American Journal of Innovation in Science and Engineering (AJISE)

ISSN: 2158-7205 (ONLINE)

VOLUME 5 ISSUE 1 (2026)

PUBLISHED BY
E-PALLI PUBLISHERS, DELAWARE, USA

American Journal of Innovation in Science and Engineering (AJISE)

Volume 5 Issue 1, Year 2026
ISSN: 2158-7205 (Online)
DOI: https://doi.org/10.54536/ajise.v5i1.7204
https://journals.e-palli.com/home/index.php/ajise

A Lightweight WDGP-1DCSP Model for High-Precision Intrusion Detection in Resource-Constrained IoT Edge Devices

Hani Iwidat[1*]

Article Information

Received: December 10, 2026
Accepted: March 03, 2026
Published: March 18, 2026

Keywords

Convolutional Neural Network, Cross-Stage Partial (CSP) Networks, Generative Adversarial Networks (GANs), Internet of Things (IoT), Intrusion Detection System (IDS), Lightweight Deep Learning

ABSTRACT

As technology advances, network intrusion techniques are diversifying, presenting significant security challenges for resource-constrained edge devices in IoT environments. Addressing the generally poor detection performance and unsuitability of traditional intrusion detection models for edge devices in the current IoT environment, which suffer from limited resources and low computing power, this paper proposes a lightweight model based on Generative Adversarial Networks (GANs) and Convolutional Neural Networks (CNNs) for detecting intrusion behavior in the IoT environment. Firstly, GAN technology is used to solve the data imbalance problem. Secondly, a lightweight CNN with a cross-stage local structure is used to extract traffic features, and H-swish is selected as the activation function to reduce computational load and improve computational efficiency. Finally, Softmax is used to classify the traffic data. Experimental results on the TON_IOT and CICIDS2018 datasets demonstrate the model's exceptional performance. The proposed model achieved accuracy rates of 99.52% and 99.53%, precision rates of 99.41% and 99.28%, recall rates of 99.57% and 99.52%, and F1-scores of 99.44% and 99.37% on the TON_IOT and CICIDS2018 datasets, respectively. The model size is limited to 21-32 KB. These results demonstrate that the proposed model, while maintaining intrusion detection accuracy, reduces model size and computational load, thereby meeting the high-precision intrusion detection requirements of demanding IoT environments.

INTRODUCTION

With the rapid development of Internet of Things (IoT) technology, its application scope has been continuously expanding, covering multiple fields such as public services, Internet of Vehicles (IoV), smart retail, and smart homes, significantly enhancing everyday convenience and quality of life. However, the popularization of IoT has also brought increasingly severe security threats. Data transmission between IoT devices usually takes place in an open network environment, making them vulnerable to abnormal traffic attacks. This abnormal traffic may manifest as excessive data requests, suspicious communication patterns, or unauthorized access attempts, which may lead to device malfunction or leakage of sensitive information in severe cases. Therefore, building an effective intrusion detection system (IDS) is the key to maintaining IoT security.

Currently, the academic community has conducted extensive research on intrusion detection, yielding promising results (Chen *et al.,* 2022). Early intrusion detection systems mainly relied on rule-based and feature-matching approaches, which are effective against known attacks but limited in detecting novel or unknown threats. With the development of machine learning technology, researchers have begun to use decision trees (Sarhan *et al.,* 2024), support vector machines (Hadem *et al.,* 2021), random forests (Hamidou & Mehdi, 2025) and other methods to identify abnormal traffic. (Muhammad, 2026) provides a comprehensive review of Zero Trust Architecture (ZTA) design and implementation, which complements intrusion detection by emphasizing continuous verification in IoT networks. These methods can detect traffic that deviates from normal behavior by learning normal traffic patterns. However, with technological advancements, network attack methods have become increasingly diverse, has resulted in declining performance" for better parallelism in machine learning-based intrusion detection systems, which rely on manual feature engineering and struggle with large-scale data. In recent years, the emergence of deep learning has brought new opportunities for intrusion detection, and numerous deep learning-based intrusion detection studies have been published (Li *et al.,* 2022), such as convolutional neural networks (Xiao *et al.,* 2019) (CNN), recurrent neural networks (Zhao *et al.,* 2023) (RNN), and autoencoders (Khan *et al.,* 2022) (AE). However, since traditional intrusion detection models are deployed on the hosts, they cannot adapt to the limited resources and low computing power of edge nodes in the Internet of Things environment. Researchers have begun to explore new methods to optimize intrusion detection systems to better adapt to the security needs of the Internet of Things. Jiménez-López *et al.* (Jiménez-López *et al.,* 2025) proposed a lightweight intrusion detection model that

[1] Department of Data Science, Al-Istiqlal University, Jericho, Palestine
[*] Corresponding author's e-mail: hani.iwidat@pass.ps

combines one-dimensional convolutional neural networks and differential privacy, which reduces the number of model parameters and makes it lightweight by manually modifying the number of convolution channels. Akpaku *et al.* (E. Akpaku *et al.,* 2025) proposed a model based on attention mechanism and Bi-Directional temporal convolutional neural network (Bi-Directional Temporal Convolutional Network). The lightweight BiTCN model reduces parameters through its structure and enhances detection via an attention mechanism that dynamically weights input parts (Ernest Akpaku *et al.,* 2025).

To further improve the detection accuracy of the lightweight intrusion detection model, researchers use oversampling methods to balance the dataset. For example, Zhang *et al.* (Zhang *et al.,* 2020) proposed an oversampling method that combines the SMOTE method (Synthetic Minority Over-Sampling Technique) and the Gaussian Mixture Model (GMM); Araujo-Filho *et al.* (Freitas de Araujo-Filho *et al.,* 2021) proposed an oversampling method based on Generative Adversarial Network (GAN). These oversampling methods help the model learn the features of all categories better by balancing the dataset, thereby improving the model's intrusion detection performance and robustness.

Recent works on lightweight IDS for IoT include Misrak *et al.* (Misrak & Ayele, 2025) with DNN-BiLSTM and quantization on CIC-IDS2017; (Wisanwanichthan *et al.,* 2025) using knowledge distillation for DNNs on CIC-IDS2017; and Hnamte *et al.* (Hnamte *et al.,* 2025) proposing LWIDS-DCNN on CICIDS2017 and InSDN. These approaches focus on resource efficiency, complementing our model's performance on TON_IOT and CICIDS2018.

In summary, current IoT intrusion detection systems often face challenges like incompatibility with resource-limited environments and suboptimal performance. To address these issues, this paper introduces a novel lightweight intrusion detection model based on generative adversarial networks (GANs) and convolutional neural networks (CNNs). The main contributions are as follows:

1. A generative adversarial network model, WDGP (Wasserstein Deep Convolutional GAN with Gradient Penalty), is proposed, combining the advantages of WGAN-GP (Wasserstein GAN with Gradient Penalty) and DCGAN (Deep Convolutional Generative Adversarial Network). WDGP effectively solves the problems of training instability and mode collapse in traditional GANs by utilizing the loss function and gradient penalty mechanism of WGAN-GP. Furthermore, the deep convolutional network structure provided by DCGAN significantly improves the quality of data generated by the model, resolving the data imbalance problem in intrusion detection training and enhancing the accuracy and robustness of the model's intrusion detection capabilities.

2. A novel lightweight intrusion detection model is proposed, combining the advantages of one-dimensional convolutional neural networks (1DCNNs) and cross-stage partial (CSP) structures. 1DCSP, which utilizes the efficient feature extraction capability of 1DCNN to enable the model to quickly extract key features of traffic in intrusion detection. Then, by introducing the CSP structure, the number of model parameters and computational complexity are reduced, while maintaining the integrity of feature expression. This lightweight design enables the model to achieve efficient intrusion detection at a low computational cost in resource-constrained environments.

3. Experiments demonstrate 99.52% accuracy, 99.41% precision, 99.57% recall, and 99.44% F1 score on the TON_IOT (Booij *et al.,* 2022) dataset with a size of 21 kB; and 99.53% accuracy, 99.28% precision, 99.52% recall and 99.37% F1 score on the CICIDS2018 (Kilincer *et al.,* 2021) dataset with a size of 32 kB. Experiments were conducted on two datasets using the proposed model. The inference times for 128 samples were 2.253 ms and 20.390 ms, respectively. Experiments demonstrate that the lightweight architecture and real-time performance make it possible to deploy the model for intrusion detection on edge devices in an IoT environment.

MATERIALS AND METHODS

Dataset Preparation and Preprocessing

The experiments in this paper were conducted on the CICIDS2018 and TON_IOT datasets. The TON_IOT dataset consisted of data containing 44 features and 10 traffic types, while the CICIDS2018 dataset consisted of data containing 78 features and 11 traffic types.

Raw traffic feature data must be cleaned and labeled before use as model input. This paper follows the following steps for dataset preprocessing:

• Feature Reduction and Removal: Non-essential features in the CIC-IDS-2018 dataset, such as destination port and timestamp, were filtered using expert judgment to reduce dimensionality. The CIC-IDS-2018 dataset includes non-essential features, such as destination port and timestamp, which are accessible to both intruders and users. To reduce the dimensionality of these features, expert judgment was employed to filter them.

• Outlier Handling: The dataset contained data that the model cannot recognize, such as NAN and NULL values. Such outliers hinder effective model training. The dropna function was used to remove these outliers.

• Data Standardization: Different features have different dimensions and value ranges. To avoid the impact of inconsistent feature dimensions on model classification, the LabelEncoder function was first used to convert non-numerical features in the traffic feature data into numerical features to ensure that all features are numerical values. Then, the StandardScaler function was used to standardize the data and eliminate the influence of dimensions.

Intrusion Detection Model Based on WDGP and 1DCSP

Introduction to WDGP

Generative Adversarial Networks (GANs) are deep learning models proposed by Goodfellow *et al.*

(Goodfellow *et al.,* 2014) in 2014. They are mainly used to generate high-quality new data similar to the original data. The core concept of GANs involves training through adversarial competition between two neural networks: a generator and a discriminator. These two networks are the generator and the discriminator. The goal of the generator is to learn to make the generated data as close as possible to the real data to "deceive" the discriminator. The goal of the discriminator is to learn to judge the data generated by the generator as fake as possible while judging the real data as real. The specific process is shown in Figure 1.

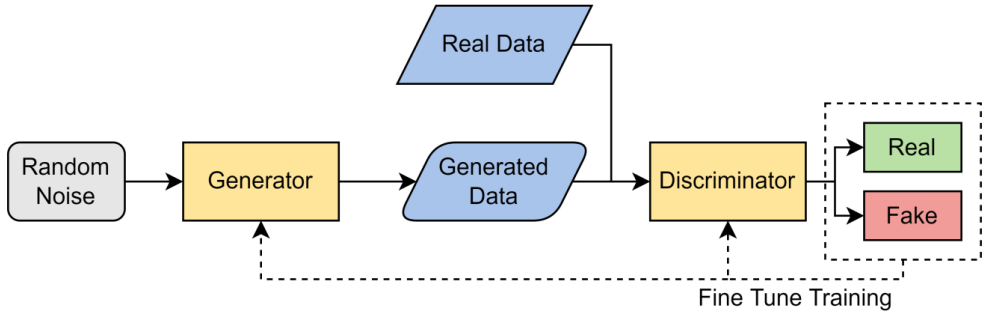

Figure 1: Architecture of GANs training.

In Figure 1, D Loss is the sum of the discriminator's loss on real data and its loss on generated data, which is responsible for guiding the discriminator to optimize; G Loss is the loss of the data generated by the generator being identified as real data by the discriminator, which is responsible for guiding the generator to optimize. Figure 1 illustrates that GAN training can be viewed as a two-player minimax game. The generator and the discriminator compete against each other and alternately optimize different objective functions to train the model, to achieve making the discriminator unable to distinguish between real data and generated data. The GAN loss function is shown in Equation (1):

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim P_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim P_z(z)}[\log\{1 - D[G(z)]\}]) \quad (1)$$

Where: D represents the discriminator's loss, G represents the generator's loss, x~P_(data(x)) represents sampling from real data, x~Pz (z) represents sampling from fake data generated by the generator, D(x) is the discriminator's output on real data x, and D[G(z)] is the discriminator's output on fake data G(z) generated by the generator. WGAN (Arjovsky *et al.,* 2017) (Wasserstein GAN) is an improvement based on traditional GANs, proposed to solve issues of instability and mode collapse in GAN training. The proposed improvement method aims to solve the problems of instability and model collapse in GAN training. It improves the loss function by introducing Wasserstein distance, thereby making the training more stable. Compared with the JS divergence loss function used by GAN, the Wasserstein distance loss function of WGAN provides a more intuitive distance metric, especially in cases of imbalanced data and extreme situations, which can help the model optimize parameters more effectively. After using Wasserstein distance, the discriminator output of WGAN is no longer output probability, but a scoring function used to measure the "realness" of the sample. The loss function in WGAN becomes Equation (2):

$$L_C = \mathbb{E}_{z\sim P_z(z)}[C(G(z))] - \mathbb{E}_{x\sim P_{data}(x)}[C(x)] \quad (2)$$

Where; C(x) is the score of the discriminator for the real sample x, and C[G(x)] is the score of the discriminator for the generated sample. As can be seen from equation (2), the overall performance of WGAN only depends on the score of the discriminator. The higher the score, the better the generator performance. WGAN-GP (Gulrajani *et al.,* 2017) is an important improvement of WGAN. It mainly introduces gradient penalty to replace the traditional weight pruning, so as to further improve the training stability and generation quality. Compared with WGAN using weight pruning to maintain the Lipschitz continuity of the discriminator, that is, limiting the weight of the discriminator to a certain range after each update, WGAN-GP introduces a gradient penalty mechanism to directly penalize the gradient of the discriminator. This not only improves the training stability and generation quality of the model, but also effectively ensures Lipschitz continuity. Its loss function changes as shown in equation

$$L_C = \mathbb{E}_{z\sim P_z(z)}[C(G(z))] - \mathbb{E}_{x\sim P_{data}(x)}[C(x)] + \lambda \mathbb{E}_{\hat{x}}[(\| \nabla D(\hat{x}) \|_2 - 1)^2] \quad (3)$$

Where $\hat{x}$ is the sample interpolated between the real data and the generated data, and λ is the parameter that controls the penalty intensity. This paper builds upon WGAN-GP to propose the WDGP model. The WDGP model combines the benefits of WGAN-GP and DCGAN (Radford *et al.,* 2016), and retains the advantages of both. The changes in the discriminator loss function in WGAN-GP, combined with the changes in the generator and discriminator architectures in DCGAN, have the advantage of improving the quality of the generator's generated samples while making the training process of the generative adversarial network model more stable and efficient. The loss function of the WDGP discriminator is the same as that of WGAN-GP, as shown in Equation (3). The architecture of the WDGP generator and discriminator is the same as that of DCGAN. Compared with the fully connected architecture of the generator and discriminator in WGAN-GP, WDGP adds convolutional operations to the generator

to help it generate traffic features with spatial structure and details. Such feature data is more consistent with real traffic features. Convolutional operations are added to the discriminator to improve the discriminator's feature extraction capability, helping the discriminator to better distinguish between real samples and generated samples. At the same time, due to the improved performance of the discriminator, it can also help the generator avoid overfitting during the training process. The specific architectures of the WDGP generator and discriminator are shown in Table 1 and Table 2.

Table 1: WDGP Generator Architecture

Layer	Kernel	Output	Parameters	Activation
In	-	128×8×200	0	-
Conv	7×7	128×4×194	228	LeakyReLU
Conv	5×5	128×2×190	42	LeakyReLU
Conv	3×3	128×1×188	7	LeakyReLU
FC	-	128×1×66	12 870	Tanh
Out	-	128×1×66	-	-

Table 2: WDGP Discriminator Architecture

Layer	Kernel	Output	Parameters	Activation
In	-	128×1×66	0	-
Conv	3×3	128×2×194	8	LeakyReLU
Conv	5×5	128×4×190	44	LeakyReLU
Conv	7×7	128×8×188	232	LeakyReLU
FC	-	128×200	86 600	LeakyReLU
FC	-	128×1	201	-
Out	-	128×1	-	-

In the generator architecture shown in Table 1, three convolutional layers and one fully connected layer are used to generate features from the traffic data. The input consists of 128 random noise vectors with a dimension of 8 × 200. Using high-dimensional noise vectors allows the generator to explore and generate more diverse features. If the noise vector dimension is too low, the generator may be limited in mapping to real data and fail to capture the complexity of the real data. The input noise vectors are processed through three convolutional layers with decreasing kernel sizes, followed by a fully connected layer that maps them to high-dimensional vectors simulating real traffic data. Using the LeakyReLU activation function instead of the ReLU function in the convolutional operations effectively avoids the "dead neuron" problem caused by ReLU in some cases and allows a small portion of the gradient to propagate along the negative half-axis, thus preventing the gradient vanishing problem. The Tanh activation function is used in the final fully connected layer to ensure that the traffic sample data generated by the generator is the same as the normalized traffic data in reality. (See Table 2) In the architecture of the discriminator, three convolutional layers and two fully connected layers are used to discriminate the traffic data. Its input is 128 traffic feature vectors. Corresponding to the convolutional operation of the generator, the discriminator fully extracts the features of the input data through three convolutional operations with convolutional kernels from small to large, and then uses two fully connected layers to output the results. Since WDGP adopts the same discrimination method as WGAN-GP discriminator, unlike the traditional discriminator which uses Sigmoid activation function in the last fully connected layer, it does not need to output the probability between [0,1], but outputs a real number to evaluate the authenticity of the data. Therefore, linear activation is used in the end, that is, without changing the value of the input data, it directly returns the input data itself.

1DCSP Introduction

CSPNet (Cross Stage Partial Network) is a deep learning architecture used to improve the efficiency and performance of convolutional neural networks. It was first proposed by Wang *et al.*(Wang *et al.,* 2020) in 2020, aiming to optimize the computational performance of the network, reduce the model complexity, and maintain high accuracy. CSPNet The main idea is to improve the model's learning ability and reduce redundant computation by fusing features locally across stages. The specific structure is shown in Figure 2.

In Figure 2, Conv mainly extracts important features from the input data through convolution operations; Copy is responsible for copying the input data, preserving the original information, and avoiding the loss of key information during convolution operations; Concat is responsible for concatenating the convolution-processed feature map with the copied original feature map along

the channel dimension to achieve feature fusion.

As shown in Figure 2, the CSP structure divides the input feature map into two parts. One part is directly output to the result through Copy, and the other part is convolutional in the set Block to extract features. After feature extraction, the result is merged with the other part of the data on the channel to achieve feature fusion.

H-swish is a new activation function proposed by

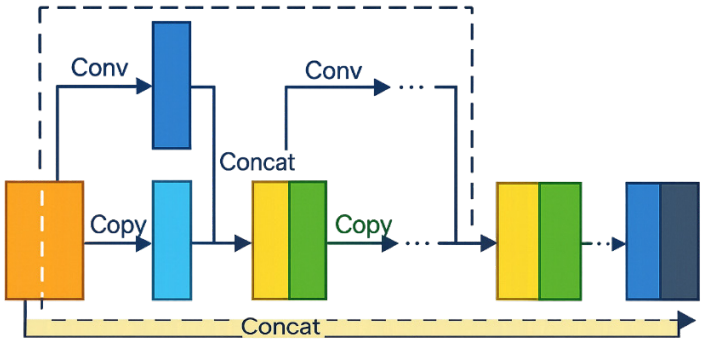

Figure 2: CSP structure

the Google team in its MobileNetV3 (Howard *et al.,* 2019) architecture. It is designed for lightweight neural networks to improve the performance and computational efficiency of the model, especially suitable for IoT and edge computing scenarios. The H-swish calculation formula is:

$$H\text{-swish}(x)=x\,(\mathrm{ReLU}\,(x+3))/6 \quad (4)$$

Compared with the non-continuous characteristic of ReLU formula, which directly converts negative input values to 0, H-swish combines linear and non-linear characteristics. Its design makes it behave like ReLU when the input value is negative, while providing richer feature information in the positive value range. It is a smooth activation function.

This design mitigates gradient vanishing, enhances feature representation, and improves training and inference efficiency by preserving more information.

This paper proposes the 1DCSP module based on

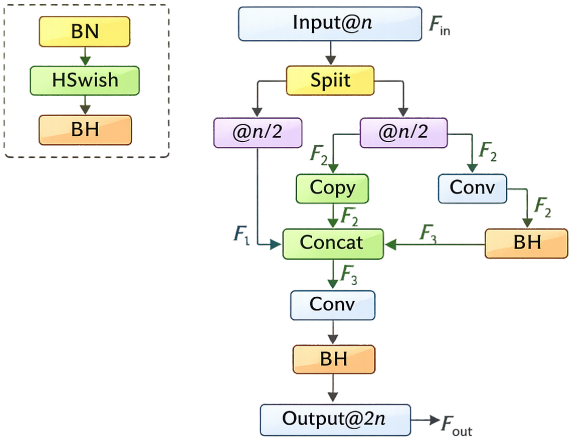

Figure 3: 1DCSP Module Structure.

1DCNN, combining the characteristics of CSP and H-swish, as shown in Figure 3. In Figure 3, Split is used to segment the input data channels; BN (Batch Normalization) is used to accelerate training and improve model stability; the activation function H-swish is used to help the model learn complex features and patterns while optimizing computational efficiency; @n represents the current number of data channels as n; BH is a combination of BN and H-swish.

When choosing the number of channels to segment, too few shallow feature channels will lead to information loss, while too many deep channels may lead to excessive computation. Based on the complexity of the intrusion detection task and computational resources, the 1DCSP module divides the input feature Fin channel into two parts [F1, F2] according to the ratio of [50%, 50%], where F1 preserves the input features. F1 contains the shallow features of the original information, including basic features from simple patterns, ensuring that important information is not lost during the extraction of deeper features. F2 is similar to F1 but has different functions; it serves as the benchmark for extracting deeper features,

providing a reference for subsequent deep feature extraction. After successfully segmenting the channels, local feature extraction is performed on F2. First, an unmodified feature information is preserved through a copy operation. Then, a Conv operation with a kernel of 3 and a batch normalization (BH) operation are used to extract deeper features F3 from F2. F3 is a deep feature obtained from the original information after convolution and activation processing, which helps the model better understand high-level patterns in the data, especially the deep correlations that need to be captured in complex tasks. Afterwards, F1, F2, and F3 are concatenated according to channels using a Concat operation. The concatenated features retain both shallow and deep information, allowing the model to focus on both low-level details and high-level abstractions. Finally, to achieve cross-stage local feature fusion, the concatenated features are processed by a Conv operation with a kernel of 1. The operation and BH operation processing help the model effectively integrate information from different levels to obtain the feature Fout that achieves cross-stage local fusion. The specific operation is shown in Equation (5) and Equation (6):

$$F_3 = \mathrm{BH}[\mathrm{Conv}(F_2)] \quad (5)$$

$$F_{out} = \mathrm{BH}\{\mathrm{Conv}[\mathrm{Concat}(F_1, F_2, F_3)]\} \quad (6)$$

Through the cross-stage local fusion feature method and H-swish activation function, the 1DCSP module can effectively improve the feature learning ability of the model, while reducing the number of parameters required for model training, thereby achieving the purpose of lightweighting the model and improving the model detection performance.

The specific structure of the 1DCSP model is shown in Figure 4, which consists of 1DCNN, 1DCSP, Maxpool1D and FC. Among them, 1DCNN and 1DCSP are responsible for feature extraction; Maxpool1D is the max pooling layer, which is responsible for reducing feature size and extracting key features; FC is the fully connected layer, which is responsible for integrating features and prediction results.

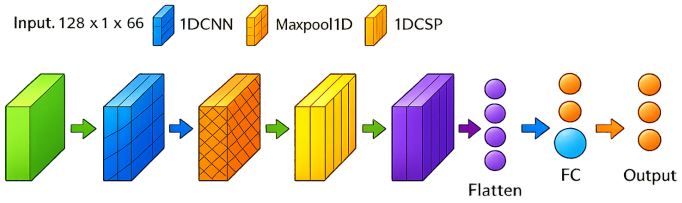

Figure 4: 1DCSP model structure

Specific Process of Model Training

This paper proposes an intrusion detection model WDGP-1DCSP based on GAN and CNN. The specific training process is shown in Figure 5. As shown, the model training process includes three stages: data preprocessing, model training, and model testing. First, data preprocessing includes feature reduction, outlier handling, and data standardization. Second, model training includes dataset partitioning, WDGP model training and sample generation, and 1DCSP model training. Third, model testing involves testing the trained model on a test set and analyzing the results.

First, the original feature data is preprocessed to become suitable input data for the model. Then, by statistically analyzing various types of traffic data, minority class samples requiring oversampling are input into WDGP for training. WDGP mainly consists of two models: a generator and a discriminator. The generator is constructed using a convolutional-fully connected architecture, with input being reconstructed noisy data and output being standardized, normalized feature data. The discriminator is also constructed using a convolutional-fully connected architecture, with input being standardized, normalized feature data and outputting a real value that scores the sample's authenticity, determining whether the data is genuine. In WDGP... During training, the generator and discriminator improve their performance through adversarial methods. Once the generator can fool the discriminator, the trained generator model is saved as a tool for generating data. After dividing the dataset into training and test sets in an 8:2 ratio, the saved generator model is used to generate minority class samples to balance the training set. The training set is then fed into the 1DCSP model for training. The 1DCSP model is constructed using a 1DCNN-pooling-1DCSP-fully connected approach. The input is standard normalized feature data, and the output is the traffic type determined by the model. Since the input data is a single channel, and the 1DCSP module requires at least two channels for partitioning, the data is first pre-processed using 1DCNN to generate two channels for subsequent use by the 1DCSP module while extracting features. In the 1DCSP module, the input data is processed as shown in Figure 3. The cross-stage local fusion feature method shown yields richer feature representations. This method not only effectively reduces the computational load of the model but also enhances the model's ability to extract features. It enables the model to better capture feature information at different levels when processing complex traffic data, thereby helping to reduce model weight and improve intrusion detection performance. Finally, the trained 1DCSP model is tested on the test set, and the experimental results are analyzed.

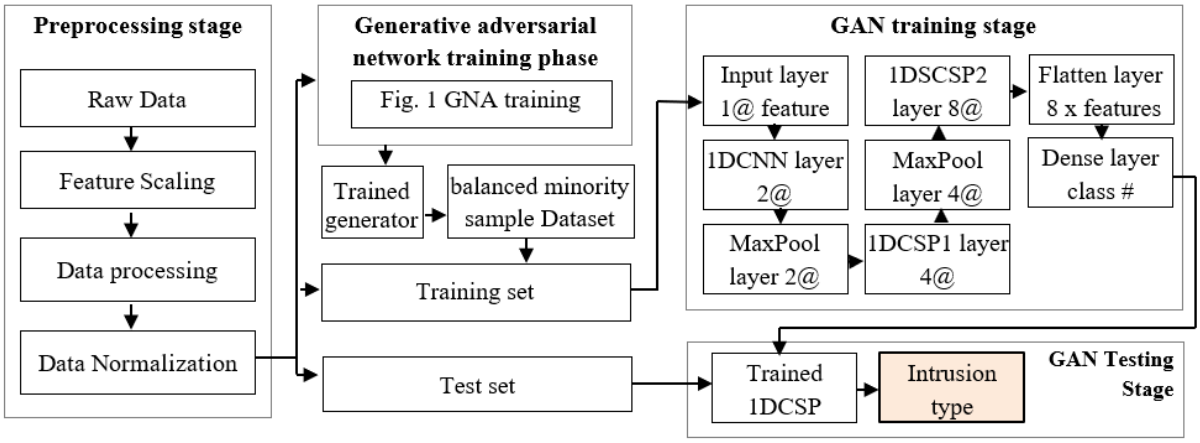

Figure 5: WDGP-1DCSP Architecture

RESULTS AND DISCUSSIONS

Experimental Setup and Evaluation Metrics

The experiments in this paper were conducted on a 64-bit Intel® Core™ i7-12700H CPU@ 2.30GHz computer, equipped with 16 GB RAM and a Python-based Nvidia GeForce GTX 3060 GPU (16 GB). The model in this paper was implemented using the PyTorch library in Python. The relevant parameters of the model are shown in Table 3.

For a classification problem, the classification results are shown in Table 4.

Table 3: Model-related parameters

Model Parameter	Parameter Content
epoch	50
Batch Size	128
Loss Function	CrossEntropyLoss
Optimizer	Adam
Optimizer Learning Rate	0.005
Dropout	0.5
Dataset Split Ratio n_1/n_2	0.2/0.8

Table 4: Classification results

Class	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

The evaluation metrics are accuracy, precision, recall, and F1 score, as shown in equations (7) - (10):

$$\text{Accuracy}=(TP+TN)/(TP+FP+TN+FN) \quad (7)$$

$$\text{Precision}=TP/(TP+FP) \quad (8)$$

$$\text{Recall}=TP/(TP+FN) \quad (9)$$

$$\text{F1-score}=(2\times\text{Precision}\times\text{Recall})/(\text{Precision}+\text{Recall}) \quad (10)$$

Experimental Results and Analysis

The intrusion detection experiment was divided into two categories (intrusive and non-intrusive) and multiclass intrusion detection.

Comparative Analysis of Two-Class Intrusion Detection Models

In order to prove that 1DCSP has good detection performance in lightweight binary intrusion detection, this paper selects to conduct a comparison experiment of multiple models on TON_IOT and CICIDS2018. The lightweight models compared include: MLP (Li *et al.,* 2022), 1DCNN(Jiménez-López *et al.,* 2025), Attention-BiTCN (E. Akpaku *et al.,* 2025), Mo-bileNet (Howard *et al.,* 2019), LSTM-ResNet (Tolba *et al.,* 2024), LNN (Zhao *et al.,* 2021) and 1DCSP

proposed in this paper. As shown in Figure 5, 1DCSP outputs eight channels. To evaluate the model structure more accurately the merits of the model structure itself, the output channels of the other comparison models were also changed to eight channels while keeping the overall framework unchanged. A special case is MLP, which consists of fully connected layers; therefore, the number of hidden layers was adjusted to ensure that the number of parameters is as similar as possible to 1DCSP. In the comparison experiment, Dropout is used before the final fully connected layer of all models. To enhance the robustness and generalization of the model, comparative

experimental results are shown in Tables 5 and 6.

Analysis of Tables 5 and 6 shows that the proposed 1DCSP model achieves good results on all four metrics across different datasets. It achieves accuracies of 99.52% and 99.53% on TON_IOT and CICIDS2018, respectively, demonstrating that 1DCSP can correctly identify most traffic. Furthermore, it achieves F1 scores of 99.44% and 99.37% on TON_IOT and CICIDS2018, respectively,

Table 5: Intrusion detection results based on TON_IOT

Intrusion Detection Model	Accuracy (%)	Precision (%)	Recall (%)	F_1 Score (%)
MLP(Li *et al.,* 2022)	97.24	97.12	97.48	97.30
1DCNN (Jiménez-López *et al.,* 2025)	98.27	98.11	98.38	98.24
Attention-BiTCN (E. Akpaku *et al.,* 2025)	97.57	97.44	97.74	97.58
MobileNet (Howard *et al.,* 2019)	99.24	99.25	99.20	99.23
LSTM-ResNet (Tolba *et al.,* 2024)	96.31	96.10	96.58	96.34
LNN (Zhao *et al.,* 2021)	95.89	95.69	95.79	95.74
1DCSP	99.52	99.41	99.57	99.44

Table 6: Intrusion detection results based on CICIDS2018

Intrusion Detection Model	Accuracy (%)	Precision (%)	Recall (%)	F_1 Score (%)
MLP (Li *et al.,* 2022)	99.57	99.21	99.53	99.36
1DCNN (Jiménez-López *et al.,* 2025)	99.18	99.23	98.78	98.99
Attention-BiTCN (E. Akpaku *et al.,* 2025)	99.41	99.33	99.12	99.22
MobileNet (Howard *et al.,* 2019)	99.37	99.18	99.43	99.30
LSTM-ResNet (Tolba *et al.,* 2024)	99.52	99.33	99.39	99.36
LNN (Zhao *et al.,* 2021)	99.15	99.13	98.10	98.61
1DCSP	99.53	99.28	99.52	99.37

indicating that 1DCSP has fewer false negatives and false positives, showing good robustness and generalization. Overall, 1DCSP exhibits better intrusion detection performance compared to other models.

Lightweight Comparison Experiment

To further demonstrate that 1DCSP has good lightweight performance and can adapt to edge device conditions in the Internet of Things environment, this paper conducts a comparison experiment on TON_IOT and CICIDS2018 datasets, measuring inference time as the duration to process one epoch of data. The results of the lightweight comparison experiment are shown in Table 7. Table 7 indicates that the 1DCSP model exhibits excellent lightweight properties across datasets, making it suitable for IoT edge device deployment. Compared with other models, 1DCSP has a significant advantage in model size. Although the inference time of 1DCSP is not the best, it is still within an acceptable range.

Table 7: Model size (kB) and inference time

Intrusion Detection Model	TON_IOT		CICIDS2018	
	Size	Time (ms)	Size	Time (ms)
MLP (Li *et al.,* 2022)	58	0.785	69	18.588
1DCNN (Jiménez-López *et al.,* 2025)	34	1.131	46	18.678
Attention-BiTCN (E. Akpaku *et al.,* 2025)	129	1.294	282	18.742
MobileNet (Howard *et al.,* 2019)	27	1.342	39	22.382
LSTM-ResNet (Tolba *et al.,* 2024)	137	2.196	290	19.913
LNN (Zhao *et al.,* 2021)	42	1.628	54	19.787
1DCSP	21	2.253	32	20.390

Multiclass Intrusion Model Comparison Experiment

To demonstrate that WDGP can improve the intrusion detection performance of lightweight multiclass intrusion detection models in the IoT environment, this paper conducts a comparison experiment on the CICIDS2018 dataset, where the oversampling methods include SMOTE (Zhang *et al.,* 2020), GAN (Freitas de Araujo-Filho *et al.,* 2021) and WDGP proposed in this paper. Before starting the comparative experiment, we first counted the minority class samples in the

CICIDS2018 dataset to facilitate oversampling of the dataset. The CICIDS2018 dataset has 11 different types of data, including 5 minority classes, namely Bot, Web, XSS, LOIC-UDP and Slowloris, and their distribution is shown in Table 8. Among them, the Bot attack occupy the server resources of the target by sending a large amount of traffic, so that the user cannot obtain the service normally. Its attack traffic characteristics are a large number of field requests sent from the same IP or several IPs in a short period of time; the Web and XSS attacks occupies the server resources of the target by sending a large amount of traffic with injection characteristics, so that the attacker can bypass the system and obtain user information. Their attack traffic characteristics are a large number of requests with malicious code or special characters in the request parameters; the LOIC-UDP attack is similar to the Bot attack, also sending a large amount of repeated useless traffic. However, the Bot is generally aimed at the application layer, while the LOIC-UDP is aimed directly at the network layer. Its attack traffic characteristics are a large number of simple and useless requests sent by random IPs as UDP packets. Slowloris attacks exhaust the resources of the target server by sending incomplete requests, i.e., low-speed traffic. Its attack traffic characteristics are slow packet sending and long connection time.

Table 8: CICIDS2018 Minority Class Sample Data Distribution

Class	Training Set	Test Set
Bot	16 690	4 102
Web	502	109
XSS	184	46
LOIC-UDP	1 391	339
Slowloris	8 791	2 199

Table 8 shows the distribution analysis of minority class samples in CICIDS2018. Table 8 shows that the distribution of CICIDS2018 data types is uneven. Compared with the training set which has 86,523 majority class samples of Benign type, the least number of XSS types in the minority class samples in Table 8 is only 0.21% of the Benign type. This will cause the model to tend to predict the majority class and ignore the minority class, thus affecting the overall prediction effect of the model. Multiclass intrusion detection experiments using 1DCSP were conducted on the normal dataset, SMOTE (Zhang *et al.,* 2020) oversampled dataset, GAN (Freitas de Araujo-Filho *et al.,* 2021) oversampled dataset and WDGP oversampled dataset respectively. The experimental results are shown in Tables 9 to 12.

Table 9 shows the results of 1DCSP Multiclass intrusion

Table 9: 1DCSP Multiclass Intrusion Detection Results

Oversampling Method	Evaluation Metrics %			
	Accuracy	Precision	Recall	F_1 Score
Normal	93.52	94.43	85.00	89.47
SMOTE(Zhang *et al.,* 2020)	99.08	87.15	95.69	91.22
GAN (Freitas de Araujo-Filho *et al.,* 2021)	99.44	99.06	91.74	95.20
WDGP	99.41	99.54	91.68	95.75

Table 10: 1DCSP multiclass minority class sample intrusion detection precision

Oversampling Method	Class				
	Bot	Web	XSS	LOIC-UDP	Slowloris
Normal	100.00	89.87	100.00	100.00	99.13
SMOTE(Zhang *et al.,* 2020)	95.11	29.51	2.12	85.13	95.81
GAN (Freitas de Araujo-Filho *et al.,* 2021)	97.78	88.42	70.73	100.00	97.33
WDGP	98.65	98.68	63.79	100.00	98.96

detection under three oversampling methods. Except for the lower intrusion detection precision on the SMOTE oversampled dataset compared to the un-oversampled normal dataset, all other intrusion detection evaluation metrics on the normal dataset are lower than those on the other three oversampled datasets. This indicates that all three oversampling methods can effectively improve the model's intrusion detection performance in Multiclass scenarios. The intrusion detection accuracy under the three oversampling methods is not significantly different; this is due to data imbalance. Since the majority class data is much larger than the minority class data in the

Table 11: 1DCSP multiclass minority class sample intrusion detection recall

Oversampling Method	Class				
	Bot	**Web**	**XSS**	**LOIC-UDP**	**Slowloris**
Normal	49.14	65.13	50.00	99.12	98.95
SMOTE(Zhang *et al.,* 2020)	48.82	77.98	91.30	99.71	100.00
GAN (Freitas de Araujo-Filho *et al.,* 2021)	94.90	77.06	63.04	100.00	99.82
WDGP	99.98	68.81	84.09	100.00	99.59

Table 12: 1DCSP Multiclass Minority Class Sample Intrusion Detection F_1 Score

Oversampling Method	Class				
	Bot	**Web**	**XSS**	**LOIC-UDP**	**Slowloris**
Normal	65.89	75.52	66.67	99.56	99.03
SMOTE(Zhang *et al.,* 2020)	64.52	42.81	4.14	91.46	97.86
GAN (Freitas de Araujo-Filho *et al.,* 2021)	96.31	82.35	66.66	100.00	98.55
WDGP	99.31	81.08	72.55	100.00	99.28

test set, this cannot be used as a reference. SMOTE, because it generates samples by interpolating between the two nearest samples, the generated minority class samples are mixed with normal samples, causing the model to learn incorrect features and mistakenly identify features that do not belong to the minority class as minority class features. This makes it prone to false positives but not false negatives, resulting in the lowest precision but the highest recall. GANs, because they do not use convolution operations in the generator and discriminator, generate slightly inferior sample features compared to WDGP, and their accuracy, precision, and F1 score are also slightly lower than WDGP. WDGP has the highest accuracy, precision, and F1 score, proving that its Multiclass intrusion detection performance on the oversampled dataset is the best among the three oversampling methods. Tables 10-12 show the intrusion detection results of minority class samples in multiclass classification using 1DCSP under the three oversampling methods.

Although LOIC-UDP and Slowloris are minority classes, their distinct features—such as LOIC-UDP's simple, repetitive UDP packets and Slowloris's long connection times—make them easier to detect. Slowloris, with its long connection time and slow packet transmission per unit time, shows little difference in precision, recall, and F1 score compared to other methods besides SMOTE, so these two types cannot be used as a reference. Bot traffic is characterized by repetitive traffic within a short period. Analysis reveals that it alternates between two traffic streams: one resembling a user action and the other representing an invalid action. In normal datasets and SMOTE oversampled datasets, the repetitive pseudo-normal traffic, lacking obvious characteristics, is easily mistaken for normal traffic, leading to missed detections. However, in GAN and WDGP oversampled datasets, the generator-generated traffic surrounds the pseudo-normal traffic, helping the model better identify Bot-type traffic. Web and XSS traffic are characterized by large volumes and request parameters containing special characters. Analysis shows that Web and XSS are similar to Bot, except that Web alternates between three traffic streams: two invalid and one valid, while XSS also alternates between two streams.

Therefore, like Bot, the model's performance on Web and XSS is similar. The combined intrusion detection performance of GAN and WDGP oversampled datasets is better than that of normal datasets and SMOTE oversampled datasets. A comprehensive analysis of Tables 9-12 shows that the WDGP oversampling method exhibits good oversampling performance in both 1DCSP multiclass intrusion detection and multiclass minority class intrusion detection experiments, effectively improving the performance of lightweight intrusion detection models in multiclass intrusion detection within the IoT environment.

CONCLUSION

Addressing the limitations of edge node resources, low computational power, and generally poor model performance in IoT environments, this paper proposes an intrusion detection model based on WDGP and 1DCSP, WDGP-1DCSP. This model first combines the advantages of WGAN-GP and DCGAN, significantly improving the quality of the generated data and resolving the data imbalance problem through the integration of the loss function and gradient penalty mechanism of WGAN-GP with the network structure of DCGAN, thus expanding the dataset for training subsequent lightweight multiclass intrusion detection models. Then, the model employs CSP on top of 1DCNN. The proposed framework, through a cross-stage local feature fusion method, improves the model's detection accuracy while reducing model complexity and computational cost. Finally, comparative experiments on multiple datasets demonstrate that the proposed model, WDGP-1DCSP, can achieve efficient intrusion detection with limited computational and storage resources, offering valuable

insights for future IoT intrusion detection research.

Although the proposed model WDGP-1DCSP achieves high accuracy on the given dataset, several aspects remain for improvement. Future research will examine the implementation of this model in a real-world environment for testing and introducing more lightweight techniques to further reduce the model's computational cost and improve its generalization and intrusion detection capabilities.

REFERENCES

Akpaku, E., Chen, J., Ahmed, M., Sosu, R., Agbenyegah, F. K., & Louis, D. K. (2025). eBiTCN: Efficient bidirectional temporal convolution network for encrypted malicious network traffic detection. *Journal of Computer Security, 33*. https://doi.org/10.1177/0926227X251326282

Akpaku, E., Chen, J., Ahmed, M., Sosu, R. N. A., Agbenyegah, F. K., & Louis, D. K. (2025). eBiTCN: Efficient bidirectional temporal convolution network for encrypted malicious network traffic detection. *Journal of Computer Security*. https://doi.org/10.1177/0926227X251326282

Arjovsky, M., Chintala, S., & Bottou, L. (2017). *Wasserstein generative adversarial networks*. Sydney, NSW, Australia.

Booij, T. M., Chiscop, I., Meeuwissen, E., Moustafa, N., & den Hartog, F. T. H. (2022). ToN_IoT: The role of heterogeneity and the need for standardization of features and attack types in IoT network intrusion data sets. *IEEE Internet of Things Journal, 9*(1), 485-496. https://doi.org/10.1109/JIOT.2021.3082844

Chen, Z., Liu, J., Shen, Y., Simsek, M., Kantarci, B., Mouftah, H. T., & Djukic, P. (2022). Machine learning-enabled iot security: Open issues and challenges under advanced persistent threats. *ACM Computing Surveys, 55*(5), 1-35. https://doi.org/10.1145/3530812

Freitas de Araujo-Filho, P., Kaddoum, G., Campelo, D. R., Gondim Santos, A., Macedo, D., & Zerguine, A. (2021). Intrusion detection for cyber-physical systems using generative adversarial networks in fog environment. *IEEE Internet of Things Journal, 8*(8), 6247-6256. https://doi.org/10.1109/JIOT.2020.3028439

Goodfellow, I., Pouget-Abadie, J., & Mirza, M. (2014). *Generative adversarial nets*. Montreal, QC, Canada.

Gulrajani, I., Ahmed, F., & Arjovsky, M. (2017). *Improved training of Wasserstein GANs*. Long Beach, CA, USA.

Hadem, P., Saikia, D. K., & Moulik, S. (2021). An SDN-based intrusion detection system using SVM with selective logging for IP traceback. *Computer Networks, 191*, 108015. https://doi.org/10.1016/j.comnet.2021.108015

Hamidou, S. T., & Mehdi, A. (2025). Enhancing IDS performance through a comparative analysis of Random Forest, XGBoost, and Deep Neural Networks. *Machine Learning with Applications, 22*, 100738. https://doi.org/10.1016/j.mlwa.2025.100738

Hnamte, V., Hussain, J., & Saikia, L. C. (2025). A lightweight intrusion detection system using deep convolutional neural network. *Computers & Electrical Engineering, 120*, 109662. https://doi.org/10.1016/j.compeleceng.2025.109662

Howard, A., Sandler, M., & Chen, B. (2019). *Searching for MobileNetV3*. Seoul, South Korea.

Jiménez-López, D., Rodríguez-Barroso, N., Luzón, M. V., Del Ser, J., & Herrera, F. (2025). Membership inference attacks fueled by few-shot learning to detect privacy leakage and address data integrity. *Machine Learning and Knowledge Extraction, 7*(2), 43. https://doi.org/10.3390/make7020043

Khan, I. A., Moustafa, N., Pi, D., Sallam, K. M., Zomaya, A. A., & Li, Y. (2022). A new explainable deep learning framework for cyber threat discovery in industrial IoT networks. *IEEE Internet of Things Journal, 9*(13), 11604-11613. https://doi.org/10.1109/JIOT.2021.3124956

Kilincer, I. F., Ertam, F., & Sengur, A. (2021). Machine learning methods for cyber security intrusion detection: Datasets and comparative study. *Computer Networks, 188*, 107840. https://doi.org/10.1016/j.comnet.2021.107840

Li, Y. X., Zuo, Y., & Song, H. B. (2022). Deep learning in security of internet of things. *IEEE Internet of Things Journal, 9*(22), 22133-22146.

Misrak, S. F., & Ayele, T. W. (2025). Lightweight intrusion detection system for IoT with improved feature engineering and advanced dynamic quantization. *Discover Internet of Things, 5*(1), 1-25. https://doi.org/10.1007/s43926-025-00203-8

Muhammad, Z. (2026). Zero Trust Architecture (ZTA) Design and Implementation, A Comprehensive Review. *American Journal of Innovation in Science and Engineering, 5*, 18-25. https://doi.org/10.54536/ajise.v5i1.5905

Radford, A., Metz, L., & Chintala, S. (2016). *Unsupervised representation learning with deep convolutional generative adversarial networks*. San Juan, Puerto Rico.

Sarhan, M., Layeghy, S., Moustafa, N., & Gallagher, M. (2024). Feature extraction for machine learning-based intrusion detection in IoT networks. *Digital Communications and Networks, 10*(2), 456-467. https://doi.org/10.1016/j.dcan.2022.07.003

Tolba, A., Nabil, N., & Sallam, K. (2024). Hybrid Deep Learning-Based Model for Intrusion Detection. *Artificial Intelligence in Cybersecurity,* (1) 2024. https://doi.org/10.61356/j.aics.2024.1198

Wang, C. Y., Liao, H. Y. M., & Wu, Y. H. (2020). *CSPNet: A new backbone that can enhance learning capability of CNN*. Seattle, WA, USA.

Wisanwanichthan, T., Thammarat, S., & Netisopakul, P. (2025). A Lightweight Intrusion Detection System for IoT and UAV Using Deep Neural Networks with Knowledge Distillation. *Computers, 14*(7), 291. https://doi.org/10.3390/computers14070291

Xiao, Y., Xing, C., Zhang, T., & Zhao, Z. (2019). An intrusion detection model based on feature reduction and convolutional neural networks. *IEEE Access, 7*, 42210-42219. https://doi.org/10.1109/ACCESS.2019.2904620

Zhang, H., Huang, L., Wu, C. Q., & Li, Z. (2020). An effective convolutional neural network based on SMOTE and Gaussian mixture model for intrusion detection in imbalanced dataset. *Computer Networks, 177*, 107315. https://doi.org/10.1016/j.comnet.2020.107315

Zhao, R., Gui, G., Xue, Z., Yin, J., Ohtsuki, T., Adebisi, B., & Gacanin, H. (2021). A Novel Intrusion Detection Method Based on Lightweight Neural Network for Internet of Things. *IEEE Internet of Things Journal,* 1-1. https://doi.org/10.1109/JIOT.2021.3119055

Zhao, Z., Li, Z., Jiang, J., Yu, F., Zhang, F., Xu, C., Zhao, X., Zhang, R., & Guo, S. (2023). ERNN: Error-resilient RNN for encrypted traffic detection towards network-induced phenomena. *IEEE Transactions on Dependable and Secure Computing, 20*(1), 1-15. https://doi.org/10.1109/TDSC.2022.3223603