Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.078591

ARTICLE

# ATC-FusionNet: A Hybrid Deep Learning Ensemble for Network Intrusion Detection Systems

**Liping Wang**[1], **Jiang Wu**[1,2,*] **and Liang Wang**[3]

[1]School of Information Science and Engineering (School of Cyberspace Security), Zhejiang Sci-Tech University, Hangzhou, China
[2]Key Laboratory of Intelligent Textile and Flexible Interconnection of Zhejiang Province, Hangzhou, China
[3]Zhejiang Shannon Communication Technology Company Ltd., Hangzhou, China

*Corresponding Author: Jiang Wu. Email: wujiang@zstu.edu.cn

Received: 04 January 2026; Accepted: 13 March 2026; Published: 08 May 2026

**ABSTRACT:** The rapid growth of networked systems and the increasing diversity of cyberattack behaviors have posed significant challenges to intrusion detection, particularly in scenarios characterized by high-dimensional features and severe class imbalance. Conventional detection approaches based on handcrafted rules or shallow representations often exhibit limited robustness under such conditions. To address these issues, this paper presents a hybrid deep learning framework for network intrusion detection that integrates complementary feature learning mechanisms within a dual-branch architecture. Specifically, a Transformer branch is employed to model long-range temporal dependencies in network traffic, while a convolutional neural network branch (CNN) is used to capture localized and fine-grained feature patterns. An attention-based fusion strategy is further introduced to adaptively aggregate branch-specific representations and enhance intrusion-sensitive features. In addition, an autoencoder-based feature reconstruction module is incorporated before the dual-branch network to compress and reconstruct input features through an encoder–decoder structure, thereby preserving essential behavioral characteristics of network traffic and improving feature discriminability. To mitigate the impact of class imbalance, a Dynamic Weighted Logit-adjusted Focal Loss (DWLF) is introduced to reduce the bias toward majority classes during model optimization. Extensive experiments conducted on two public benchmark datasets demonstrate the effectiveness of the proposed approach. The proposed model achieves an overall accuracy of 90.01% on the UNSW-NB15 dataset and 97.82% on the NF-CSE-CIC-IDS2018 dataset. Experimental results indicate improved robustness under highly imbalanced data distributions, demonstrating stable performance across datasets with different traffic characteristics.

**KEYWORDS:** Network intrusion detection; Transformer networks; convolutional neural networks; autoencoder; class-imbalanced

## 1 Introduction

The continuous advancement of information technologies and the accelerating pace of digital transformation have resulted in large-scale, highly interconnected network environments. Along with this evolution, cyber attacks have become increasingly diverse, concealed, and automated, posing persistent threats to modern networked systems [1]. Various security threats, including distributed denial-of-service (DDoS) attacks [2], privilege escalation, and multi-stage advanced persistent threats (APTs) [3], have become more frequent and destructive, creating serious risks for critical infrastructure and enterprise network security. In this context, timely and accurate identification of abnormal network traffic [4] has become a key research direction for safeguarding network security.

Copyright © 2026 The Authors. Published by Tech Science Press. This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Network intrusion detection systems (NIDS) serve as essential security solutions that continuously analyze network environments to identify and monitor malicious behaviors [5]. Their goal is to detect abnormal activities or potential attacks from massive communication flows. Existing NIDS approaches can generally be divided into signature-based methods and anomaly-based methods. Signature-based methods identify known attacks using predefined attack signature databases, providing low false positive rates and high real-time efficiency [6]. However, they struggle to detect previously unseen or variant attacks. In contrast, anomaly-based approaches learn normal behavior patterns through statistical or machine learning models and detect deviations as potential intrusions [7]. Although these methods can identify unknown attacks, they often suffer from higher false positive rates.

Earlier intrusion detection studies were largely dominated by traditional machine learning techniques [8], where algorithms such as decision trees, support vector machines (SVM), and random forests were widely adopted. These methods offer advantages in terms of simplicity and interpretability but exhibit limitations when dealing with high-dimensional traffic features, complex attack behaviors, and severe class imbalance. With the rapid development of deep learning, neural network-based approaches have become increasingly popular for intrusion detection tasks. Convolutional neural networks (CNNs) [3] can automatically capture local structural patterns in traffic feature representations, while recurrent neural networks (RNNs) [9] and their variants [10] are effective for modeling sequential dependencies and temporal evolution in traffic data. More recently, Transformers [11], originally developed for natural language processing, have been introduced into cybersecurity research due to their powerful attention mechanisms [12] that enable long-range dependency modeling.

To address these challenges, this study proposes a principle-driven dual-branch intrusion detection framework termed ATC-FusionNet. The key idea is to explicitly decouple heterogeneous traffic representation learning into two complementary components: a Transformer branch dedicated to modeling long-range temporal dependencies and global traffic semantics, and a CNN branch responsible for capturing local structural patterns and short-term behavioral variations. By combining these two specialized learning pathways, the proposed architecture can more effectively model complex network traffic patterns.

The primary contributions of this work can be summarized as follows:

- **Principle-driven dual-branch intrusion detection architecture.** We design a dual-branch learning framework that explicitly separates global temporal dependency modeling and local structural pattern extraction through Transformer and CNN branches, respectively. This design enables complementary feature representation learning and improves the model's capability to capture heterogeneous network traffic patterns.
- **Autoencoder-assisted representation enhancement.** An autoencoder-based feature reconstruction module is introduced before the dual-branch network to compress redundant information and enhance latent feature representations. This mechanism improves feature compactness and strengthens the representation of minority attack categories.
- **Dynamic Weighted Logit-adjusted Focal Loss (DWLF).** To address the severe class imbalance problem in intrusion detection datasets, we propose an imbalance-aware loss function that combines logit-level prior correction with focal modulation. The proposed DWLF effectively mitigates prior bias and gradient domination during training, leading to more stable optimization and improved detection performance for minority attack classes.

## 2 Related Work

In recent years, network intrusion detection systems (NIDS) have seen growing adoption of machine learning and deep learning techniques, achieving notable performance improvements. Existing studies can be broadly categorized according to their modeling strategies and feature learning paradigms.

### *2.1 Traditional Machine Learning Approaches*

Early intrusion detection approaches mainly relied on traditional machine learning algorithms. Gu and Lu [13] introduced an IDS utilizing both support vector machines and Naive Bayes classifiers, where Naive Bayes was employed to modify original features and generate superior-quality datasets for training the SVM. This hybrid strategy achieved improved detection rates and reduced false alarms, demonstrating the effectiveness of probabilistic feature modeling. To further reduce false positives and false negatives in IDS, Aljawarneh et al. [14] presented a hybrid approach combining information gain-based feature selection with ensemble voting mechanisms, integrating classifiers such as Naive Bayes, AdaBoostM1, REPTree, Random Tree, and J48.Testing with the NSL-KDD dataset showed higher detection accuracy alongside fewer false positives.

### *2.2 Hybrid and Deep Learning Approaches*

As deep learning techniques continue to evolve, hybrid architectures have gained widespread attention in NIDS research. Yang et al. [15] employed DCGAN to generate synthetic samples for data augmentation and applied an improved LSTM classifier for intrusion detection, demonstrating the importance of dataset expansion. However, GAN-based methods generally require substantial computational resources and lack interpretability.

Rajesh Kanna and Santhi [16] introduced a hybrid approach for intrusion detection, merging HMLSTM with OCNN. Spatial features were learned via CNN with hyperparameters optimized using Lion Swarm Optimization (LSO), while temporal dependencies were captured by HMLSTM. Wei et al. [17] further introduced a CNN-BiLSTM-Attention framework, where spatial and temporal features were jointly modeled and weighted using an attention mechanism. To address class imbalance, Equalization Loss v2 was adopted, allowing the model to better detect infrequent attack categories.

### *2.3 Autoencoder-Based Feature Learning and Imbalanced Data Handling*

To address high-dimensionality and class imbalance, autoencoder-based models have been increasingly adopted. Liu et al. [18] developed an IDS which utilizes ADASYN oversampling in conjunction with LightGBM. By increasing minority class samples and leveraging ensemble learning, the model achieved improved detection performance while maintaining low computational complexity.

Li et al. [19] developed a lightweight online autoencoder-based IDS (AE-IDS), which was trained using only normal traffic data. The framework employed random forests for feature selection and affinity propagation for feature grouping, enhancing the representational capability of autoencoders. Autoencoders were employed alongside clustering methods, such as K-means and GMM, for anomaly detection.

## 3 Proposed Method

This section presents the proposed ATC-FusionNet framework for network traffic classification. The model is motivated by two major challenges in modern intrusion detection: (i) heterogeneous traffic patterns that exhibit both long-range temporal dependencies and short-term local variations, and (ii) severe class imbalance in real-world network datasets. To address these issues, ATC-FusionNet adopts a dual-branch

architecture that decouples representation learning into complementary pathways, enabling more effective modeling of complex traffic behaviors.

### *3.1 Model Overview and Structure*

Fig. 1 illustrates the overall architecture of the proposed model, which integrates an autoencoder with a dual-branch fusion framework.

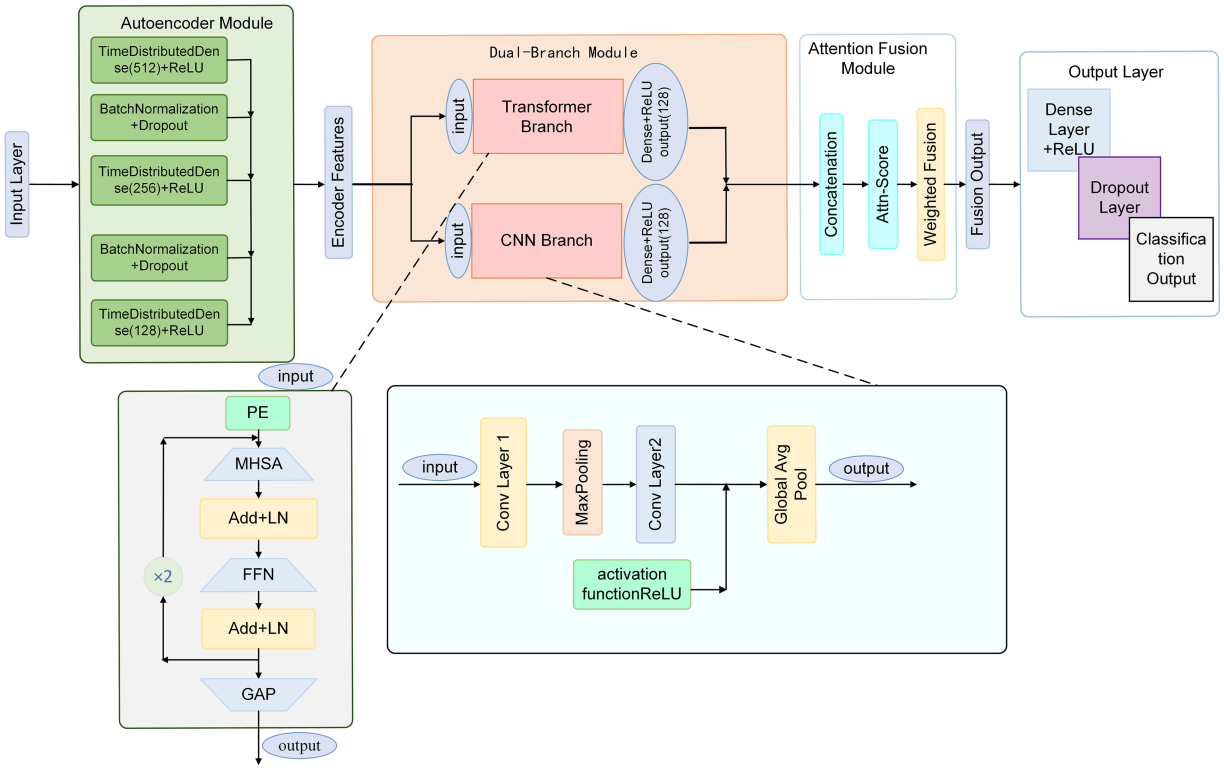



**Figure 1:** Overview of the proposed ATC-FusionNet architecture, including autoencoder-based representation learning, dual-branch feature extraction, and attention-based fusion.

Raw network traffic is first preprocessed into fixed-length time-window sequences. The autoencoder performs nonlinear dimensionality reduction and feature reconstruction to produce stable low-dimensional embeddings. These encoded features are then fed into two parallel branches: a Transformer branch that captures global contextual dependencies and a CNN branch that extracts local spatial patterns. An attention-based fusion module dynamically learns the contribution of each branch and integrates their representations for final classification. Additionally, the framework employs a Dynamic Weighted Logit-adjusted Focal Loss (DWLF) to mitigate the impact of severe class imbalance during training.

### *3.2 Autoencoder-Based Feature Representation*

An autoencoder (AE) [20] is employed as a pre-training module to extract compact and informative representations from network traffic features. An autoencoder consists of an encoder and a decoder, which project high-dimensional inputs into a lower-dimensional latent space [21]. The overall architecture of the autoencoder used in this study is illustrated in Fig. 2.

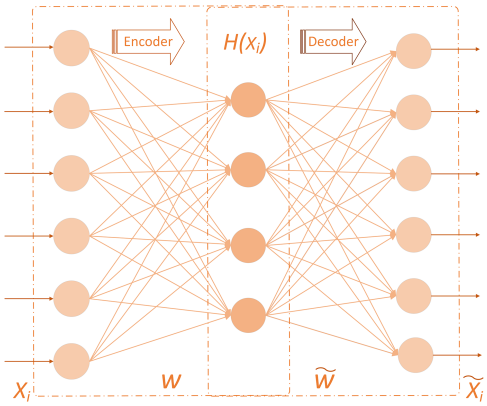



**Figure 2:** Autoencoder framework.

Let the input sample be $\mathbf{x} \in \mathbb{R}^d$, where $d$ is the dimensionality of the feature vector. The encoder maps $\mathbf{x}$ to a latent vector:

$$\mathbf{h} = f(\mathbf{W}_e\mathbf{x} + \mathbf{b}_e), \tag{1}$$

where $\mathbf{h} \in \mathbb{R}^k$ with $k \ll d$, $\mathbf{W}_e$ and $\mathbf{b}_e$ are the encoder parameters, and $f(\cdot)$ denotes a nonlinear activation function. The decoder reconstructs the original input from the latent vector:

$$\hat{\mathrm{x}} = g(\mathbf{W}_d\mathbf{h} + \mathbf{b}_d), \tag{2}$$

where $\mathbf{W}_d$ and $\mathbf{b}_d$ are the decoder parameters, $g(\cdot)$ is a nonlinear activation function, and $\hat{\mathrm{x}}$ is the reconstructed output. The autoencoder is optimized by minimizing the mean squared error (MSE) between the input and reconstruction:

$$\mathcal{L}_{\mathrm{MSE}} = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{x}_i - \hat{\mathrm{x}}_i\|^2, \tag{3}$$

where $\mathbf{x}_i$ and $\hat{\mathrm{x}}_i$ denote the $i$-th input and reconstructed sample, and $N$ is the number of training samples. In our implementation, the input to the autoencoder is a windowed sequence of shape $8 \times 264$. The encoder consists of three TimeDistributed dense layers with dimensions 512, 256, and 128. Each layer is followed by batch normalization and a dropout layer (rate = 0.2). The decoder symmetrically consists of two TimeDistributed dense layers (sizes 256 and 512) and a final linear reconstruction layer. The encoder's output, a 128-dimensional vector per time step, serves as the compact latent representation fed into the dual-branch network.

### *3.3 CNN Branch Design*

The CNN branch [22] is designed to capture local temporal dependencies and short-term dynamic patterns from the latent feature sequences generated by the autoencoder. Compared with attention-based models that focus on global relationships, convolutional operations are more effective in modeling localized feature [23] interactions within short temporal windows. Therefore, the CNN branch complements the Transformer branch by providing fine-grained structural pattern extraction.

$$X \in \mathbb{R}^{T\times D}, \tag{4}$$

where $T$ indicates the time window length and $D$ refers to the dimensionality of the latent features.

The CNN branch adopts a hierarchical one-dimensional convolutional architecture [24]. The first convolutional layer consists of $C_1 = 64$ one-dimensional convolutional filters with kernel width $K_1 = 3$. For an input sequence $X$, the activation of the $j$-th filter at time step $t$ is computed as

$$H_j^1(t) = \mathrm{ReLU}\left(\sum_{i=1}^{D}\sum_{k=0}^{K_1-1} W_{j,i}^1(k)\, X(t+k,i) + b_j^1\right), \tag{5}$$

where $W_{j,i}^1$ and $b_j^1$ represent the kernel weights and bias associated with the $j$-th convolutional filter, respectively. This layer performs local receptive-field modeling along the temporal dimension and produces the feature map

$$H^1 \in \mathbb{R}^{T\times C_1}, \tag{6}$$

which encodes basic local interactions and short-term temporal patterns among adjacent time steps.

Subsequently, a one-dimensional max-pooling layer is applied to $H^1$ with a pooling window size and stride of 2, defined as

$$p_j^1(t) = \max_{k\in\{0,1\}} H_j^1(2t+k), \tag{7}$$

where $p_j^1(t)$ denotes the pooled output of the $j$-th feature channel at time index $t$. This operation reduces the temporal resolution by half, yielding

$$P^1 \in \mathbb{R}^{(T/2)\times C_1}. \tag{8}$$

The pooled features $P^1$ are subsequently passed to the second convolutional layer, which contains $C_2 = 128$ one-dimensional filters. This layer extracts higher-level semantic representations from the input sequence, resulting in the feature map

$$H^2 \in \mathbb{R}^{(T/2)\times C_2}. \tag{9}$$

Finally, a global average pooling layer aggregates the temporal feature map into a fixed-length vector

$$h_{\mathrm{cnn}} \in \mathbb{R}^{C_2}, \tag{10}$$

defined as

$$h_{\mathrm{cnn}}(j) = \frac{1}{T/2}\sum_{t=1}^{T/2} H_j^2(t). \tag{11}$$

This operation summarizes the temporal information of each feature channel and produces the final output of the CNN branch, which is subsequently used for feature fusion with the Transformer branch, as illustrated in Fig. 3.

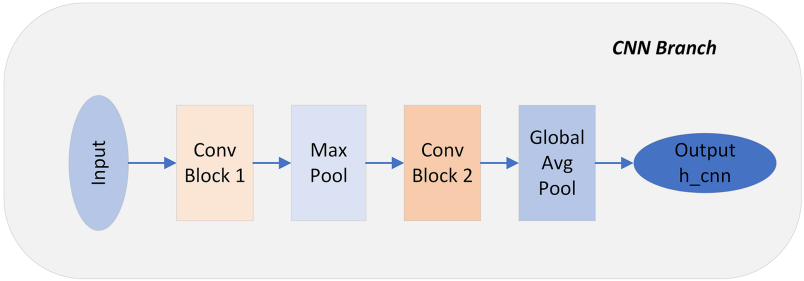



**Figure 3:** CNN branch Architecture.

### *3.4 Transformer Branch Design*

The Transformer [25] branch aims to learn long-range dependencies and global contextual patterns from the input feature sequence. Similar to the CNN branch, its input is the latent feature sequence produced by the autoencoder.

Since the Transformer architecture itself is permutation-invariant [26], explicit positional information is injected into the input sequence to preserve temporal order. In this work, a fixed sinusoidal positional encoding scheme is adopted. The positional encoding matrix is defined as

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right), \tag{12}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right), \tag{13}$$

here, $pos$ represents the absolute position of each time step in the sequence, $i$ denotes the feature dimension index, and $d_{\text{model}}$ defines the size of the embedding space. By incorporating positional encodings, the Transformer is enabled to distinguish temporal relationships among traffic features.

The position-enhanced feature sequence is then input to a deep Transformer encoder consisting of $N = 2$ identical stacked encoder layers. Each encoder layer uses $h = 2$ attention heads, and the model dimension $d_{\text{model}} = 128$. An encoder layer is structured around three main subcomponents, namely multi-head self-attention, residual connections combined with layer normalization, and a position-wise feed-forward network.

Within each encoder layer, the input sequence is initially transformed through linear projections to generate the query ($Q$), key ($K$), and value ($V$) matrices. For an individual attention head, the scaled dot-product attention is defined as

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{14}$$

where $d_k$ refers to the dimension of the key representations. To capture diverse semantic relationships across different representation subspaces, the self-attention is further implemented in a multi-head manner. Specifically, the output of the $i$-th attention head is given by

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V), \quad i = 1, \ldots, h, \tag{15}$$

and the final multi-head attention output is obtained by

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O, \tag{16}$$

here, $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, and $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ correspond to the linear projection matrices for the $i$-th attention head, and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$ is used to project the concatenated multi-head outputs. In practice, $d_k = d_v = d_{\text{model}}/h$ is adopted. Multi-head attention enables the model to capture contextual relationships from multiple representation subspaces.

Residual connections and layer normalization are applied after each sub-layer to improve training stability.

A position-wise feed-forward network is included in each encoder layer to independently transform the representation at every time step. The FFN contains two linear transformations with an intermediate ReLU activation:

$$\text{FFN}(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2, \tag{17}$$

$W_1$, $b_1$ and $W_2$, $b_2$ correspond to the weights and biases of the two linear layers. The initial layer expands the feature dimension from $d_{\text{model}} = 128$ to a higher dimension, while the subsequent layer restores it to the original embedding size.

Finally, a global average pooling layer is employed along the temporal dimension to aggregate the Transformer encoder outputs into a fixed-length feature vector,

$$h_{\text{tr}} \in \mathbb{R}^{d_{\text{model}}}, \tag{18}$$

defined as

$$h_{\text{tr}}(j) = \frac{1}{T}\sum_{t=1}^{T} Z_j(t), \tag{19}$$

here, $Z(t) \in \mathbb{R}^{d_{\text{model}}}$ denotes the Transformer encoder output at time step $t$. This output serves as the final representation of the Transformer branch and is later combined with the CNN branch.

### *3.5 Attention-Based Feature Fusion Layer*

Different from prior hybrid architectures that concatenate or statically fuse CNN and Transformer outputs, our fusion strategy is motivated by the observation that local flow-level patterns and global temporal dependencies contribute unevenly across attack categories, especially under extreme class imbalance. Therefore, a fixed fusion scheme may amplify dominant-class representations while suppressing minority-class cues.

To address this issue, we adopt an attention-based dynamic fusion mechanism that adaptively adjusts the contribution of each branch conditioned on the learned feature context, enabling the model to emphasize discriminative cues for minority attacks when necessary.

Let the output of the Transformer branch be denoted as the global feature vector $\mathbf{h}_{\text{tr}} \in \mathbb{R}^{128}$, and the local features extracted by the CNN branch are represented as $\mathbf{h}_{\text{cnn}} \in \mathbb{R}^{128}$. A fused candidate representation is formed by concatenating the two feature vectors:

$$\mathbf{h}_f = [\mathbf{h}_{\text{tr}}; \mathbf{h}_{\text{cnn}}] \in \mathbb{R}^{256}, \tag{20}$$

where [, ; ,] indicates feature concatenation. The concatenated feature vector $\mathbf{h}_f$ (256-dimensional) is passed through a dense layer with 64 units and ReLU activation. A subsequent dense layer with a single unit and sigmoid activation produces the scalar attention weight $\alpha$.

$$\alpha = \sigma\left(\mathbf{W}_2\,\mathrm{ReLU}\left(\mathbf{W}_1\mathbf{h}_f + \mathbf{b}_1\right) + \mathbf{b}_2\right), \tag{21}$$

where $\alpha \in (0,1)$ represents the attention weight assigned to the Transformer branch, $\sigma(\cdot)$ denotes the Sigmoid function, and $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{b}_1$, and $\mathbf{b}_2$ are trainable parameters. Based on the learned attention weight, the final fused representation is obtained through a linear weighted combination of the two branch features:

$$\mathbf{h}_{\text{fused}} = \alpha \cdot \mathbf{h}_{\text{tr}} + (1-\alpha) \cdot \mathbf{h}_{\text{cnn}}. \tag{22}$$

Here, $\alpha$ is a scalar attention coefficient applied uniformly across feature dimensions, enabling sample-wise adaptive branch weighting. The learned coefficient $\alpha$ controls the relative contribution of the two branches for each sample. To improve the separability of the fused features, a dense projection is introduced and regularized using Dropout, as formulated below:

$$\mathbf{z} = \mathrm{Dropout}\left(\mathrm{ReLU}\left(\mathbf{W}_f\mathbf{h}_{\text{fused}} + \mathbf{b}_f\right)\right), \tag{23}$$

where $\mathbf{W}_f$ and $\mathbf{b}_f$ represent the learnable parameters of the projection layer. The ultimate classification results are computed via a linear transformation:

$$\mathbf{y} = \mathbf{W}_o\mathbf{z} + \mathbf{b}_o, \tag{24}$$

where $\mathbf{y} \in \mathbb{R}^C$ corresponds to the unnormalized logits of $C$ intrusion categories. Training is performed by minimizing the cross-entropy loss:

$$\mathcal{L}_{\text{cls}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log\left(\mathrm{Softmax}(\mathbf{y}_i)_c\right), \tag{25}$$

with $N$ as the batch size and $C$ as the total number of classes, $y_{i,c}$ corresponds to the one-hot encoded ground-truth label for the $i$-th sample in class $c$, and $\mathrm{Softmax}(\mathbf{y}_i)_c$ gives the predicted probability for class $c$. The classification loss is jointly backpropagated through the fusion layer and both branches, enabling coordinated optimization of the Transformer and CNN components. Overall, this fusion mechanism enables adaptive integration of global and local traffic representations, improving robustness across diverse attack patterns and imbalanced traffic distributions.

### *3.6 Loss Function*

To address the severe class imbalance commonly observed in network intrusion detection systems (NIDS), a Dynamic Weighted Logit-adjusted Focal Loss (DWLF) is employed. Class imbalance in intrusion detection typically manifests through skewed class priors and the dominance of easily classified majority samples. The proposed DWLF loss is designed to decouple these effects by jointly incorporating logit-level prior correction, focal modulation for hard-sample emphasis, and dynamic class reweighting.

Let $s_{n,i}$ denote the raw logit predicted by the network for the $n$-th sample and class $i$. Logit adjustment is first applied based on the class prior distribution:

$$s_{n,i}^{\text{adj}} = s_{n,i} + \tau\log(\pi_i), \tag{26}$$

where $\pi_i$ represents the empirical prior probability of class $i$ estimated from the training dataset, and $\tau$ controls the strength of the adjustment.

The adjusted posterior probability is then obtained via the Softmax function

$$p_{n,i}^{\text{adj}} = \frac{\exp(s_{n,i}^{\text{adj}})}{\sum_{j=1}^{C} \exp(s_{n,j}^{\text{adj}})}, \tag{27}$$

where $C$ denotes the total number of intrusion categories.

To emphasize hard-to-classify samples, a focal modulation term is introduced based on the adjusted posterior probability:

$$\text{FM}_{n,i} = (1 - p_{n,i}^{\text{adj}})^{\gamma}, \tag{28}$$

where $\gamma > 0$ is the focusing parameter. Larger values of $\gamma$ increase the contribution of samples with lower predicted confidence.

Considering that class distributions may vary across mini-batches. Therefore, a final class weight is constructed by combining a static global weight with a dynamically updated batch-level weight:

$$w_i^{\text{final}} = \alpha_i^{\text{static}} \cdot \beta_i^{\text{dyn}}, \tag{29}$$

where $\alpha_i^{\text{static}}$ is derived from global class statistics, and $\beta_i^{\text{dyn}}$ reflects the batch-wise class frequency.

The dynamic weight $\beta_i^{\text{dyn}}$ is updated using an exponential moving average (EMA) scheme:

$$\beta_i^{\text{dyn}}(t) = \lambda\, \beta_i^{\text{dyn}}(t-1) + (1-\lambda)\, \hat{f}_i^{(t)}, \tag{30}$$

where $\hat{f}_i^{(t)}$ denotes the proportion of class $i$ samples in the $t$-th mini-batch, and $\lambda \in (0,1)$ is the EMA decay factor.

By integrating logit adjustment, focal modulation, and dynamic class weighting, the overall DWLF objective is formulated as

$$\mathcal{L}_{\text{DWLF}} = -\frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{C} w_i^{\text{final}}\, y_{n,i}\, (1 - p_{n,i}^{\text{adj}})^{\gamma}\, \log\left(p_{n,i}^{\text{adj}}\right), \tag{31}$$

where $N$ is the mini-batch size and $y_{n,i} \in \{0,1\}$ denotes the one-hot encoded ground-truth label of the $n$-th sample for class $i$.

This loss function is fully differentiable and can be optimized using standard stochastic gradient-based methods. During training, gradients are jointly propagated through all network components, enabling the proposed model to learn discriminative representations under dynamically changing and highly imbalanced traffic distributions. Overall, DWLF enables stable optimization under highly skewed class distributions, which is critical for real-world intrusion detection scenarios.

## 4 Experiments and Discussion

This section reports the experimental results and discussion of the proposed intrusion detection model,evaluated on two datasets: UNSW-NB15 and NF-CSE-CIC-IDS2018.

### *4.1 Experimental Setup*

Experiments are conducted on Windows 11 using Python 3.8 and TensorFlow 2.5 (Conda), with an NVIDIA RTX 3090 Ti GPU, Intel Core i7 CPU, and 32 GB RAM. To guarantee a fair comparison, all models

are trained and tested under identical experimental settings. The Adam optimizer is used with a learning rate of 0.001, and the batch size is fixed at 128. The temporal window size is set to $T = 8$, which provides a good trade-off between capturing short-term temporal dependencies in network flows and maintaining computational efficiency, consistent with common practice in flow-based intrusion detection literature. The Transformer module adopts a lightweight configuration with $N = 2$ encoder layers and $h = 2$ attention heads. The hidden dimension is fixed to $d_{\text{model}} = 128$ to match the latent representation of the autoencoder. For the proposed Dynamic Weighted Logit-adjusted Focal Loss (DWFL), the logit adjustment temperature is set to $\tau = 0.7$ to mitigate class imbalance, the focal parameter is set to $\gamma = 2.0$ to emphasize hard samples, and the dynamic class weights are updated using an exponential moving average with decay factor $\lambda = 0.3$. The final class weights are obtained by equally mixing static and dynamic weights with $\beta = 0.5$.To further justify these settings, a brief sensitivity analysis was conducted on a validation subset of the UNSW-NB15 dataset. We varied the temporal window size $T \in \{4, 8, 16\}$ and the number of Transformer encoder layers $N \in \{1, 2, 4\}$. The configuration $T = 8$ and $N = 2$ provided the best balance between detection performance and computational efficiency, while larger configurations yielded only marginal improvements. Similar stability was observed when varying the DWLF parameters ($\tau$, $\gamma$, and $\lambda$) within reasonable ranges, indicating that the proposed framework is not overly sensitive to moderate hyperparameter variations.

### *4.2 Evaluation Metrics*

Four commonly adopted classification metrics are used: Precision, Recall, F1-score, and Accuracy. The evaluation metrics are computed as follows:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{32}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{33}$$

$$\text{F1-score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{34}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{35}$$

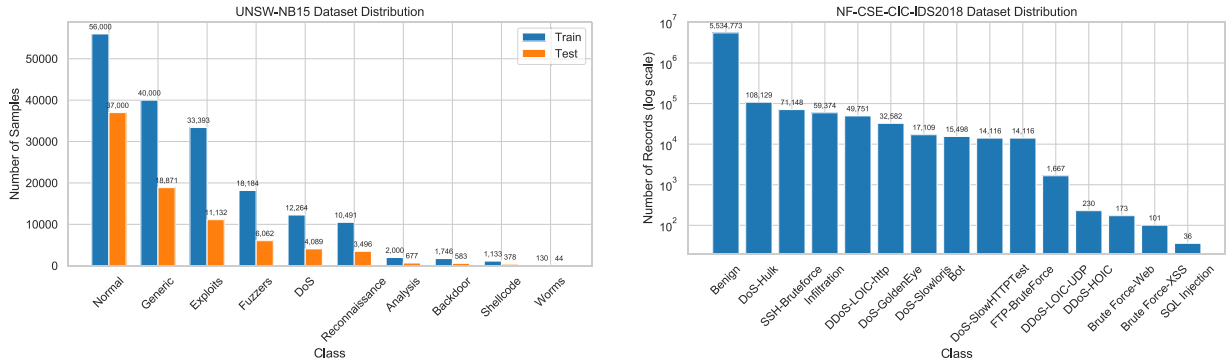
True positives (TP) are instances of the positive class that are correctly identified as positive, whereas true negatives (TN) are negative instances correctly recognized as negative. False positives (FP) occur when negative instances are mistakenly labeled as positive, and false negatives (FN) correspond to positive instances incorrectly assigned to the negative class. For multi-class classification, the metrics are computed using a macro-averaging strategy across all classes.

### *4.3 Datasets*

To assess the performance and generalization of the proposed intrusion detection model, experiments are performed on two publicly accessible benchmark datasets: UNSW-NB15 and NF-CSE-CIC-IDS2018.

#### *4.3.1 UNSW-NB15*

UNSW-NB15 [27] is a benchmark dataset designed to address limitations of earlier intrusion detection datasets. It contains both normal traffic and nine contemporary attack categories generated in a controlled laboratory environment. Each flow is described by 49 traffic features, excluding label information. The distribution of training and testing samples is shown in Fig. 4a.




**(a)** UNSW-NB15 **(b)** NF-CSE-CIC-IDS2018

**Figure 4:** Class distribution of samples in the two benchmark datasets.

#### *4.3.2 NF-CSE-CIC-IDS2018*

The NF-CSE-CIC-IDS2018 [28] dataset is a benchmark for network intrusion detection. It contains over eight million flow records collected over multiple days, including benign traffic and various attack types such as DoS/DDoS, brute-force, web-based attacks, botnet activity, and infiltration. Each flow record has more than eighty features. As shown in Fig. 4b, the class distribution is highly imbalanced, with benign traffic and large-scale attacks dominating.

### *4.4 Data Preprocessing*

Both datasets are preprocessed using a unified pipeline prior to training. First, discriminative flow-level features are extracted from raw network traffic records following the unified flow format specification. No explicit feature selection or manual dimensionality reduction is applied at this stage. Instead, all available flow-level features are retained to preserve complete traffic semantics.This design choice avoids introducing manual feature-selection bias and allows the autoencoder to automatically learn compact latent representations from the full feature space.

A sliding window mechanism with a fixed length of eight time steps is then employed to segment continuous traffic flows into sequential samples, where each window represents a complete network behavior unit and captures short-term temporal evolution patterns.

Numerical features are normalized using Z-score standardization to ensure comparable value ranges across different feature dimensions.Categorical features are encoded using a bounded strategy, where the number of categorical levels is explicitly limited (32 levels per categorical attribute), and infrequent categories are mapped to an auxiliary token.This design prevents uncontrolled dimensionality expansion after encoding.

Missing or invalid values are handled using standard preprocessing operations to maintain data consistency. After preprocessing, the resulting flow sequences are transformed into normalized tensor representations and fed into the autoencoder module, which performs implicit feature compression and representation learning prior to downstream classification. Following preprocessing, the resulting feature dimensionality is fixed at 264 for UNSW-NB15 and 290 for NF-CSE-CIC-IDS2018, respectively.

### *4.5 Baseline*

In this study, the proposed approach is compared and analyzed against the following methods.

- **XGBoost** [29]: A comparative study by Lawal et al. evaluated multiple machine learning models on the CSE-CIC-IDS2018 dataset using undersampling to address class imbalance.
- **Support Vector Machine (SVM)** [30]: A classical supervised learning method that constructs optimal separating hyperplanes in high-dimensional feature spaces and is frequently used as a baseline in intrusion detection studies.
- **Graph Attention(NEGAT)** [31]: A self-supervised GNN-based model that exploits graph attention and contrastive learning to model structural dependencies among network flows for unsupervised intrusion detection.
- **CNN-LSTM** [32]: A hybrid deep learning framework combining convolutional neural networks for local feature extraction with LSTM layers to capture temporal dependencies in network flow data.
- **CNN-BiLSTM-Attention** [17]: An improved hybrid model utilizing bidirectional LSTM layers alongside an attention mechanism to capture temporal dependencies in both directions and highlight discriminative features.
- **FlowTransformer** [25]: A transformer-based network intrusion detection framework that models long-term dependencies in flow-level traffic and provides a modular architecture for flexible component substitution across different datasets.
- **AI-SCAN** [33]: A deep learning IDS proposed by [authors] utilizes a CNN architecture with SMOTE-based data augmentation and class weighting strategies to handle imbalanced data.

### *4.6 Performance Comparison*

Two publicly available benchmark datasets are adopted to evaluate the effectiveness of the proposed intrusion detection model: UNSW-NB15 and NF-CSE-CIC-IDS2018.

Table 1 summarizes the multi-class classification performance on the UNSW-NB15 and NF-CSE-CIC-IDS2018 datasets. In addition to classical machine learning and CNN-based baselines, FlowTransformer is included as a representative Transformer-based NIDS baseline, whose results are reproduced under the same experimental settings to ensure fair comparison.

**Table 1:** Comparison of multi-classification results (%).

| Model | UNSW-NB15 | | | NF-CSE-CIC-IDS2018 | | |
|---|---|---|---|---|---|---|
| | Acc | Macro-F1 | Recall | Acc | Macro-F1 | Recall |
| SVM | 74.80 | – | 83.71 | – | – | – |
| CNN-LSTM | 82.20 | – | 82.41 | – | – | – |
| CNN-BiLSTM-Attention | 88.83 | – | 98.51 | – | – | – |
| FlowTransformer | 90.11 | 85.21 | 98.40 | 97.05 | 74.23 | 96.23 |
| NEGAT (Graph Attention) | – | – | – | 94.79 | – | 95.79 |
| XGBoost | – | – | – | 98.00 | 80.00 | 78.00 |
| AI-SCAN | – | – | – | 97.50 | – | 95.00 |
| **ATC-FusionNet (Proposed)** | 90.01 ± 0.10 | 85.14 ± 0.08 | 98.48 ± 0.08 | 97.82 ± 0.07 | 77.20 ± 0.09 | 96.58 ± 0.09 |

Given the severe class imbalance present in both datasets, Macro-F1 is emphasized as the primary evaluation metric, since it reflects balanced performance across attack categories rather than being dominated by majority classes. On the UNSW-NB15 dataset, ATC-FusionNet achieves a Macro-F1 of 85.14% ± 0.08% and an accuracy of 90.01%, which is comparable to FlowTransformer while consistently outperforming the CNN-LSTM and CNN-BiLSTM-Attention baselines. These results indicate enhanced discrimination capability for minority attack classes without compromising overall detection performance.

On the more challenging NF-CSE-CIC-IDS2018 dataset, ATC-FusionNet achieves a Macro-F1 of 77.20% ± 0.09% and an accuracy of 97.82%, surpassing both FlowTransformer and the graph-attention-based NEGAT model. Compared with AI-SCAN, which mainly improves detection accuracy through enhanced feature engineering, the proposed ATC-FusionNet attains comparable overall accuracy while maintaining more balanced performance across different attack categories. Although the XGBoost model reports slightly higher accuracy, its lower Macro-F1 suggests that tree-based methods may still encounter difficulties in recognizing minority attack classes under severe class imbalance. By contrast, the integration of autoencoder-based representation learning, dual-branch Transformer–CNN feature extraction, and the proposed DWLF loss contributes to improved robustness for minority-class detection.

To evaluate robustness and mitigate the influence of stochastic factors, all results for ATC-FusionNet are reported as the mean and standard deviation over multiple runs with different random seeds. The consistently low variance across metrics further confirms that the observed performance gains are stable rather than arising from random fluctuations.

Overall, the results on both benchmark datasets demonstrate that ATC-FusionNet achieves stable and competitive performance compared with recent Transformer-based and deep learning NIDS approaches, highlighting the effectiveness of the proposed architecture for multi-class intrusion detection tasks.

Table 2 reports the per-class classification performance of the proposed model on the UNSW-NB15 dataset. Overall, the model achieves stable performance across most traffic categories. In particular, the *Normal* class attains an F1-score of 0.9961, demonstrating that the proposed model can accurately separate benign samples from attack instances with a low false alarm rate.

**Table 2:** Per-class classification performance on the UNSW-NB15 dataset.

| Class | Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|
| Analysis | 0.7260 | 0.7020 | 0.7260 | 0.7138 |
| Backdoor | 0.6850 | 0.6720 | 0.6850 | 0.6784 |
| Normal | 0.9991 | 0.9992 | 0.9930 | 0.9961 |
| Reconnaissance | 0.7420 | 0.7580 | 0.7955 | 0.7499 |
| Generic | 0.9420 | 0.9045 | 0.9595 | 0.9228 |
| Exploits | 0.9580 | 0.9140 | 0.9561 | 0.9354 |
| Fuzzers | 0.9410 | 0.9030 | 0.9339 | 0.9216 |
| Dos | 0.9620 | 0.8930 | 0.9712 | 0.9261 |
| Shellcode | 0.9680 | 0.8840 | 0.9525 | 0.9241 |
| Worms | 0.7580 | 0.7340 | 0.7415 | 0.7458 |
| Average | 0.9001 | 0.9040 | 0.9848 | 0.8958 |

For the majority of attack categories, including *Generic*, *Exploits*, *Fuzzers*, and *DoS*, the model consistently achieves F1-scores above 0.92. These attacks typically generate relatively stable traffic patterns with

distinctive statistical characteristics, allowing the dual-branch architecture to capture both global temporal dependencies and local flow-level variations effectively.

From an error analysis perspective, Fig. 5 provides further insight into the misclassification behavior of the proposed model on the UNSW-NB15 dataset. Most samples are concentrated along the main diagonal, indicating that ATC-FusionNet achieves stable and reliable detection performance for the majority of traffic categories. Nevertheless, misclassifications are not uniformly distributed across classes and are mainly concentrated among minority attack types.

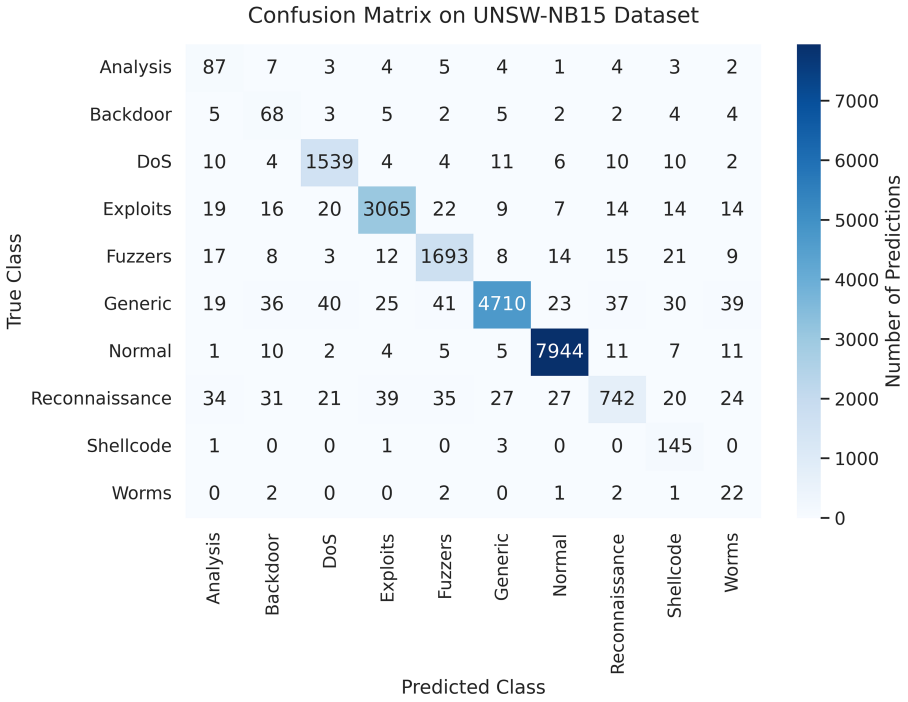



**Figure 5:** Confusion matrix (UNSW-NB15).

Specifically, samples from the *Reconnaissance* and *Worms* classes are occasionally misclassified as semantically related attack categories or benign traffic. This observation is consistent with their lower F1-scores reported in Table 2. Such failure cases are primarily caused by two factors. First, these attack categories suffer from severe sample scarcity, which limits the model's ability to learn sufficiently discriminative decision boundaries. Second, their flow-level statistical characteristics exhibit substantial overlap with other attack types, reducing feature separability at the flow representation level.

Table 3 presents the per-class performance on the NF-CSE-CIC-IDS2018 dataset. The proposed model achieves outstanding detection performance for the dominant *Benign* traffic, with an accuracy of 0.9994, as well as for several high-frequency attack categories such as *DoS-Hulk*, *DDoS-HOIC*, and *DoS-GoldenEye*. The findings show that the model captures highly distinctive features for significant and large-scale attack behaviors.

**Table 3:** Per-class classification performance on the NF-CSE-CIC-IDS2018 dataset.

| Class | Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|
| Benign | 0.9994 | 0.9965 | 0.9910 | 0.9937 |
| DDoS-LOIC-Http | 0.9633 | 0.9150 | 0.9650 | 0.9488 |
| Brute Force-Web | 0.3031 | 0.2999 | 0.3112 | 0.3226 |
| SQL Injection | 0.2125 | 0.2333 | 0.2114 | 0.2214 |
| DoS-Hulk | 0.9768 | 0.9620 | 0.9810 | 0.9714 |
| Brute Force-XSS | 0.2667 | 0.2544 | 0.2223 | 0.1999 |
| Infiltration | 0.8158 | 0.8520 | 0.8520 | 0.8470 |
| Bot | 0.6601 | 0.6850 | 0.6850 | 0.6280 |
| SSH-Bruteforce | 0.9877 | 0.9210 | 0.9790 | 0.9491 |
| DDoS-LOIC-UDP | 0.8999 | 0.8500 | 0.8900 | 0.9140 |
| FTP-BruteForce | 0.8755 | 0.8750 | 0.8667 | 0.9333 |
| DoS-SlowHTTPTest | 0.6667 | 0.5000 | 0.5000 | 0.6991 |
| DoS-GoldenEye | 0.9773 | 0.9500 | 0.9700 | 0.9641 |
| DDoS-HOIC | 0.9764 | 0.9400 | 0.9720 | 0.9557 |
| DoS-Slowloris | 0.9761 | 0.9250 | 0.9620 | 0.9621 |
| Average | 0.9782 | 0.9740 | 0.9658 | 0.9731 |

However, for extremely low-frequency attack types, including *Brute Force-Web* and *SQL Injection*, the detection performance is noticeably reduced. As illustrated by the class distribution in Fig. 4b, these categories suffer from severe sample imbalance, contain extremely few samples. Consequently, even a small number of misclassifications can lead to disproportionately low evaluation scores, a phenomenon commonly observed in highly imbalanced intrusion detection scenarios.

Fig. 6 further reveals that misclassifications are mainly concentrated among these rare attack categories. Specifically, samples from *Brute Force-Web* and *SQL Injection* are occasionally confused with other web-related attacks or benign traffic due to their short-lived, low-intensity behaviors and substantial overlap in flow-level statistical features. These attacks typically involve short-lived and low-intensity interactions that generate traffic patterns very similar to benign web activities. As a result, their flow-level statistical features show substantial overlap with normal traffic, which significantly reduces feature separability even for deep representation models.

Overall, this analysis suggests that the remaining classification errors are mainly driven by intrinsic data characteristics rather than training instability or model underfitting. While the proposed Dynamic Weighted Logit-adjusted Focal Loss alleviates class-prior bias and enhances minority class learning, accurately detecting extremely rare and behaviorally similar attack types remains challenging.

Fig. 7 presents the reconstruction loss evolution during autoencoder pretraining on the UNSW-NB15 and CIC-IDS2018 datasets.

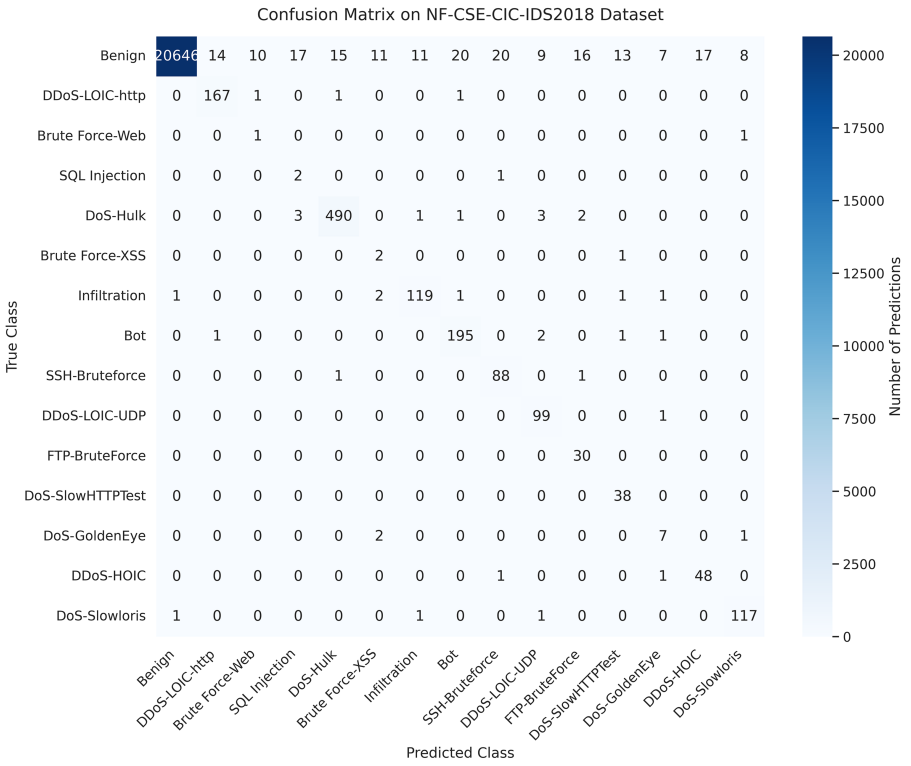
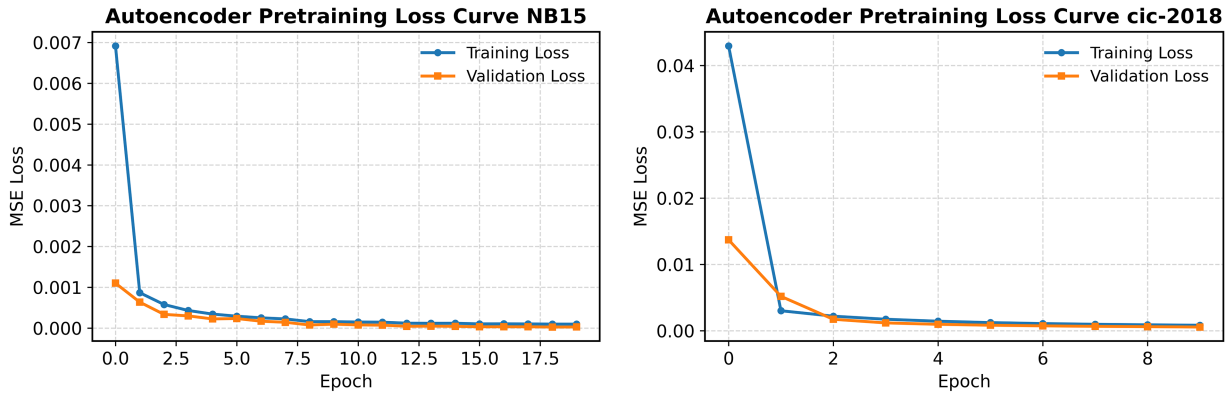



**Figure 6:** Confusion matrix (NF-CSE-CIC-IDS2018).




**(a)** UNSW-NB15 Dataset **(b)** CIC-IDS2018 Dataset

**Figure 7:** Autoencoder training loss curve.

For both datasets, the reconstruction loss decreases rapidly in the initial epochs, indicating that the autoencoder quickly captures dominant statistical regularities in network flow features. As training proceeds, the training and validation losses converge smoothly with a small gap, suggesting stable optimization and good generalization.

Different convergence behaviors are observed between the two datasets. UNSW-NB15 exhibits a relatively smooth convergence toward a low-loss plateau, reflecting more stable traffic patterns. In contrast, CIC-IDS2018 shows a higher initial loss and a steeper early decline, likely due to the greater diversity and complexity of its traffic behaviors.

Overall, these results indicate that the autoencoder effectively compresses raw flow features into a compact latent space while preserving essential information for downstream intrusion detection.

To further evaluate the effectiveness of the dual-branch architecture, the performance of single-branch models was compared with that of the dual-branch fusion model, as shown in Figs. 8 and 9.

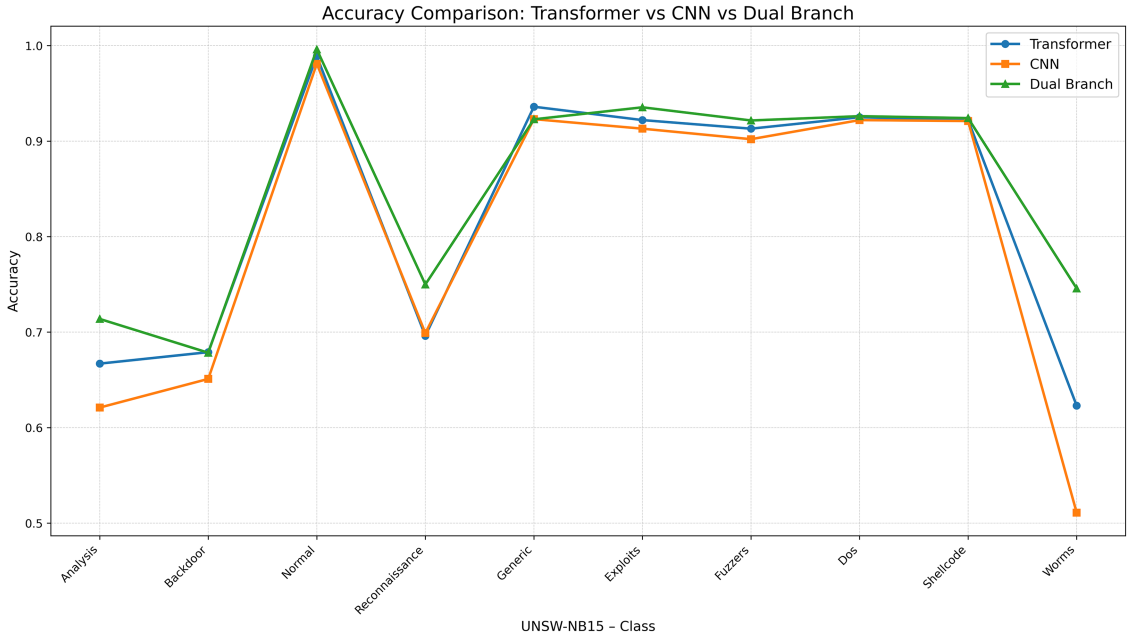



**Figure 8:** Single- and dual-branch model comparison on UNSW-NB15.

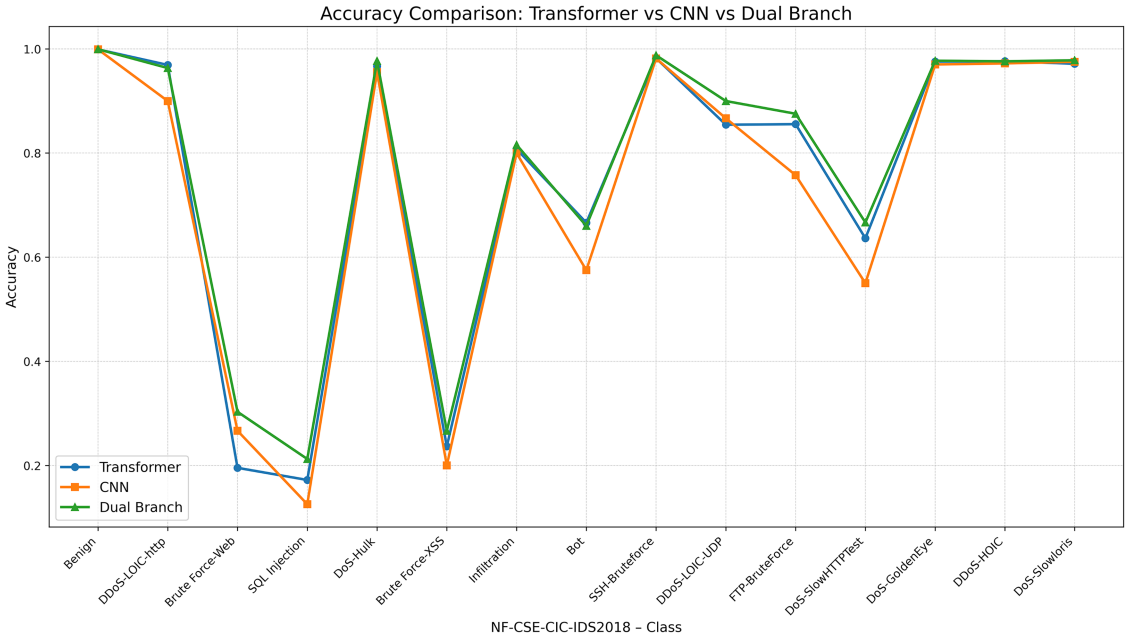



**Figure 9:** Single- and dual-branch model comparison on NF-CSE-CIC-IDS2018.

As shown in Fig. 8, the dual-branch model consistently outperforms the single-branch models across multiple attack categories in the UNSW-NB15 dataset. This improvement indicates that combining global contextual modeling and local feature extraction provides complementary information for network traffic representation.

Notably, for the *Worms* category, the dual-branch model achieves improvements of approximately 0.12 and 0.23 over the Transformer and CNN models, respectively. This result highlights the advantage of feature fusion when dealing with extremely low-sample attack scenarios, where relying on a single representation type may lead to unstable decision boundaries.

In contrast, the CNN-only model exhibits relatively lower performance and larger fluctuations across several categories, suggesting that local convolutional patterns alone are insufficient to capture long-range dependencies in sequential network traffic.

Fig. 9 shows that all three models achieve high detection accuracy for Benign traffic and several high-frequency attack categories in the NF-CSE-CIC-IDS2018 dataset, reflecting stable performance in modeling mainstream traffic features.

However, for attack categories with limited samples and complex feature distributions, the dual-branch model demonstrates more pronounced advantages. In minority attack classes such as *Brute Force-Web*, *SQL Injection*, *Brute Force-XSS*, and *Infiltration*, the dual-branch model significantly outperforms both the Transformer-only and CNN-only models.

These attack types often involve short-lived or stealthy behaviors that generate subtle traffic variations. As a result, their statistical flow features exhibit substantial overlap with other attack categories or benign traffic. The fusion of global temporal dependencies and local feature patterns enables the proposed architecture to better capture these subtle variations, leading to improved detection performance.

To evaluate the effectiveness of DWLF and to compare it with simpler imbalance-handling strategies, systematic ablation experiments were conducted on both datasets, as summarized in Table 4.

**Table 4:** Ablation study of the dynamic weighted logit-adjusted focal (DWLF) loss.

| Loss Configuration | UNSW-NB15 Acc (%) | CIC-IDS2018 Acc (%) |
|---|---|---|
| Standard Cross-Entropy Loss | 82.33 | 88.26 |
| + Logit Adjustment | 87.25 | 93.52 |
| + Focal Loss | 87.43 | 93.77 |
| + Static Re-Weighting | 89.51 | 94.30 |
| + Dynamic Weighting (Full DWLF) | 90.01 | 97.82 |

The ablation results in Table 4 show that each component of the DWLF loss contributes to performance improvements. Introducing logit adjustment significantly improves the baseline cross-entropy performance, indicating that correcting class-prior bias is beneficial in highly imbalanced intrusion detection scenarios.

Adding the focal modulation further enhances detection performance by increasing the contribution of hard-to-classify samples during training. Static class re-weighting provides additional improvement by compensating for global class imbalance.

Finally, incorporating the dynamic weighting mechanism yields the best overall results, achieving accuracies of 90.01% and 97.82% on UNSW-NB15 and CIC-IDS2018, respectively. This result suggests that dynamically adapting class weights according to batch-level distributions helps stabilize optimization under highly skewed traffic distributions.

### *4.7 Computational Complexity and Deployment Feasibility*

To assess the practical feasibility of ATC-FusionNet, we perform a quantitative complexity–performance analysis, reporting parameter count, FLOPs, training time per epoch, and inference latency in addition to detection accuracy. Two simplified baselines, namely CNN-only and Transformer-only models, are implemented with the same input representation and classifier head. Inference latency is measured as the average per-sample forward time over 100 runs under batch size 1 on an NVIDIA RTX 3090 Ti GPU, with the model evaluated in inference mode.

As shown in Table 5, ATC-FusionNet incurs a moderate increase in computational cost due to the dual-branch design and attention-based fusion. Nevertheless, the model remains compact with fewer than 1M parameters and an inference latency of approximately 3 ms per sample. Compared to the Transformer-only baseline, the additional overhead is modest, while consistently yielding improved detection performance. These results suggest that the architectural complexity of ATC-FusionNet is well motivated by its performance gains in challenging intrusion detection scenarios.

**Table 5:** Complexity comparison with baseline models.

| Model | Params (M) | FLOPs (G) | Train Time/Epoch (s) | Inference Latency (ms) |
|---|---|---|---|---|
| CNN | 0.42 | 0.18 | 21 | 1.2 |
| Transformer | 0.61 | 0.36 | 29 | 2.4 |
| ATC-FusionNet | 0.72 | 0.48 | 41 | 3.1 |

Memory usage during training and inference is primarily dominated by the Transformer encoder and the autoencoder components. Given the modest parameter scale (<1M parameters), the peak GPU memory footprint remains well within the capacity of a single modern GPU.

From a deployment perspective, the lightweight architecture and low inference latency further indicate that ATC-FusionNet is suitable for real-time intrusion detection scenarios. With an average inference time of approximately 3 ms per sample, the model can process network flow sequences efficiently without introducing significant delay in traffic monitoring pipelines.

Moreover, the compact parameter size and moderate computational requirements suggest that the model can be deployed not only on high-performance servers but also on resource-constrained environments, such as edge gateways or distributed network monitoring nodes. These characteristics make the proposed framework practical for large-scale or edge-oriented intrusion detection systems where both detection accuracy and computational efficiency are critical.

## 5 Conclusion

This study proposes an autoencoder-based dual-branch network intrusion detection model to address the limitations of existing methods in complex network environments, including low detection accuracy, insufficient recognition of minority attack classes, and inadequate feature representation. The model employs a parallel dual-branch architecture integrating Transformer [34] and CNN [35], enabling the simultaneous capture of long-range sequential dependencies and local critical behavior patterns. An attention-guided fusion mechanism (Attention Fusion) is incorporated to enhance the focus on key traffic patterns. At the feature input stage, a complete autoencoder-based feature reconstruction module is implemented, which improves the abstraction capability and stability of the model for high-dimensional network traffic through dimensionality reduction and reconstruction loss.

On the UNSW-NB15 datasets indicate that the proposed model attains an overall accuracy of 90.01% and a Macro-F1 score of 85.14%, indicating improved class-balanced performance beyond majority-class accuracy. In particular,for rare attack categories such as Analysis and Backdoor, the proposed model attains F1-scores of 0.7138 and 0.6784, respectively, which are higher than those typically reported by baseline models. On the NF-CSE-CIC-IDS2018 dataset, ATC-FusionNet achieves an accuracy of 97.82% with a Macro-F1 score of 77.20%, reflecting robust performance under more extreme class imbalance.

Despite these advancements, several challenges remain. In extremely imbalanced scenarios, the model may still produce misclassifications for certain highly similar attack categories. Specifically, samples from *Reconnaissance* and*Worms*, as well as*Brute Force-Web* and *SQL Injection* are occasionally misclassified as semantically related attack categories or benign traffic due to their short-lived, low-intensity behaviors and substantial overlap in flow-level statistical features. In addition, the dual-branch architecture combined with the autoencoder introduces computational overhead. Subsequent research will aim at lightweight models and high-throughput,low-latency real-time intrusion detection systems for practical deployment, aiming to further improve the model's operational viability and operational value.

**Acknowledgement:** Thanks to our tutors and researchers for their assistance and guidance.

**Funding Statement:** This work was supported in part by the National Natural Science Foundation of China under Grant U22A2004 and in part by Zhejiang Key Laboratory of Digital Fashion and Data Governance, Zhejiang Sci-Tech University, Hangzhou, China.

**Author Contributions:** Study conception and design: Liping Wang and Jiang Wu; analysis and intepretation of results: Liping Wang and Jiang Wu; data collection: Liang Wang; draft manuscript preparation: Liping Wang. All authors reviewed and approved the final version of the manuscript.

**Availability of Data and Materials:** The datasets used in this study are publicly available. The UNSW-NB15 dataset is available at https://research.unsw.edu.au/projects/unsw-nb15-dataset. The NF-CSE-CIC-IDS2018 dataset is available from the Canadian Institute for Cybersecurity at https://www.unb.ca/cic/datasets/ids-2018.html.

**Ethics Approval:** Not applicable.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Odeh A, Taleb AA. Robust network security: a deep learning approach to intrusion detection in IoT. Comput Mater Contin. 2024;81(3):4149–69. doi:10.32604/cmc.2024.058052.
2. Liu Z, Liu S, Zhang J. An industrial intrusion detection method based on hybrid convolutional neural networks with improved TCN. Comput Mater Contin. 2024;78(1):411–33. doi:10.32604/cmc.2023.046237.
3. Alrayes FS, Zakariah M, Amin SU, Khan ZI, Alqurni JS. Network security enhanced with deep neural network-based intrusion detection system. Comput Mater Contin. 2024;80(1):1457–90. doi:10.32604/cmc.2024.051996.
4. Deng J, Shen W, Feng Y, Lu G, Shen G, Cui L, et al. Lightweight intrusion detection using reservoir computing. Comput Mater Contin. 2024;78(1):1345–61. doi:10.32604/cmc.2023.047079.
5. Salih AA, Abdulrazaq MB. Cybernet Model: a new deep learning model for cyber DDoS attacks detection and recognition. Comput Mater Contin. 2024;78(1):1275–95. doi:10.32604/cmc.2023.046101.
6. Patel S, Kumar R, Singh G. Enhanced hybrid deep learning models-based anomaly detection method for two-stage binary and multi-class classification of attacks in intrusion detection systems. Algorithms. 2025;18(2):69. doi:10.3390/a18020069.
7. Neto ECP, Iqbal S, Buffett S, Sultana M, Taylor A. Deep learning for intrusion detection in emerging technologies: a comprehensive survey and new perspectives. Artif Intell Rev. 2025;58(11):340. doi:10.1007/s10462-025-11346-z.

8. Ayantayo A, Kaur A, Kour A, Schmoor X, Shah F, Vickers I, et al. Network intrusion detection using feature fusion with deep learning. J Big Data. 2023;10(1):167. doi:10.1186/s40537-023-00834-0.
9. Abdel-Basset M, Manogaran G, Mirjalili S, Surendiran M. Artificial intelligence driven cyberattack detection system using integration of deep belief network with convolutional neural network on industrial IoT. Alex Eng J. 2025;110(2):438–50. doi:10.1016/j.aej.2024.10.009.
10. Liu H, Lang B. Machine learning and deep learning methods for intrusion detection systems: a survey. Appl Sci. 2019;9(20):4396. doi:10.3390/app9204396.
11. Kamal H, Mashaly M. Advanced hybrid transformer-convolutional neural network (transformer-CNN) deep learning model for effective intrusion detection systems with class imbalance mitigation using resampling techniques. Future Internet. 2024;16(12):481. doi:10.3390/fi16120481.
12. Zhang C, Li J, Wang N, Zhang D. Research on intrusion detection method based on transformer and CNN-BiLSTM in Internet of Things. Sensors. 2025;25(9):2725. doi:10.3390/s25092725.
13. Gu J, Lu S. An effective intrusion detection approach using SVM with Naive Bayes feature embedding. Comput Secur. 2021;103(3):102158. doi:10.1016/j.cose.2020.102158.
14. Aljawarneh S, Aldwairi M, Yassein MB. Anomaly-based intrusion detection system through feature selection and hybrid model analysis. J Comput Sci. 2018;25(3):152–60. doi:10.1016/j.jocs.2017.03.006.
15. Yang J, Li T, Liang G, He W, Zhao Y. A simple recurrent unit model based intrusion detection system with DCGAN. IEEE Access. 2019;7:83286–96. doi:10.1109/ACCESS.2019.2922692.
16. Rajesh Kanna P, Santhi P. Unified deep learning approach for efficient intrusion detection using integrated spatial–temporal features. Knowl Based Syst. 2021;226(1):107132. doi:10.1016/j.knosys.2021.107132.
17. Wei D, Li X, Ji W, He S. Network intrusion detection method based on CNN, BiLSTM, and attention mechanism. IEEE Access. 2024;12:1–14. doi:10.1109/ACCESS.2024.3384528.
18. Liu J, Gao Y, Hu F. A fast network intrusion detection system using adaptive synthetic oversampling and LightGBM. Comput Secur. 2021;106(99):102289. doi:10.1016/j.cose.2021.102289.
19. Li X, Chen W, Zhang Q, Wu L. Building auto-encoder intrusion detection system based on random forest feature selection. Comput Secur. 2020;95(1):101851. doi:10.1016/j.cose.2020.101851.
20. Lopes IO, Zou DQ, Abdulqadder IH, Ruambo FA, Yuan B, Jin H. Effective network intrusion detection via representation learning: a denoising autoencoder approach. Comput Commun. 2022;194(5):55–65. doi:10.1016/j.comcom.2022.07.027.
21. Song Y, Hyun S, Cheong YG. Analysis of autoencoders for network intrusion detection. Sensors. 2021;21(13):4294. doi:10.3390/s21134294.
22. Vanlalruata H, Najar AA, Laldinsanga C, Hussain J, Hmingliana L. A lightweight intrusion detection system using deep convolutional neural network. Comput Electr Eng. 2025;127:110561. doi:10.1016/j.compeleceng.2025.110561.
23. Mohammadpour L, Ling TC, Liew CS, Aryanfar A. A survey of CNN-based network intrusion detection. Appl Sci. 2022;12(16):8162. doi:10.3390/app12168162.
24. Alashjaee AM. Deep learning for network security: an Attention-CNN-LSTM model for accurate intrusion detection. Sci Rep. 2025;15(1):21856. doi:10.1038/s41598-025-07706-y.
25. Manocchio LD, Layeghy S, Lo WW, Kulatilleke GK, Sarhan M, Portmann M. FlowTransformer: a transformer framework for flow-based network intrusion detection systems. Expert Syst Appl. 2024;241(10):122564. doi:10.1016/j.eswa.2023.122564.
26. Almadhor A, Alsubai S, Kryvinska N, Al Hejaili A, Ayari M, Bouallegue B, et al. Evaluating large transformer models for anomaly detection of resource-constrained IoT devices for intrusion detection system. Sci Rep. 2025;15(1):37972. doi:10.1038/s41598-025-21826-5.
27. Moustafa N, Slay J. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: Proceedings of the Military Communications and Information Systems Conference (MilCIS); 2015 Nov 10–12; Canberra, ACT, Australia. p. 1–6.
28. Sarhan M, Layeghy S, Moustafa N, Portmann M. NetFlow datasets for machine learning-based network intrusion detection systems. In: Big Data Technologies and Applications (BDTA 2020) and Wireless Internet (WiCON 2020). Cham, Switzerland: Springer; 2021. p. 117–35. doi:10.1007/978-3-030-72802-1_9.

29. Saidi Z, Ouidad A, El Idrissi Younes EB. A machine learning and blockchain-based framework for enhanced intrusion detection systems using the CSE-CIC-IDS2018 dataset. Informatica. 2025;49(18). doi:10.31449/inf.v49i18.9421.
30. Ahmad I, Basheri M, Iqbal MJ, Raheem A. Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection. IEEE Access. 2018;6:33789–95. doi:10.1109/ACCESS.2018.2841987.
31. Sarhan M, Layeghy S, Portmann M. Evaluating standard feature sets towards increased generalisability and explainability of ML-based network intrusion detection. Big Data Res. 2022;30(3):100359. doi:10.1016/j.bdr.2022.100359.
32. Halbouni A, Gunawan TS, Habaebi MH, Halbouni M, Kartiwi M, Ahmad R. CNN-LSTM: hybrid deep neural network for network intrusion detection system. IEEE Access. 2022;10(11):99837–49. doi:10.1109/ACCESS.2022.3206425.
33. Shivakanth G. A performance analysis of ML-based intrusion detection systems in cloud environments. Int J Electr Electron Eng Telecommun. 2025;14(4):243–52. doi:10.18178/ijeetc.14.4.243-252.
34. Wang J, Si C, Wang Z, Fu Q. A new industrial intrusion detection method based on CNN-BiLSTM. Comput Mater Contin. 2024;79(3):4297–4318. doi:10.32604/cmc.2024.050223.
35. Ren K, Yuan S, Zhang C, Shi Y, Huang Z. CANET: a hierarchical CNN-attention model for network intrusion detection. Comput Commun. 2023;205(16):170–81. doi:10.1016/j.comcom.2023.04.018.