dataset.

The L_2 regularisation term J_{weights} in the module penalises the weight and bias parameters of each convolution layer in the training process, thus inhibiting parametric changes in the high-variance terms. The absolute values of the model parameters tend to decrease, and the probability of selecting the stable parametric terms increases, thus avoiding the effect of noise and other data on the model and preserving the highly robust core features. The formulation is

$$J_{\text{weights}} = \frac{1}{2M}\sum_{k=1}^{K}\sum_{n=1}^{N}\left(w_{mn}\right)^2. \quad (8)$$

The sparse coding operation in the SSAE performs the data compression and reconstructive coding of the high-dimensional, complex features output by the cross-layer convolutional module. To reduce the number of redundant parameters and training complexity and to mitigate the overfitting problem in model training, the sparse coding operation introduces a sparse regularisation term that adjusts the number of neurons in the hidden layer. The loss value of the SSAE layer is described by the KL convergence. The sparse regularisation term, denoted J_{sparsity}, is expressed as

$$\begin{aligned} J_{\text{sparsity}} &= \sum_{m=1}^{M} \mathrm{KL}\left(\rho \| \rho_m'\right) \\ &= \sum_{m=1}^{M}\left[\rho \log\left(\frac{\rho}{\rho_m'}\right) + (1-\rho)\log\left(\frac{1-\rho}{1-\rho_m'}\right)\right]. \end{aligned} \quad (9)$$

In summary, the global loss function E_s of the CMCNN model consists of three components: the error loss value of the model, the L_2 weight regularisation loss of the cross-layer convolution layer, and the sparse coding loss of the SSAE module,

$$E_S = J_S + \lambda \cdot J_{\text{weights}} + \beta \cdot J_{\text{sparsity}}. \quad (10)$$

Here, Parameter J is the regularisation coefficient and β is the coefficient of the sparse control weight. As l increases, the regularisation effect improves, the standard deviation of the parametric distribution of the model reduces, and the model becomes more skewed towards the underfitting region during training. Increasing b, which denotes the closeness between the neuron weights, increases the independence of the neurons.

To optimise the values of all trained parameters, the weights and biases of each model layer are adjusted layer-by-layer according to the gradient-descent changes in the overall loss value, which are dictated by the backpropagation rule. The adjustment continues until the model has properly converged. To strengthen the global optimisation ability of the model parameters and accelerate the model convergence, the CMCNN model performs the following functions:

① Parametric initialisation in the cross-layer convolution layer and the SSAE layer is performed by the He method [26] and the Xavier method [27], respectively. This step prevents parametric loss, training failure, and model oscillations caused by improper initial parameter settings in the convolution and SSAE layers of the CMCNN cross-layer model.

② The learning rate η is dynamically changed by adjusting the learning stride. A relatively large learning rate is given to the training model in the early iterative stage and is gradually reduced as the model parameters stabilise. The dynamic learning rate enables parameters with large partial derivatives in the loss function to obtain a relatively large gradient-descent learning rate, which facilitates convergence; on the contrary, parameters with relatively small partial derivatives in the loss function descend more slowly in their learning rates to facilitate penalty. H is expressed as

$$\eta = \frac{\eta_{\max} - \eta_{\min}}{e^{\left(\xi \cdot n_{\text{cur\_iter}}/N_{\text{iter}}\right)}} + \eta_{\min}, \tag{11}$$

where $\eta_{\max}$ and $\eta_{\min}$ denote the maximum and minimum learning rates, respectively; ξ is the learning-rate adjustment parameter; $n_{\text{cur\_iter}}$ is the number of times of the current iteration; and N_{iter} is the total number of iteration.

③ Based on the minibatch-training process, the CMCNN model updates the computational method for adjusting the model parameters. For this purpose, it introduces an ISGD (improved stochastic gradient-descent) parameter update algorithm. The momentum parameter α maintains the stability of the gradient-descent method and accelerates the learning while ensuring the lowest points during the model training. It is expressed as

$$\begin{aligned} \Delta\theta_i &= \alpha\theta_{i-1} - \eta\nabla_{\theta_i} E_s(\theta), \\ \theta_{i+1} &= \theta_i + \Delta\theta_i, \end{aligned} \tag{12}$$

where θ is a vector of training parameters ($w_0, w_1, \ldots, w_n, b_0, b_1, \ldots, b_n$). Note that θ contains all weight values w and bias values b in the convolution and autoencoder layers of the CMCNN. As the function training targets the global loss function $E_S(\theta)$, gradient-descent training by the ISGD algorithm is performed on θ. The parametric training algorithm of the CMCNN model is described below.

2.2.3. Analysis of Time Complexity of the Model. The time complexity of the ISGD algorithm that performs the gradient-descent training in the CMCNN model is given by

$$O\left(S_{\text{data}} \cdot C \cdot \log\left(\frac{1}{\Delta\text{Accuracy}}\right)\right), \tag{13}$$

where S_{data} is the size of the training data, C is the time complexity of each round of iterative training, and ΔAccuracy is the accuracy of the target model to be trained. When computing the time complexity of a single iteration of the model update, we must consider the effects of the data dimension K_{dim} and parameter dimension N_{dim} on C. The time complexity of a single iterative training is

$$C = O\left(K_{\text{dim}} \cdot S_{\text{data}} \cdot N_{\text{dim}}\right). \tag{14}$$

The values of K_{dim} and N_{dim} are the dimensions of the input vector K in the CMCNN model and the total number of neurons in the relevant model, respectively. As K_{dim}, N_{dim} and ΔAccuracy are constants, the complexity of the model's overall training is 0 ($S2$ data), that is, the overall training complexity of the model is determined by the number of samples, consistent with the model complexities in relevant studies [13, 18, 28].

2.3. Process of DDoS-Attack Traffic Detection in Blockchain Network Based on the CMCNN Model. The CMCNN model detects traffic flow in four main stages: (1) Preprocessing stage: Real-time or historical traffic data from the blockchain network layer are collected and preprocessed to produce a training set and a test set. (2) Model training stage: The training dataset is input to the CMCNN model, and the training process begins with parameter initialisation. In the forward-propagation process, the model separately trains the parameters in each layer of the cross-layer convolution and SSAE module and outputs the corresponding feature maps x^l and z^l in a layer-by-layer manner, eventually generating a merged output vector z' of separate classes. In the backpropagation updating process, the model adjusts the weight, bias, and number of neurons of each layer in the cross-layer convolution and SSAE module, generating the loss values $_s$, J_{weights}, and J_{sparsity} and adjusting the weight and bias parameters through ISGD-based gradient-descent training. (3) Model validation stage: The performance of the classified outputs of the converged model is evaluated on the test set. If the model is successful, ΔAccuracy will conform to the designed requirements; otherwise, the model is redirected to Step 3 and the iterative training repeats until the target requirements are met. (4) Actual validation test: When the target requirements are met, the optimised model is applied to real-time detection of DDoS-attack traffic on the blockchain network layer, generating classified outputs of normal and DDoS-attack traffic. The classification is followed by evaluation and validation of the model performance. The detection process of the CMCNN model is shown in Figure 3.

```
Input: Training set $D_{\text{train}}$, CMCNN model $f$, the number of neural network layers $L$, bias $b$, learning rate $h$, momentum parameter a, and batch training size batch_size. Here, $Es(\theta)$ is the loss function of the regularisation term with $w$ and $b$; $\delta(N)$ is the overall loss value in each training round of the CMCNN model. $Z(l)^{(l)}$ is the input in the $l$th layer; $w_f$ and $b_f$ are the weight matrix and bias of the previous round of iteration, respectively; $\varepsilon$ is the threshold value of changes in accuracy rate; and $\Delta$Accuracy is the accuracy of the trained model.
Output: Parameter $\theta *$ of the CMCNN model
Start
(1)   while $\Delta$Accuracy $> e$ && $\delta(N) < 0.2$
(2)     Dsample = Sampling($D_{\text{train}}$, $M$)//Generate training sample with a batch size of $M$
(3)     $y_t$ = Label ($D_{\text{sample}}$)//Generate dataset label
(4)     $x$ = Data ($D_{\text{sample}}$)//Generate dataset to be trained
(5)     $\theta$CNN←ParaInitCNN $(w_0, w_1, w_2, \ldots, w_m, b_0, b_1, b_2, \ldots, b_m)$    //Initialise parameters of the CNN module
(6)     $\theta$SSAE←ParaInitXavier $(w_0, w_1, w_2, \ldots, w_n, b_0, b_1, b_2, \ldots, b_n)$//Initialise the parameters of the SAE module
(7)     for $l \leftarrow L-1$ to 0 do://Train the CMCNN model layer-by-layer
(8)       $w * l, b * l \leftarrow$ (wl, bl)CMCNN//Update parameters in CNN and SAE modules layer-by-layer
(9)       $\delta(N) \leftarrow ES(\theta|y_t, x')$//Compute overall error of the CMCNN model
(10)      $\delta(l+1) \leftarrow (w(l))T\delta(l)f'(z(l))$//Compute the error of each layer of the model
(11)      $\eta * \leftarrow \eta$//Adaptively adjust ISGD learning rate
(12)      $w^{(l)} = w^{(l)} - (\alpha w_f^{(l)} - \eta^* \nabla_{w^{(l)}} \delta^{(l)})$//Update weight parameter according to the ISGD method
(13)      $b^{(l)} = b^{(l)} - (\alpha b_f^{(l)} - \eta^* \nabla_{b^{(l)}} \delta^{(l)})$//Update bias parameter according to the ISGD method
(14)    end for
(15)      $\Delta$Accuracy = $d(N) - d(N-1)$//Compute changes in loss value
(16)    end while
(17)      $\theta^* \leftarrow (w_0^*, w_1^*, \ldots, w_n^*, b_0^*, b_1^*, \ldots, b_n^*)$//Output parameters of the CMCNN model
End
```

Algorithm 1: Parametric training algorithm for the CMCNN model.

Whether the CMCNN-based detection method performs binary- or multiclass model training and detection depends on the number of types in the *label* column of the preprocessed dataset, which can be separately used to detect the absence or presence of DDoS-attack traffic in the blockchain network layer and to discriminate the specific types of attack traffic. The classification detection output is a confusion matrix. The effectiveness of model training and plausibility of the test result are evaluated by tenfold cross-validation in the model training and performance test, respectively.

3. Experimental Validation

The model performance was validated on a mixed dataset of environmental traffic data, including the CSE-CIC-IDS2018 on the AWS dataset and real-world blockchain network data. The experiment aimed to validate and compare the effectiveness and performance of the CMCNN-based detection method under different DDoS traffic conditions on the blockchain network layer.

3.1. Experimental Configuration and Evaluation Indicators. The training, validation, and comparative tests of the CMCNN were performed in the following environment: a Windows 10 operating system, an Intel Core i7-9750 CPU, and 32.0 GB of RAM. The CMCNN model was programmed in the Anaconda Python deep-learning library PyTorch 1.4 in a JetBrain PyCharm 2020.3 environment. To improve the computational efficiency and reduce the training time, the computations were parallelised on an NVIDIA GeForce GTX 2060 GPU. The performances were assessed by standard model evaluation indicators, namely, the accuracy (ACC), recall rate (Recall), precision, false alarm rate (FAR), and F_1-score,

$$\begin{aligned}
\text{Accuracy} &= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}},\\
\text{Recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}},\\
\text{Precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}},\\
\text{FAR} &= \frac{\text{FP}}{\text{TN} + \text{FP}},\\
F_1 - \text{score} &= 2 \times \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}.
\end{aligned} \tag{15}$$

These measures depend on TP (true positives, denoting the number of correctly classified attack samples), FP (false positives or the number of incorrectly classified attack samples), TN (true negatives or the number of correctly classified normal samples), and FN (false negatives or the number of incorrectly classified normal samples). The detection performance was analysed by the receiver operating characteristic curve (ROC), which plots the recall as a function of FAR. The area under the ROC curve (AUC) describes the test capacity of the ROC curve. The larger the area enclosed by the ROC, the higher is the classifier performance. The confusion matrix intuitively presents the final detection results, from which we can describe the

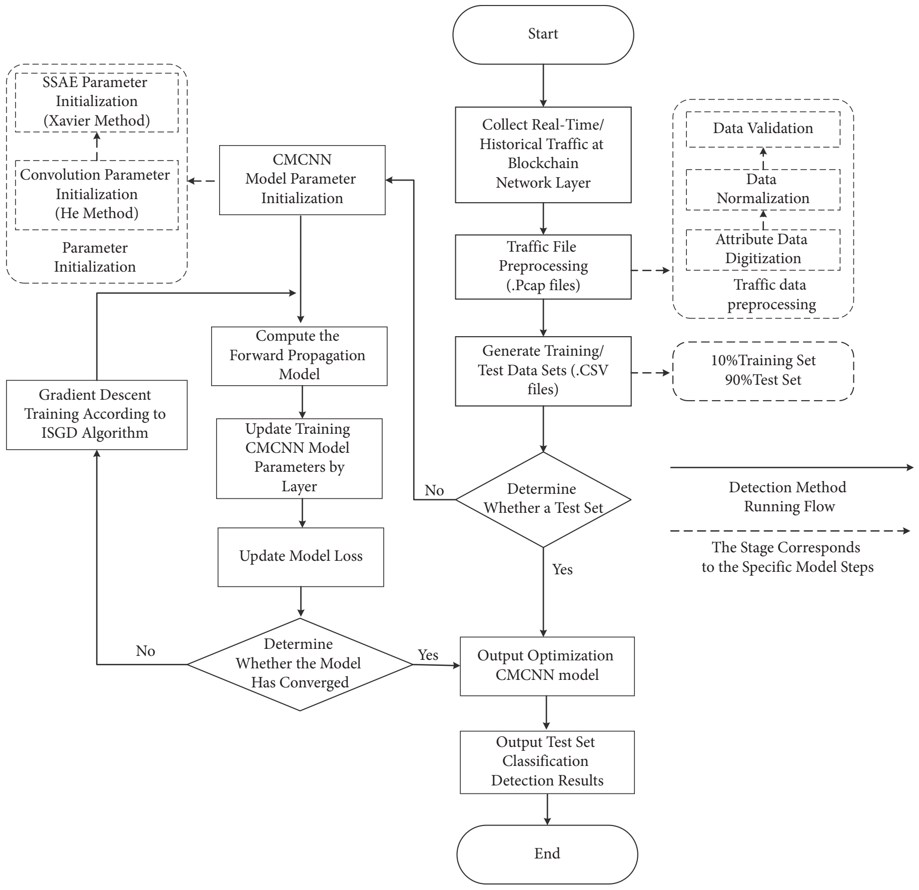

Start
Collect Real-Time/ Historical Traffic at Blockchain Network Layer
Traffic File Preprocessing (.Pcap files)
Generate Training/ Test Data Sets (.CSV files)
Data Validation
Data Normalization
Attribute Data Digitization
Traffic data preprocessing
10%Training Set
90%Test Set
SSAE Parameter Initialization (Xavier Method)
Convolution Parameter Initialization (He Method)
Parameter Initialization
CMCNN Model Parameter Initialization
Compute the Forward Propagation Model
Update Training CMCNN Model Parameters by Layer
Update Model Loss
Gradient Descent Training According to ISGD Algorithm
Determine Whether a Test Set
No
Yes
Determine Whether the Model Has Converged
No
Yes
Output Optimization CMCNN model
Output Test Set Classification Detection Results
End
Detection Method Running Flow
The Stage Corresponds to the Specific Model Steps

Figure 3: Processes of DDoS-attack traffic detection in the blockchain network based on the CMCNN model.

classification results and find the matching degree between the data and real labelled data.

3.2. Preprocessing of Experimental Data. The experimental data were collected using Wireshark, and the resulting traffic-grouping file (.pcap file) was converted by CIC-FlowMeter 4.0 into a dataset file (.csv file) for model training and testing [29]. In the traffic dataset generated by CIC-FlowMeter 4.0, each traffic record included 85 attribute features: 78 numeric attribute features, six character-type attribute features, and one Label attribute field. Subsequently, the dataset was cut for compatibility with the CMCNN model training and then preprocessed through three main operations: data validation, numeralisation of character attribute features, and data normalisation. These operations are briefly described below,

① Data validation: The network traffic data collected from the real network environment contain incomplete default-feature values. To complete the vectors input to the model, the attribute values of all features with the same attribute value in the normal traffic data were deleted and replaced by 0.

② Numeralisation of the character attribute features: Using the attribute mapping method, the character features were converted into binary numeric features and the feature values (represented in the hexadecimal system) were uniformly converted to decimal values. Massive address-type nonnumeric attribute values were converted into numbers of occurrences of their class of attribute values in the entire dataset.

③ Data normalisation: The value ranges of the features in the dataset are widely variable. To eliminate the effect of different value ranges on the model, the value-type data were normalised and all numeric traffic feature values were mapped into the [0,1] range. All data type attributes then fell into the same magnitude level. The computation is given by

$$x_i' = \frac{x_i - \min(x)}{\max(x) - \min(x)}. \tag{16}$$

In Equation (16), x_i' is the normalisation result of the ith feature value, and $\min(x)$和 and $\max(x)$ denote the maximum and minimum values of the attribute feature, respectively. The preprocessing operation converted the raw traffic features and mapped them to a 163-dimensional feature vector. At each time, the sampled traffic data were split into several batch-training samples of fixed size M with K feature dimensions for training the CMCNN model.

3.3. Experimental Environment Setup. A blockchain network layer environment was constructed and combined with the CSE-CIC-IDS 2018 on the AWS dataset. Traditional DDoS-attack approaches and those occurring in the real blockchain network layer were separately simulated. Specifically, the experiment was performed in three traffic scenarios: traffic in the blockchain network layer environment, background-network traffic and DDoS-attack traffic. The blockchain network layer link topology used in the experiment is shown in Figure 4.

This paper selects Ethereum and HyperLeger as the research objects and also as representatives of two typical consensus protocols (i.e., PoS and Fabric); detects and studies the flow of RLPx, Gossip, DEVp2p, Stratum and Gnutella protocol with the proposed investigation method.

The CSE-CIC-IDS 2018 on the AWS dataset is a widely used standard dataset composed of benign and common network traffic attacks. The dataset includes feature-processed traffic, and its format is consistent with that of CICFlowMeter 4.0 outputs. Specifically, the last column of the output is Label, which labels traffic as either benign or a specific attack type. The SE-CIC-IDS 2018 on the AWS dataset contains 13.74 million normal traffic entries and 1.12 million DDoS-attack traffic entries. Particularly, the paper replaces the web page interaction and access to each trading website for legal users in the Internet environment with network background flow to further simulate behaviours of Internet users and blockchain trading website. CICFlowMeter 4.0 is a flow preprocessing tool, which can be offline or online. The tool can monitor and generate features in real time, describe and perceive the distinguishing features of P2P flow, and greatly describe core flow features of DDoS attack on the TCP/IP network layer, transmission layer, and application layer. Therefore, it is used as a preprocessing tool.

Traffic environment of the blockchain network layer: Ethereum 2.0 transaction traffic and Hyperledger Fabric1.4 running traffic were separately designed to simulate mixed traffic on the blockchain network layer, yielding 462,000 traffic entries in total. The Ethereum 2.0 environment consisted of five computers, and transactions between the blockchain mining machines were run by the core Geth programme on Ubuntu 18.04. The IP segment was 192.168.8.0/24, and the computers were connected to Port1 of the Router via Switch1. The Hyperledger Fabric1.4 environment also consisted of five computers, but the blockchain programme was run by the Docker container of Ubuntu 18.04. The IP segment was 192.168.6.0/24, and the computers were connected to Port2 of the Router via Switch2 [30].

Background-network traffic environment: The network behaviour of the traffic in eight local computers visiting blockchain exchanges was collected in this environment. After combining the data with the benign-labelled network traffic in the CSE-CIC-IDS2018 on the AWS dataset, the interactions between legitimate users and miner nodes were simulated. The IP segment was 192.168.122.0/24, and the computers were connected to Port3 of the Router via Switch3. The simulated background-network traffic contained a total of 433,000 traffic entries. There are massive web page interaction flow data in the dataset CSE-CIC-IDS 2018 on AWS, and the author cannot acquire browsing data of the trading website so that we use the dataset CSE-CIC-IDS 2018 on AWS to replace the network flow behaviour of blockchain network nodes and trading website.

DDoS-attack traffic environment: To simulate traditional attach approaches and real-world attack scenarios, the experiment was performed on two datasets. The first dataset included traffic in the CSE-CIC-IDS2018 on the AWS dataset. In this dataset, traditional DDoS-attack solutions were simulated to test the detection capacity of the CMCNN on the traditional DDoS traffic. The second dataset was composed of DDoS-attack traffic through a real blockchain network layer, and the attacks were simulated on three computers named h1, h2, and h3. Computer h1 ran the BoNeSi software on the Red Hat 9.0 operating system to simulate redundancy- and rumour-based DDoS-attack traffic on the blockchain network. Computer h2 ran the HOIC software on a Windows 10 operating system to simulate flood-type DDoS attacks on the data-grouping layer. Computer h3 ran the TFN2K software on a Ubuntu 18.04 operating system to simulate Hypertext Transfer Protocol (HTTP) requests linking separate DDoS attacks on data-grouping and data-flow layers. The IP segment of the computer environment of the DDoS attacks was 192.168.201.0/24, and the computers were connected to Port4 of the Router via Switch4. In total, the simulated attack traffic contained 445,000 traffic entries.

The detection performance of the CMCNN model was tested under various experimental conditions. In Case (1), the detection performance was evaluated on the DDoS-attack traffic in the CSE-CIC-IDS 2018 on the AWS dataset. In Case (2), the binary-class detection performance was evaluated for different numbers of traffic samples in the blockchain network layer environment. In Case (3), the binary-class detection performance was evaluated for different time windows of fixed attack traffic in the blockchain network layer environment. In Case (4), the multiclass detection performance was evaluated for different ratios of attack traffic in the blockchain network layer environment.

3.4. Experiment Result and Analysis

3.4.1. Detection of DDoS-Attack Traffic in the CSE-CIC-IDS 2018 on the AWS Dataset. The CSE-CIC-IDS 2018 on the AWS dataset contains five types of labelled data: Normal, Web Attack, Denial-of-Service (DoS), DDoS, and Infiltration. In this case, the multiclass detection performance of the CMCNN model was assessed on traditional DDoS approaches adopted by attackers in the blockchain network layer environment. The multiclass experimental dataset was composed of the selected dataset and simulated traffic on the blockchain network layer. The attack traffic (accounting for 20% of the experimental dataset) was compiled from 10,000 randomly selected entries of records. To further demonstrate the performance of the CMCNN model on DDoS-attack traffic through the traditional P2P network, P2P network-oriented detection models were selected for parallel comparison.

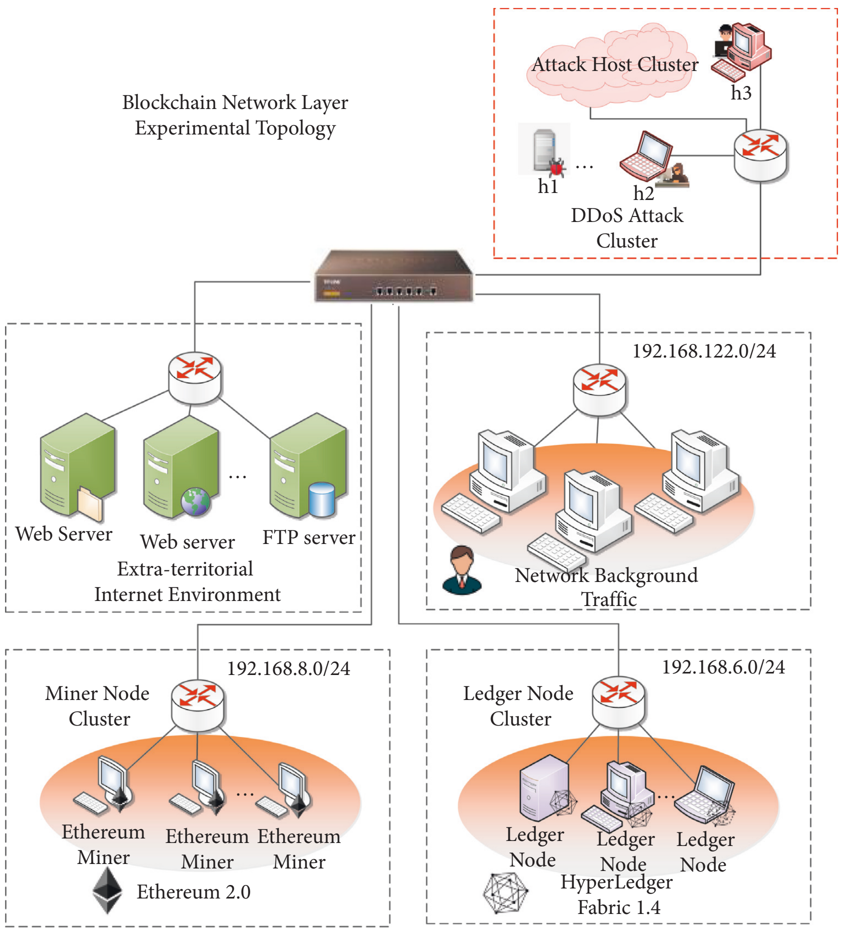


Figure 4: Experiment traffic environment of the blockchain network layer.

The training parameters of the CMCNN model are listed in Table 1. The M, K, and channel parameters were fixed, and their values were determined by the designed structure of the model. The parameters e, ξ, $h_{\min}$, $h_{\max}$, and N_{iter} were preset, and their values were determined by relevant experimental experience. The values of the hyperparameters λ, b, α, ξ, and m were adjusted via grid training until the model converged. Figure 5 plots the training error versus the number of CMCNN interactions. As the number of reverse iterations increased, the loss value gradually decreased and eventually stabilised. After 600 training iterations, the error value was stabilised at below 0.2, and the model converged smoothly with higher performance than the selected literature solutions. This success indicates the suitability of the structural design and hyperparameter settings for the model training. The CMCNN model satisfied the detection requirements.

Figure 6 shows the confusion matrix output by the model after multiclass detection of CSE-CIC-IDS 2018 on the AWS test set. The horizontal and vertical axes correspond to the predicted and actual values, respectively, of the five types of labelled data in the selected dataset. The numbers in the matrix are the numbers of samples having the specified label-type number in the CMCNN model-detection results. Deepening colours of the diagonal matrix elements indicate higher detection accuracy rates for that data type.

From the output of the CSE-CIC-IDS 2018 on the AWS dataset (Figure 6), the detection accuracy and recall rate on the DDoS dataset were determined as 93.85% and 95.59%, respectively, indicating the reasonable performance of the CMCNN model on the selected DDoS dataset. The model showed similarly high detection accuracies on Web Attack, Infiltration, and Normal data but lower detection accuracy on DoS data. The lower accuracy for DoS than for the other four data types was traced to the small size of the DoS sample in the simulation test; consequently, fewer features were extracted, and the detection capacity of the classifier was weakened. This result indicates that when training the CMCNN model, increasing the number of training samples improves the model's perception of core features; on the contrary, insufficient training sample data can lead to limited classification detection performance of the model.

To more intuitively evaluate the multiclass detection performance of CMCNN, the ROC curves of the detection results of the CSE-CIC-IDS 2018 on AWS sample data were plotted (see Figure 7). The AUC of CMCNN was 0.978, larger than those of the other two models (0.964 and 0.948). This result indicates the superior performance of the CMCNN-based detection method on samples of mixed test sets.

Table 1: Parameters selected for the CMCNN model.

Parameter	Description	Value
M	Batch training size	50
K	Dimension of input traffic features	163
Channel	Number of model channels	2
$\rho'\ m$	Average activation degree of the SSAE layer	0.52
λ	L_2 weight regularisation loss	0.49
β	Coefficient of sparse control weight	0.89
α	Momentum parameter	0.97
ξ	Learning rate adjustment parameter	50
ε	Threshold of changes in model accuracy rate	0.05
μ	Leakage parameter of Leaky ReLU	0.1
$\eta_{\min}$	Min value of the learning rate	0.5
$\eta_{\max}$	Max value of the learning rate	0.99
N_{iter}	Total number of model iterations	1200

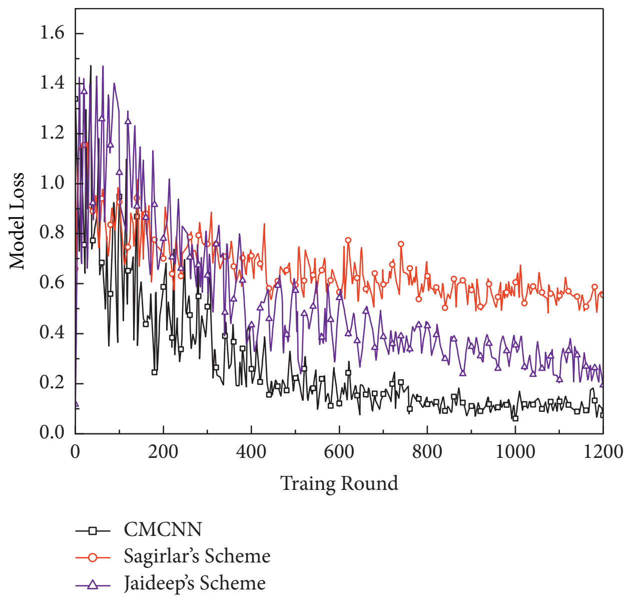


Figure 5: Relationship between CMCNN model training rounds and loss value.

3.4.2. Binary-Class Detection Performance under Different Numbers of Traffic Samples. Next, the binary-class detection performance of the CMCNN model for DDoS attacks was tested on different samples of data flow in the simulation environment of the real blockchain network layer. Normal and DDoS-attack traffic in the real blockchain network layer was collected in the simulation environment. The DDoS traffic includes User Datagram Protocol (UDP)-based DDoS attacks, TCP 3-way handshake-based SYN Flood attacks, Internet Control Message Protocol (ICMP) Flood attacks, and HTTP Flood attacks. The types of DDoS-attack traffic on the blockchain network layer were divided into the following ratios: 60% TCP traffic, 10% UDP traffic, 5% ICMP traffic, 20% redundancy P2P traffic, and 5% rumour-based HTTP request-linking traffic. In the binary classification problem, all types of attack traffic were uniformly labelled DDoS-attack traffic. The abilities of the models to distinguish DDoS-attack traffic among the mixed traffic were tested for different numbers of network traffic samples (Sample_Number = 1000, 5000, 10000, 15000, 20000, 25000, or 30000) collected offline. Table 2 presents the comparative results (Recall, Accuracy, F_1-score, and detection time) of the three detection models for each sample size.

The performances of the three detection models in this experiment are compared in Figure 8. As the number of training samples increased, the Recall, ACC, and F_1-score values of the three methods all exhibited an upward trend. When the number of training samples was constant and the number of detected samples was small, the recall rate and F_1-score of the CMCNN model was relatively low, indicating that the cross-layer CNN module of the model has limited ability to perceive malicious traffic in small-sized samples; however, when the number of samples was relatively large, the model outperformed the other two models, affirming that the SSAE module in the model can better fit large-sized datasets, reconstructively represent the core data features of attack traffic and enhance the detection efficiency. However, the detection times of the three models also linearly increased with the detection sample size. For sample sizes below 2000, the CMCNN model required a longer detection time than the other two solutions, but when the sample size exceeded 20000, its detection time was shorter than those of the other models. This result indicates that the CMCNN model can more effectively compress the core sample features and achieve higher detection performance than the other methods in a large-sized sample environment but offers no distinct advantage in feature compression on small-sized datasets.

3.4.3. Binary-Class Detection Performance with Different Time Windows under Set Attack Traffic. In the simulated real blockchain network layer environment of this experiment, the binary-class detection performance of the model was tested while changing the time window (the ratios of attack traffic remained constant). The network traffic was extracted in real time from the simulated blockchain network layer, thus simulating real DDoS attacks on the blockchain network layer. In the DDoS-attack data, the density of TCP, UDP, ICMP, and redundancy P2P traffic was 1000 times/s, the rate of rumour-based HTTP request-linking attacks was 200 entires/s, and the cycle of the infiltration attack was 10 s. In the simulated background traffic, the flow was normal request linking between users and exchanges, the number of visiting users was 100, the frequency of visits was 10 times/s, and the average flow rate of the blockchain network layer was 1.0 MB/s. The intervals of the collection windows were 10, 20, 30, 60, 120, 180, and 300. The recall rates, misreporting rates, ACC rates, and F_1-scores of real-time detection by the three models were tested under the different window conditions. The results are summarized in Table 3.

Figure 9 presents the recall rates, ACCs, F_1-scores, and FARs of the three models in this experiment. As the time window increased, the recall rates, ACCs, and F_1-scores of the three detection models all increased, but the FAR performance declined. The CMCNN model outperformed the other two models in terms of recall rate and FAR, as seen in the binary outputs. However, on small sample sizes, the CMCNN yielded lower ACCs and F_1-scores than the other

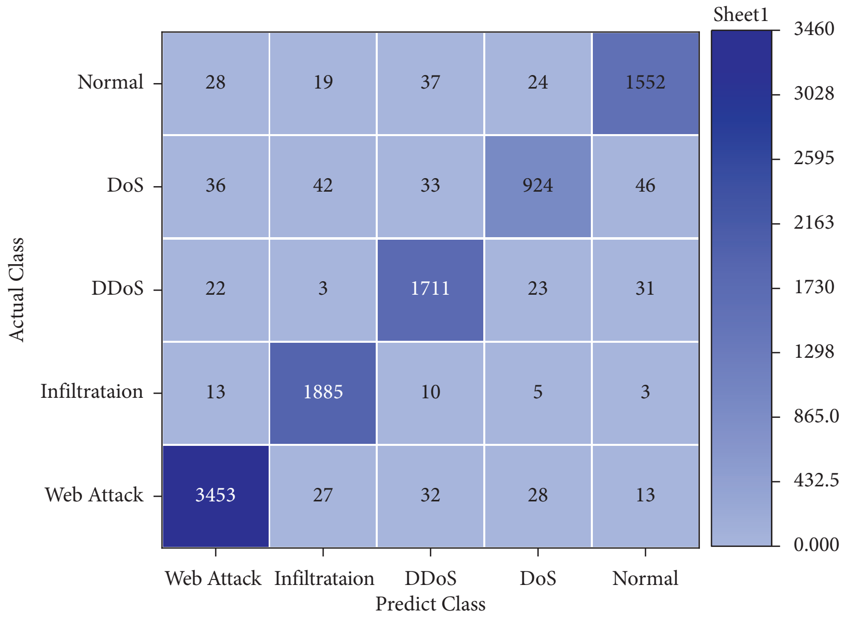


Figure 6: 5-class confusion matrix for the CSE-CIC-IDS 2018 dataset.

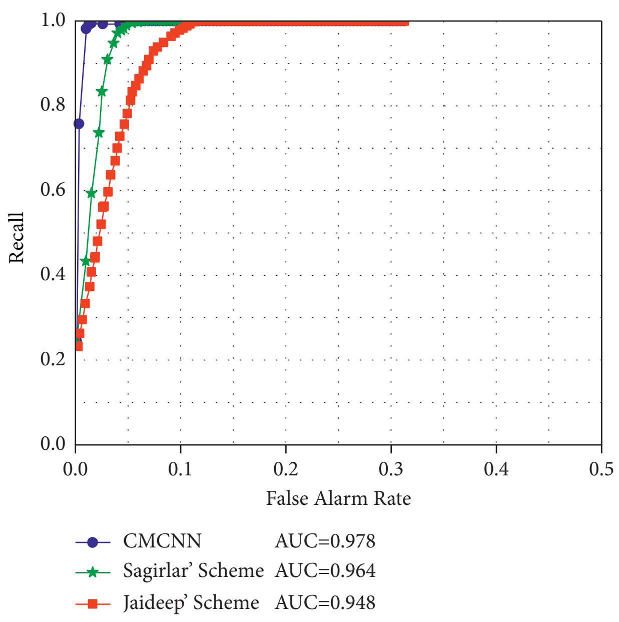


Figure 7: Comparison of ROC curve values of the 3 detection schemes based on the CSE-CIC-IDS 2018 on the AWS test set.

solutions, indicating that its binary classification performance should be improved under small sample-size conditions. Moreover, when the window interval exceeded 180 s, the detection effect of the model tended to stabilise. The appropriate size of the detection window was 200 s on the real blockchain network layer.

3.4.4. Multiclass Detection Performance of Models under Different Ratios of Attack Traffic. In the simulation environment of the real blockchain network layer, the multiclass detection performance of the CMCNN model for DDoS-attack traffic was tested while varying the ratios of attack traffic. The DDoS-attack traffic consisted of TCP flood traffic (50%), redundancy P2P traffic (20%), UDP flood traffic (10%), and ICMP and rumour-based HPPT request-linking traffic. The overall ratio of the DDoS-attack traffic on the blockchain network layer was set to 10%, 20%, 30%, 40%, or 50%, and the collection-window interval was 200 s. The CMCNN performed multiclass detection on mixed DDoS-attack traffic. Table 4 presents the recall, precision, and F_1-score detection results of this experiment.

When the DDoS-attack traffic comprised 50% of the traffic on the blockchain network layer, the recall rates of all five types of DDoS attacks exceeded 95%; when the attack ratio was reduced to 10%, the recall rates remained above 90%. When the ratio of attack traffic on the blockchain network layer was 25%, the recall rate of the CMCNN exceeded 95%, indicating that in most cases, the model could detect the majority of multiclass DDoS-attack traffic on the blockchain network layer.

The multiclass detection performances on rumour-based and redundancy P2P DDoS-attack traffic are compared in Figures 10(a) and 10(b). The CMCNN-based multiclass detection method achieved higher ACCs than the other two methods on both types of attack traffic, indicating that it sufficiently learned the core features of attack samples in multiclass detection. Although the recall rate and F_1-score of the model were weakened when the ratio of attacks was relatively small, CMCNN outperformed the other two models when the ratio of attacks increased. That is, the model has limited learning capacity for the feature details of small-sized samples but can well discriminate the attack traffic features of massive data. Therefore, the CMCNN-based detection method can effectively conduct multiclass detection of massive DDoS-

Table 2: Comparative results of model detection under different data flow samples on the blockchain network layer.

Sample number	CMCNN model				Baek scheme				Essaid scheme			
	Recall	ACC	F_1-score	Test time (s)	Recall	ACC	F_1-score	Test time (s)	Recall	ACC	F_1-score	Test time (s)
1000	0.8592	0.855	0.8961	6.4	0.8211	0.7925	0.8566	3.7	0.8712	0.8415	0.8672	4.3
5000	0.8951	0.8956	0.927	24.5	0.8427	0.8324	0.8824	17.5	0.9006	0.8712	0.8953	16.4
10000	0.9211	0.9086	0.9387	44.5	0.8512	0.8499	0.9004	32.8	0.9095	0.8716	0.9001	34.7
15000	0.952	0.9342	0.9562	85.4	0.8977	0.8873	0.9253	74.6	0.9128	0.8921	0.9172	63.4
20000	0.9604	0.9481	0.9662	146.2	0.9289	0.9175	0.9442	161.2	0.9185	0.8981	0.9217	149.2
25000	0.9709	0.958	0.973	220.7	0.9351	0.9215	0.9486	237.5	0.9287	0.92	0.944	214.4
30000	0.9811	0.9716	0.9827	280.6	0.9568	0.9327	0.8566	322	0.9491	0.9264	0.9501	330.7

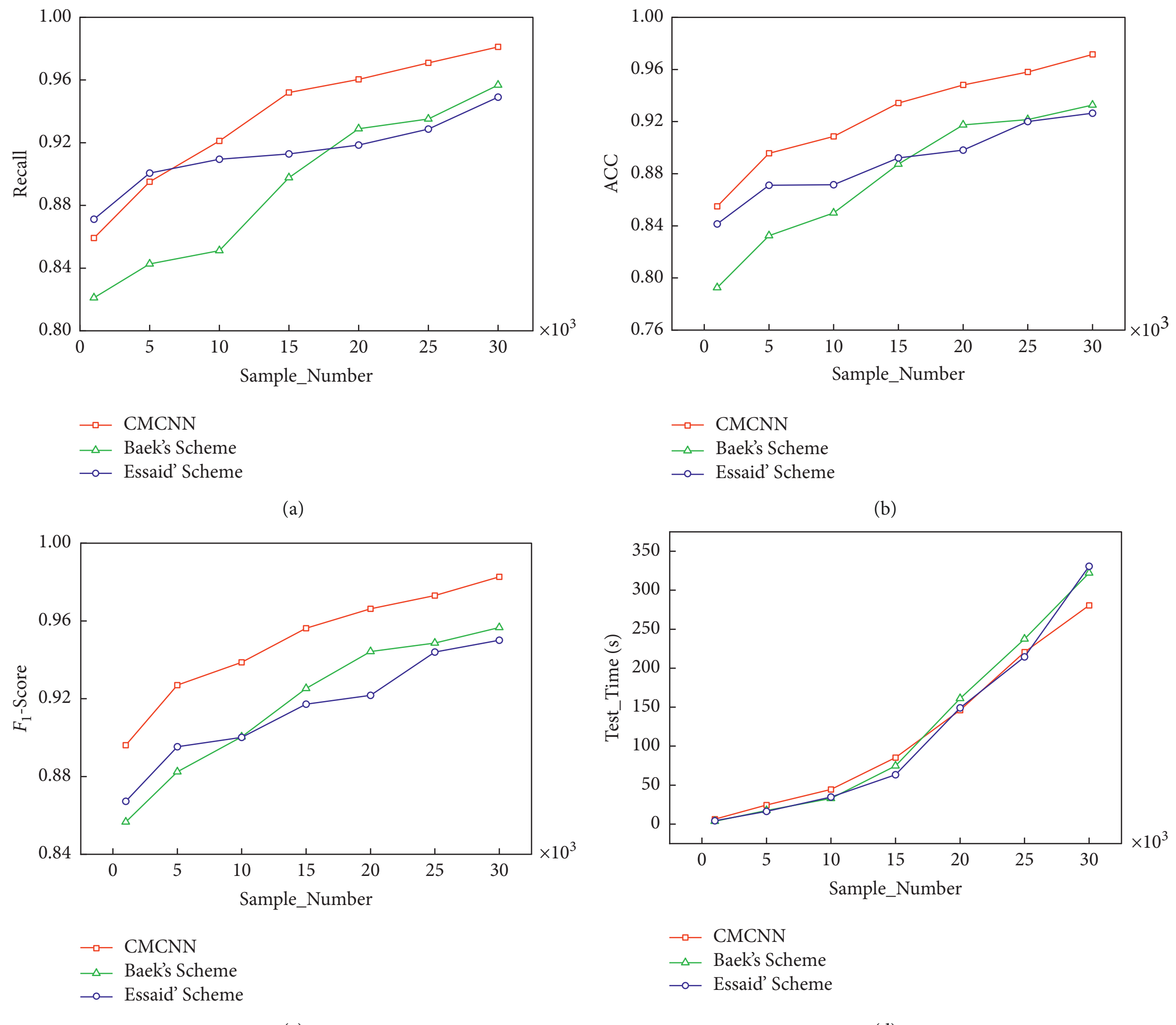


Figure 8: Comparative results of model detection under different data flow samples on the blockchain network layer. (a) Comparison of detection recall, (b) comparison of detection accuracy, (c) comparison of detection F_1-score, and (d) comparison of detection time.

attack traffic on the blockchain network layer and can also discriminate specific types of attack traffic.

Finally, the multiclass detection performances of CMCNN and the existing methods on the blockchain network layer were compared using the model data and methods proposed in the literature [30, 31]. The results are shown in Figure 11. The method proposed in this paper showed significantly higher detection performance on

Table 3: Comparative results of model detection within different time windows on the blockchain network layer.

Interval	CMCNN model				Baek scheme				Essaid scheme			
	Recall	ACC	F_1-score	FAR	Recall	ACC	F_1-score	FAR	Recall	ACC	F_1-score	FAR
10	0.7943	0.807	0.8699	0.1383	0.7876	0.7937	0.8463	0.1903	0.7779	0.794	0.8423	0.1672
20	0.8351	0.8425	0.8991	0.1187	0.7914	0.7975	0.8559	0.1833	0.7849	0.7995	0.8561	0.1542
30	0.8812	0.8817	0.9268	0.1156	0.8429	0.8433	0.8923	0.1551	0.8299	0.8353	0.8859	0.1464
60	0.9203	0.9198	0.9511	0.0829	0.8859	0.8815	0.9198	0.1331	0.8729	0.8735	0.9137	0.1246
120	0.9343	0.9338	0.9600	0.0689	0.9015	0.8905	0.9268	0.146	0.8798	0.876	0.917	0.1373
180	0.9489	0.9461	0.9682	0.0714	0.9156	0.9058	0.9387	0.1301	0.9016	0.8972	0.9321	0.1189
240	0.9578	0.9538	0.9732	0.075	0.9278	0.9221	0.9496	0.0994	0.9152	0.9091	0.9409	0.1137

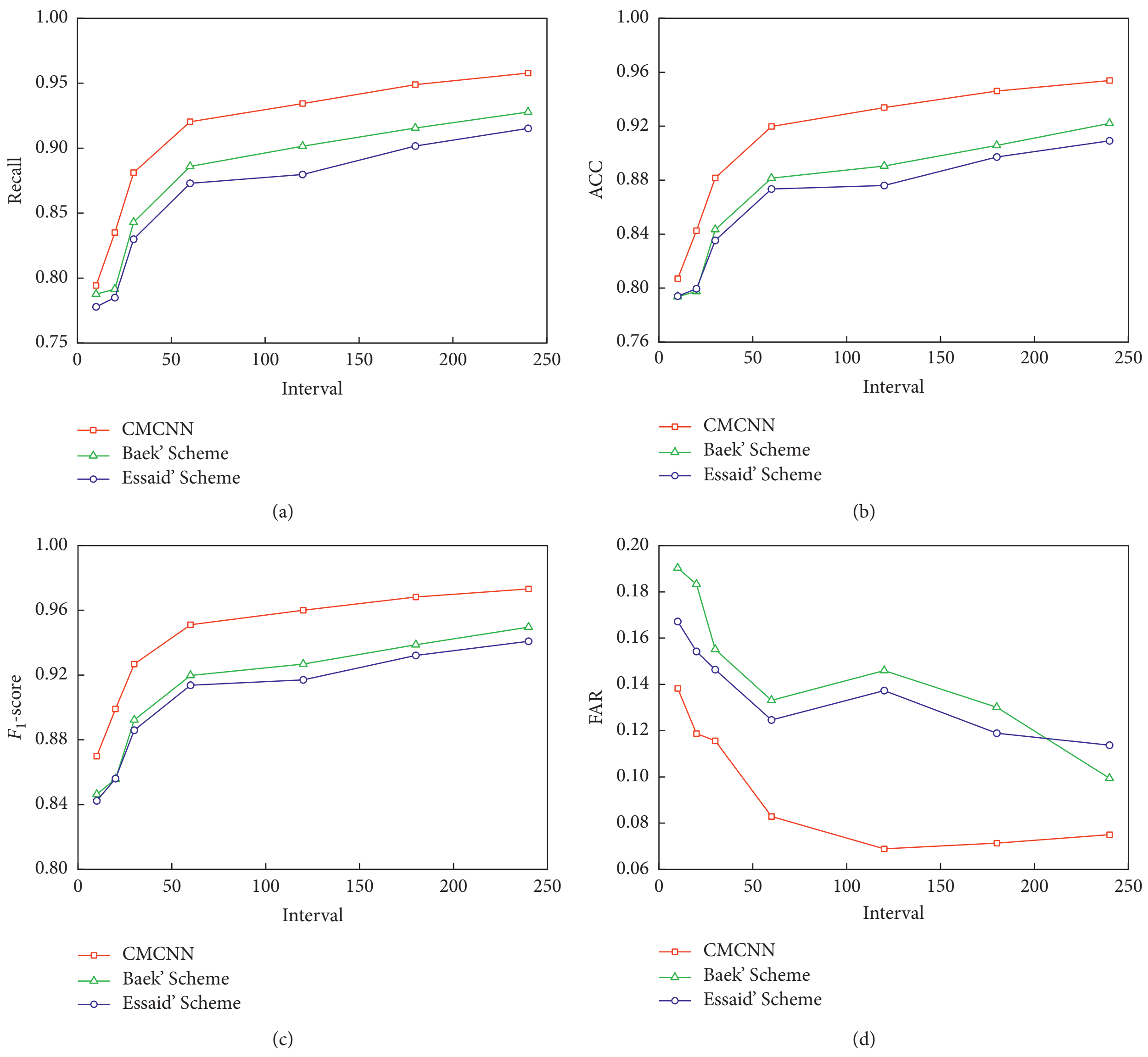


Figure 9: Comparative results of model detection within different time windows on the blockchain network layer: (a) comparison of detection recall, (b) comparison of detection accuracy, (c) comparison of detection F_1-score, and (d) comparison of detection FAR.

Table 4: Detection efficiencies over different attacks under different ratios of DDoS-attack traffic on the blockchain network layer.

Attack ratio (%)	TCP flood			UDP flood			ICMP			Redundancy P2P			Rumour		
	Precision	Recall	F_1-score	Precision	Recall	F_1-score	Precision	Recall	F_1-score	Precision	Recall	F_1-score	Precision	Recall	F_1-score
10	90.9	73.4	81.2	94.4	69.4	80	96.7	80.3	87.7	97.4	67.4	79.7	93.4	78.1	85.1
20	91.2	79.7	85.1	95.2	73.6	83	95.8	83.7	89.3	96.8	75.3	84.7	93.8	83.4	88.3
30	92.7	85.6	89.0	95.5	79.2	86.6	95.4	89.6	92.4	95.3	82.5	88.4	94.2	85.6	89.7
40	93.3	90.2	91.7	95.3	84.1	89.4	95.2	91.4	93.3	96.4	87.3	91.6	94.7	87.1	90.6
50	95.4	93.5	94.4	94.8	88.5	91.5	94.8	94.9	94.8	96.5	91.2	93.8	95.7	90.5	92.8

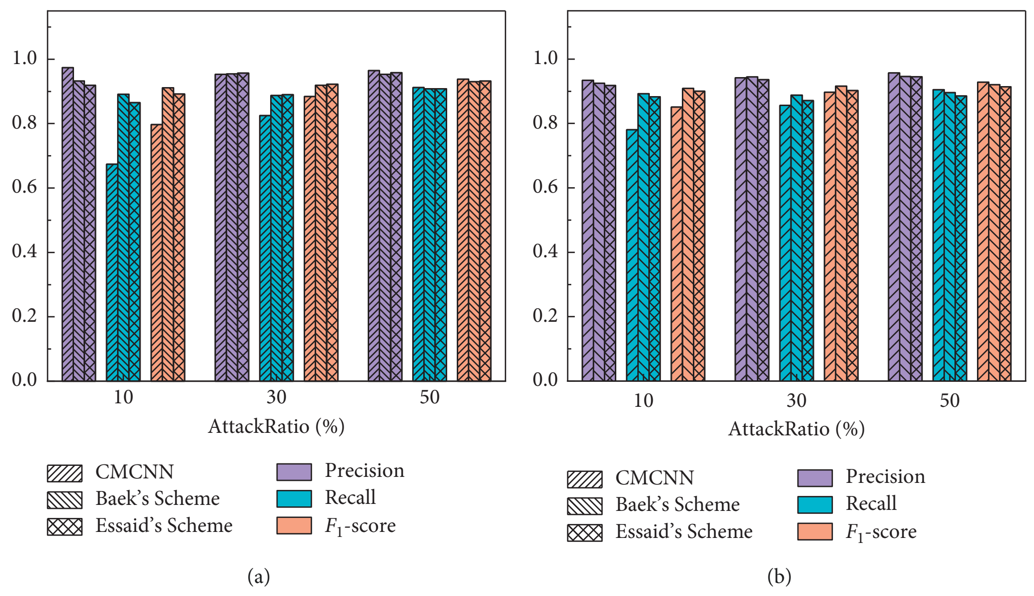


Figure 10: Comparison of detection performances over redundancy P2P and rumour-based traffic with related method: (a) performance comparison of redundant P2P traffic detection and (b) performance comparison of rumour-based traffic detection.

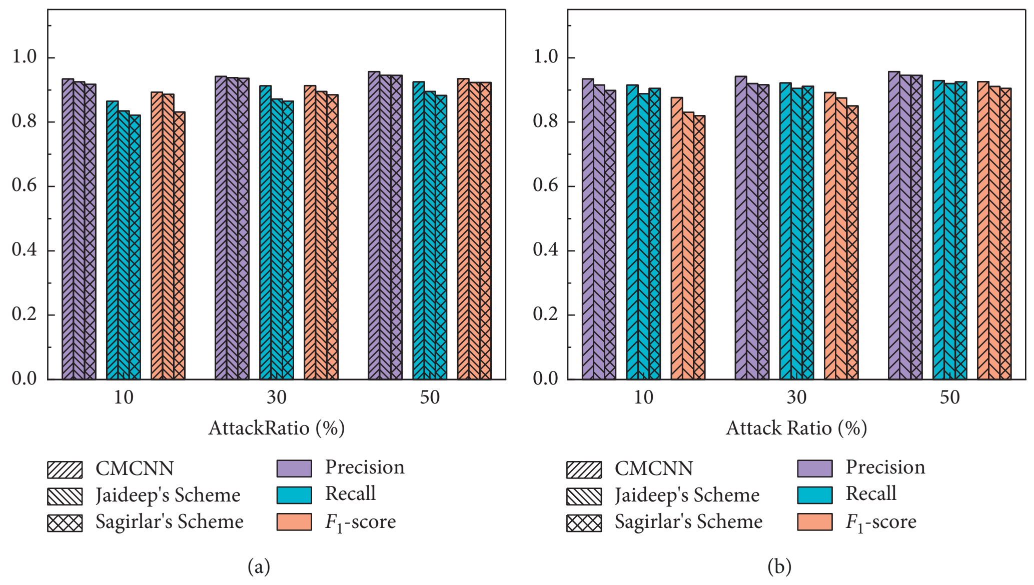


Figure 11: Comparison of detection performances over redundancy P2P and rumour-based traffic with the P2P DDoS detection method: (a) performance comparison of redundant P2P traffic detection and (b) performance comparison of rumour-based traffic detection.

rumour-based and redundancy P2P DDoS-attack traffic than the existing solutions for P2P network attacks, confirming the application value of the CMCNN model in detecting real DDoS traffic attacks on blockchain network layers.

4. Conclusions

As a typical complex network environment, a blockchain network layer is threatened by dynamically changing DDoS-attack traffic such as rumour-based and malicious redundancy P2P traffic. This study proposed a CMCNN-based detection method that effectively detects DDoS traffic. The model applies a cross-layer CNN with an L_2 regularisation term and an SSAE module based on KL divergence to mine the features of DDoS-attack traffic on the blockchain network layer. The model parameters are optimised by the ISGD algorithm. The model can effectively mitigate the problems of existing detection methods, such as low generalisability and poor accuracy rate of attack-traffic detection. In a series of experiments, the detection method based on the proposed model detected DDoS-attack traffic on the blockchain network layer with significantly higher performance than two existing methods. In future studies, dynamically self-learning, unsupervised deep-learning approaches will be considered for DDoS-attack traffic on the blockchain network layer. These solutions might improve the model's self-learning capacity for detecting attack traffic in small-sized samples.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Authors' Contributions

Qian-yi Dai and Bin Zhang contributed equally to this work.

Acknowledgments

This work was supported by the Open Fund Project of Information Assurance Technology Key Laboratory of China (Grant No. KJ-15-109) and the National Natural Science Foundation of China (Grant No. 61471344).

References

[1] S. H. E. N. Xin, P. E. I. Qing-qi, and L. I. U. Xue-feng, "Survey of block chain," *Chinese Journal of Network and Information Security*, vol. 2, no. 11, pp. 11–20, 2016.

[2] M. Vasek, M. Thornton, and T. Moore, "Empirical analysis of denial-of-service attacks in the Bitcoin ecosystem," in *Proceedings of the International conference on financial cryptography and data security*, pp. 57–71, Springer, Christ Church, Barbados, March 2014.

[3] Blockchain threat intelligence, "Blockchain threat intelligence," 2018, https://bti.slowmist.com/.

[4] R. Stephen and A. Alex, "A review on blockchain security," in *Proceedings of the IOP Conference Series: Materials Science and Engineering*, vol. 396, no. 1, April 2018, Article ID 012030.

[5] X. Han, Y. Yuan, and F.-Y. Wang, "Security problems on blockchain: the state of the art and future trends," *Acta Automatica Sinica*, vol. 45, no. 1, pp. 206–225, 2019.

[6] A. Abhishta, R. Joosten, S. Dragomiretskiy, and L. J. M. Nieuwenhuis, "Impact of successful ddos attacks on a major crypto-currency exchange," in *Proceedings of the 2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pp. 379–384, IEEE, Pavia, Italy, February 2019.

[7] F. Tramèr, D. Boneh, and K. Paterson, "Remote side-channel attacks on anonymous transactions," in *Proceedings of the 29th {USENIX} Security Symposium ({USENIX} Security 20)*, pp. 2739–2756, Boston, MA, USA, August 2020.

[8] C. C. Ye, G. Q. Li, H. M. Cai, and Y. G. Gu, "Security detection model of blockchain," *Ruan Jian Xue Bao/Journal of Software*, vol. 29, no. 5, pp. 1348–1359.

[9] J. Tapsell, R. N. Akram, and K. Markantonakis, "An evaluation of the security of the bitcoin peer-to-peer network," in *Proceedings of the 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 1057–1062, IEEE, Halifax, NS, Canada, August 2018.

[10] G. H. Tian, Y. H. Hu, and X. F. Chen, "Research progress on attack and defense techniques in block-chain system," *Journal of Software*, vol. 32, no. 5, pp. 1495–1525, 2021.

[11] A. R. Jamader, P. Das, and B. R. Acharya, "BcIoT: blockchain based DDoS prevention architecture for IoT," in *Proceedings of the 2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, pp. 377–382, IEEE, Madurai, India, May 2019.

[12] J. Rüth, T. Zimmermann, K. Wolsing, and O. Hohlfeld, "Digging into browser-based crypto mining," in *Proceedings of the Internet Measurement Conference 2018*, pp. 70–76, Boston MA USA, October 2018.

[13] U. J. Baek, S. H. Ji, and J. T. Park, "DDoS attack detection on bitcoin ecosystem using C-learning," in *Proceedings of the 2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 1–4, IEEE, Matsue, Japan, September 2019.

[14] M. Essaid, D. Y. Kim, S. H. Maeng, S. Park, and H. T. Ju, "A collaborative DDoS mitigation solution based on ethereum smart contract and RNN-LSTM[," in *Proceedings of the 2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 1–6, IEEE, Matsue, Japan, September 2019.

[15] M. Mirkin, Y. Ji, J. Pang, A. K. Mundt, I. Eyal, and A. Juels, "BDoS: Blockchain Denial-of-Service Attacks," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, November 2020.

[16] T. Idé, "Collaborative anomaly detection on blockchain from noisy sensor data," in *Proceedings of the 2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 120–127, IEEE, Singapore, November 2018.

[17] B. Jia and Y. Liang, "Anti-D chain: a lightweight DDoS attack detection scheme based on heterogeneous ensemble learning in blockchain," *China Communications*, vol. 17, no. 9, pp. 11–24, 2020.

[18] A. Eduardo, J. Sousa, V. C. Oliveira et al., “Fighting underprice DoS attack in ethereum with machine learning techniques,” *ACM SIGMETRICS - Performance Evaluation Review*, vol. 48, no. 4, pp. 24–27, 2021.

[19] A. Gervais, G. O. Karame, V. Glykantzis, H. Ritzdorf, and S. Capkun, “On the security and performance of proof of work blockchains,” in *Proceedings of the ACM Sigsac Conf. on Computer and Communications Security*, pp. 3–16, Vienna Austria, October 2016.

[20] M. Saad, M. T. Thai, and A. Mohaisen, “POSTER: deterring DDoS attacks on blockchain-based cryptocurrencies through mempool optimization,” in *Proceedings of the ASIACCS*, pp. 809–811, Incheon, Korea, June 2018.

[21] A.-T. Pănescu and V. Manta, “Smart contracts for research data rights management over the ethereum blockchain network,” *Science & Technology Libraries*, vol. 37, no. 3, pp. 235–245, 2018.

[22] T.-T. Kuo, J. Kim, and R. A. Gabriel, “Privacy-preserving model learning on a blockchain network-of-networks,” *Journal of the American Medical Informatics Association*, vol. 27, no. 3, pp. 343–354, 2020.

[23] S. Delgado-Segura, C. Pérez-Solà, J. Herrera-Joancomartí, G. N. Arribas, and J. Borrell, “Cryptocurrency networks: a new P2P paradigm,” *Mobile Information Systems*, vol. 2018, Article ID 2159082, 16 pages, 2018.

[24] H. Wang, Z. Cao, and B. Hong, “A network intrusion detection system based on convolutional neural network,” *Journal of Intelligent and Fuzzy Systems*, vol. 38, no. 6, pp. 7623–7637, 2020.

[25] N. Y. Almusallam, Z. Tari, P. Bertok, and A. Y. Zomaya, “Dimensionality reduction for intrusion detection systems in multi-data streams-A review and proposal of unsupervised feature selection scheme,” *Emergent Computation*, pp. 467–487, 2017.

[26] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: surpassing human-level performance on imagenet classification,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034, Santiago, Chile, December 2015.

[27] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. JMLR Workshop and Conference Proceedings*, pp. 249–256, Sardinia, Italy, May 2010.

[28] W. Jung and S. Park, “Preventing DDoS attack in blockchain system using dynamic transaction limit volume,” *International Journal of Control and Automation*, vol. 10, no. 12, pp. 131–138, 2017.

[29] Ahlashkari, “CICFlowmeter-V4.0,” 2022, https://github.com/ahlashkari/CICFlowMeter.

[30] G. Jaideep and B. P. Battula, “A framework for agent-based detection and prevention of DDoS attacks in distributed p2P networks,” in *Pervasive Computing: A Networking Perspective and Future Directions*, pp. 15–30, Springer, Singapore, 2019.

[31] G. Sagirlar, B. Carminati, and E. Ferrari, “AutoBotCatcher: blockchain-based P2P botnet detection for the internet of things,” in *Proceedings of the 2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, pp. 1–8, IEEE, Philadelphia, PA, USA, October 2018.