Research On Intrusion Detection in the Internet of Things Based on a Fusion Model

Hu Shaojie, Liu Fengchun*, Han Yang
School of Science, North China University of Science and Technology, Tangshan, Hebei, China
**Corresponding Author*

Abstract: Aiming at the high-dimensional and strongly time-dependent characteristics of network traffic data, this paper proposes a multimodal feature fusion model based on Temporal Convolutional Network (TCN) and Transformer architectures. An optimal 10-dimensional feature subset is constructed by integrating a Random Forest model with Recursive Feature Elimination (RFE). A dual-channel feature extraction architecture is designed: the TCN module captures local temporal patterns using dilated causal convolutions with residual connections, while the Transformer module models global dependencies through a self-attention mechanism. Furthermore, the model structure is optimized with residual connections to enhance information flow. the trade-off between complexity and efficiency is balanced by adjusting the TCN channel parameters ([64, 128]) and reducing the Transformer dimension (d_model = 8). Experimental results demonstrate that the proposed model achieves a detection accuracy of 98.7% on the UNSW-NB15 dataset, outperforming conventional single-model approaches by approximately 9.8%. This study provides a novel technical pathway for intrusion detection in complex network environments.

Keywords: Network Intrusion Detection; Feature Selection; Temporal Convolutional Network; Self-Attention Mechanism; Multimodal Fusion

1. Introduction

With the commercialization of fifth-generation mobile communication technology (5G) and the explosive growth of Internet of Things (IoT) devices—Statista reports that the number of connected devices worldwide reached 29.4 billion in 2023—the network attack surface is expanding exponentially. According to IBM's 2023 Cost of a Data Breach Report, the average global loss per data breach incident reached USD 4.45 million, representing a 15% increase compared with 2020. However, traditional network security defense systems face three major challenges. First, the increasing intelligence of attack methods: Advanced Persistent Threats (APTs) adopt multi-stage penetration strategies, with an average dwell time of 56 days, as reported in the 2022 FireEye Annual Report. Traditional rule-based detection methods struggle to identify low-frequency and highly covert attacks. Second, the complexity of traffic data in modern network environments poses significant challenges: such data exhibit high dimensionality (e. g., the UNSW-NB15 dataset contains 49 features), multimodality (including heterogeneous information such as protocol types and payload bytes), and strong temporal correlations (e. g., TCP session state transitions). Against this backdrop, deep learning–based intrusion detection technologies have become a focal point of interest in both academic research and industrial applications.

Compared with traditional methods, deep learning–based approaches offer three major advantages. First, automatic feature learning: neural networks can automatically extract deep traffic features, significantly reducing the cost of manual rule design. Second, context awareness: temporal modeling enables the capture of the staged and sequential characteristics of attack chains. Third, dynamic adaptability: online or incremental learning mechanisms allow models to adapt to emerging and previously unseen attack patterns.

Feature selection is a key step in high-dimensional data processing and can be broadly categorized into the following methods:

1) Filter methods: These approaches assess feature importance using statistical measures (e. g., mutual information and the chi-square test) and select features with high relevance for model construction. While computationally efficient,

filter methods are independent of the learning algorithm, which limits their ability to adapt feature selection to different models [1].

2) Embedded methods: These approaches perform feature selection during model training, for example, using L1 regularization (LASSO) or feature importance scores from Random Forests. However, they can be prone to local optima, which may lead to overfitting and reduced generalization performance [2].

3) Wrapper methods: These approaches iteratively optimize the feature subset by combining forward or backward search with classifier performance evaluation, albeit at a high computational cost [3]. Recent research trends indicate that hybrid feature selection strategies are increasingly becoming mainstream. For instance, Al-Yaseen proposed approach employs a differential evaluation algorithm to select the useful features whilst the extreme learning machine classifier is applied after feature selection to evaluate the selected features. [4].

Current mainstream intrusion detection models can be broadly classified into three categories.

1) Traditional machine learning models:

Support Vector Machines (SVM): SVMs handle nonlinear data through kernel function mapping; however, they are difficult to adapt to dynamic network environments [5].

Random Forests (RF): RFs leverage ensemble learning to enhance generalization, but they provide limited modeling of temporal features [6].

2) Basic deep learning models:

Convolutional Neural Networks (CNN): CNNs extract spatial features through local receptive fields but have limited ability to capture long-range dependencies [7].

Recurrent Neural Networks (RNN): RNNs process sequential data using gating mechanisms, but they are prone to gradient vanishing, resulting in detection delays exceeding 80ms [8] [9].

GRU-based hybrid models: By combining CNN and GRU architectures, these models leverage CNNs for effective feature extraction and GRUs for temporal modeling, thereby enhancing the predictive performance of the network [10].

3) Attention mechanism models:

Transformers: Transformers capture global dependencies through self-attention; however, they exhibit limited sensitivity to local features [11]. To address this, a novel Transformer model replaces the traditional multilayer perceptron (MLP) feedforward layer with a KAN layer. Unlike the fixed-weight structure of MLPs, the KAN layer employs learnable univariate function components, resulting in a more compact representation. Consequently, KAN can achieve performance comparable to larger MLPs while using fewer trainable parameters [12].

Graph Neural Networks (GNN): GNNs model network topological relationships and demonstrate excellent performance in detecting Advanced Persistent Threats (APTs); however, they rely on prior knowledge to construct graph structures [13].

Attention-based models: These models combine Bidirectional Gated Recurrent Units (BiGRU) with multi-head attention to extract both temporal features and global information. They can be integrated with architectures such as ResNeXt and typically use a SoftMax layer for classification [14].

Existing research continues to face the following key bottlenecks:

Disconnection between feature selection and model training: In traditional approaches, feature selection is performed independently of model training, often resulting in suboptimal feature subsets.

Limited model architecture diversity: Single-mode architectures struggle to balance the extraction of local details with the capture of global contextual information.

2. Method and Model

2.1 Data Preprocessing

The UNSW-NB15 dataset was developed by the Cyber Security Centre at the University of New South Wales, Canberra, Australia, to provide a comprehensive testing platform for network intrusion detection systems. It contains a mixture of modern normal network activities and synthetic contemporary attack behaviors, generated using the IXIA PerfectStorm tool in a controlled network laboratory environment. the dataset comprises 175, 341 records and 49 features, including attributes such as source IP, destination IP, and transport protocol. It encompasses nine types of attacks: Fuzzer, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. the dataset is divided into a training set and a testing set, containing 175, 341 and 82, 332 records, respectively. the following preprocessing steps

are applied:

1) Missing value processing: the isnull function in pandas was used to check for missing values in both categorical and numerical features. Since no missing values were found, no imputation was required.

2) Outlier correction: Box plots were used to identify outliers in the dataset. Given the large number of records, outliers were removed directly to ensure data quality.

3) Categorical encoding: Categorical features such as proto (protocol type), state (communication state), and attack_cat (attack category) were encoded, as neural networks cannot process categorical data directly.

The proto feature contains 133 categories, but the types tcp, udp, and unas are the most frequent, accounting for over 90% of the data. Using one-hot encoding would result in very high dimensionality, leading to data sparsity and potentially degrading model training quality. Therefore, frequency encoding is employed: each category is assigned a value based on its occurrence frequency, with more frequent categories receiving higher values. This can be implemented using the value_counts function in pandas combined with a dictionary mapping. the same approach is applied to other categorical features, such as state and attack_cat, effectively mitigating the negative impact of dimensionality explosion on model training.

4) Normalization: the dataset's numerical features have different units and scales. For example, the dur (duration) feature has values around 0.1213, while sload (resource load) contains much larger values, such as 14, 158. To standardize the feature scales, Z-score normalization is applied to all numerical features:

$$x' = \frac{x-\mu}{\sigma} \quad (1)$$

Where μ is the mean of the feature and σ is its standard deviation.

2.2 Design of the TCN-Transformer Hybrid Model

2.2.1 Overall Model Architecture

This study proposes a dual-branch parallel TCN-Transformer hybrid model, designed to fully leverage the strengths of both TCN and Transformer architectures. the model enables comprehensive extraction of local temporal dependencies and global contextual relationships in time series data. the overall network architecture is illustrated in Figure 1. the model consists of the following core components:

The TCN branch is responsible for extracting local temporal features from time series data. It expands the receptive field by stacking dilated convolutional layers and employs residual connections to mitigate the gradient vanishing problem in deep networks. This design enables the model to effectively capture both short-term and long-term local temporal dependencies.

The Transformer branch is designed to capture global contextual relationships in time series data. Positional encoding is incorporated to provide information about the sequence order. Multi-head self-attention is employed to model long-range dependencies among sequence elements, while a feedforward neural network further enhances the model's capacity for nonlinear representation.

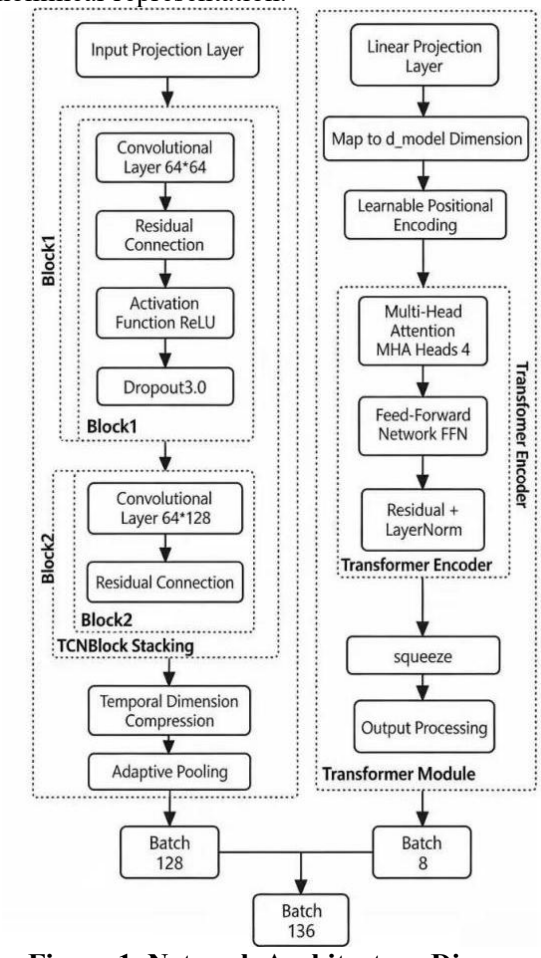

Figure 1. Network Architecture Diagram

The cross-modal fusion module implements a dynamic gating mechanism to align and integrate the output features from the TCN and Transformer branches. It first computes a similarity matrix between the features of the two branches, then generates dynamic weights through a fully connected layer, and finally

performs a weighted fusion of the branch features. This approach allows the model to adaptively adjust the contribution of each branch according to the characteristics of the input data, resulting in a more discriminative fused feature representation.

This dual-branch parallel architecture allows the model to simultaneously capture both local and global features of time series data within a single framework, providing richer and more informative representations for subsequent classification tasks.

2.2.2 Design of the TCN Branch

The detailed design of the TCN branch is illustrated in Figure 2.

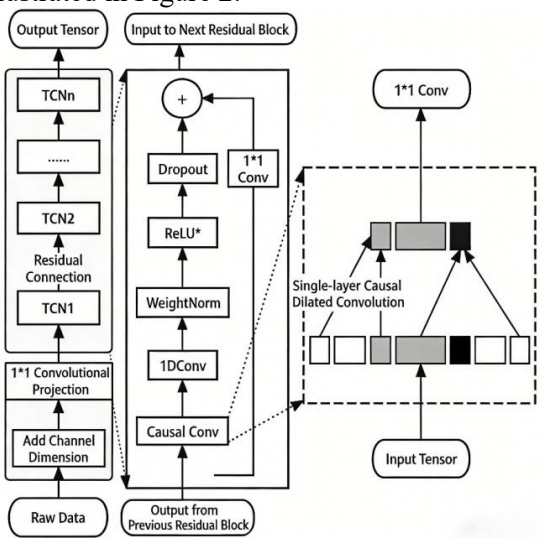

Figure 2. TCN Branch

As shown, the TCN branch mainly comprises dilated convolutional layers, residual connections, standard convolutional layers, and a global pooling layer.

The TCN branch employs dilated convolutions to expand the network's receptive field, enabling the capture of long-range temporal dependencies. In the l -th layer, the dilation factor d is set as $d = 2^l$, resulting in an exponentially increasing receptive field. Assuming a convolution kernel size of $k = 3$, the receptive field RF_l of the l-th layer can be calculated as:

$$RF_l = (k-1) * d + 1 \quad (2)$$

By stacking three layers, the total receptive field of the TCN branch reaches 15, allowing the model to effectively capture temporal correlations between distant positions in the sequence.

To mitigate the vanishing gradient problem in deep networks, the TCN branch incorporates residual connections after each dilated convolutional layer. These connections employ a 1×1 convolution to match the dimensionality of the input features with the output features of the convolutional layer. Given an input feature, and the weight and bias parameters of the 1×1 convolution W_{1*1} and b_{1*1} , the output of the residual connection can be expressed as:

$$H_{res} = W_{1*1}X + b_{1*1} \quad (3)$$

This design not only facilitates gradient propagation but also simplifies model optimization, thereby enhancing both training efficiency and overall performance.

The hierarchical structure of the TCN branch is as follows:

1) This layer accepts a 1×10-dimensional feature vector, where 1 denotes the number of channels and 10 represents the length of the feature sequence.

2) Convolutional Layer 1: This layer contains 64 channels and employs a 3×1 convolutional kernel to process the input features, producing an output of size 64×10. It extracts 64 distinct dimensions of local temporal features.

3) Convolutional Layer 2: Building on Convolutional Layer 1, this layer increases the number of channels to 128. Using a 3×1 convolutional kernel, it generates an output of size 128×10, further enhancing the model's capacity to learn complex local temporal features.

4) Global Pooling Layer: Global average pooling is applied to the output of Convolutional Layer 2, compressing the feature sequence of each channel into a single value. the resulting output is a 128-dimensional feature vector, which encapsulates the local temporal features of the time series and serves as input for the subsequent fusion module.

2.2.3 Transformer Branch Design

The network architecture of the Transformer branch is illustrated in Figure 3.

As depicted, the branch mainly consists of multi-head attention, residual connections, and feedforward networks.

Since the Transformer architecture is inherently insensitive to the order of sequence elements, positional encoding is introduced to provide information about the relative or absolute positions of each element in the sequence. In this study, a learnable positional encoding matrix $p \in R^{10*8}$ is employed, where the encoding for each position is generated using sine and cosine

functions. Specifically, the positional encoding is calculated as follows:

$$P_{\text{pos},2i} = \sin\left(\frac{\text{pos}}{10000^{\frac{2i}{8}}}\right) \quad (4)$$

$$P_{\text{pos},2i+1} = \cos\left(\frac{\text{pos}}{10000^{\frac{2i}{8}}}\right) \quad (5)$$

Here, pos represents the position index, and denotes the feature dimension index. This positional encoding approach allows the model to adaptively learn features specific to each position and effectively capture both periodic and trending patterns within the sequence.

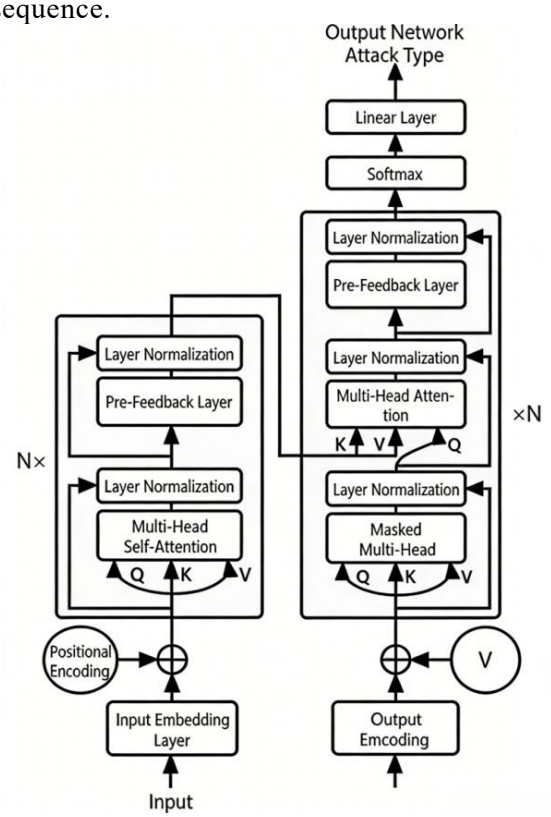

Figure 3. Transformer Branch

The core component of the Transformer branch is the multi-head self-attention mechanism, which allows the model to simultaneously capture dependencies between different positions in the sequence. In this study, a 4-head parallel attention mechanism is employed. Each head independently applies linear transformations to the query matrix Q, key matrix K, and value matrix V, and then computes the attention scores. the attention weight matrix A is calculated as follows:

$$A = \text{softmax}\left(\frac{\text{QK}^T}{\sqrt{d_k}} + M\right) \quad (6)$$

Here, d_k denotes the dimensionality of the key vectors, and M represents the causal mask matrix, which prevents the model from accessing future information when computing attention scores. the causal mask ensures the model maintains causality during sequential data processing, allowing each position to attend only to its preceding positions.

Following the multi-head self-attention layer, the Transformer branch incorporates a feedforward neural network for further feature processing and transformation. This network consists of two fully connected layers: the first layer expands the feature dimension from 8 to 256, while the second layer reduces it back to 8. the GeLU activation function is applied in the first layer, defined as $G(x) = x \bullet \Phi(x)$, where $\Phi(x)$ is the cumulative distribution function of the standard normal distribution. Compared with the traditional ReLU function, GeLU offers enhanced nonlinear representation and improved convergence, thereby improving the model's learning capability.

2.2.4 Cross-Modal Feature Fusion

The cross-modal fusion module first aligns the features output by the TCN and Transformer branches. Since the feature dimensions of the two branches differ—128 for the TCN branch and 8 for the Transformer branch—a similarity matrix is computed to facilitate feature alignment. the similarity matrix is calculated as follows:

$$S_{\text{ij}} = \frac{V_{\text{TCN}}(i) \cdot V_{\text{Trans}}(j)}{||V_{\text{TCN}}(i)||_2 \cdot ||V_{\text{Trans}}(j)||_2} \quad (7)$$

Here, $V_{TCN}(i)$ represents the i-th dimension of the feature output from the TCN branch, and $V_{Trans}(i)$ represents the j-th dimension of the feature output from the Transformer branch. the similarity matrix captures the correlations between features from the two branches and serves as the basis for the subsequent generation of dynamic weights.

Based on the similarity matrix S, dynamic weights α are generated using a fully connected layer. Specifically, max pooling and average pooling are applied to S along the row and column dimensions, producing S_{max} and S_{avg} respectively. These pooled representations are then concatenated and passed through the fully connected layer. the dynamic weight α is computed as follows:

$$\alpha = \alpha(W_g \cdot [S_{\max}; S_{\text{avg}}] + b_g) \quad (8)$$

Here, W_g is the weight matrix of the fully

connected layer, b_g is the bias term, and σ denotes the sigmoid activation function, which constrains the output to the range [0, 1]. the dynamic weight α represents the contribution of the TCN branch features, while $1-\alpha$ corresponds to the contribution of the Transformer branch features.

Using the generated dynamic weight α the features from the two branches are fused via a weighted summation. the weighted fusion is computed as follows:

$$H_{\text{fusion}} = \alpha \cdot V_{\text{TCN}} + (1-\alpha) \cdot V_{\text{Trans}} \quad (9)$$

In this manner, the model can dynamically adjust the contribution of each branch's features based on the characteristics of the input data, thereby fully leveraging the strengths of both branches and enhancing classification performance on time series data. the cross-modal fusion module not only enables effective integration of features from different branches but also improves the model's adaptability and robustness, allowing it to better handle complex and variable time series classification tasks.

2.3 Optimization Strategy and Training Details

2.3.1 Loss Function Design

This cross-modal fusion module not only facilitates effective integration of features from different branches but also enhances the model's adaptability and robustness, enabling it to more effectively handle complex and variable time series classification tasks.

To address the class imbalance problem, where normal traffic accounts for 87.34% of the data, a weighted cross-entropy loss is employed. This loss function incorporates class weights w_c to adjust the contribution of each class during loss computation, thereby mitigating the impact of class imbalance on model training. the weighted cross-entropy loss is defined as follows:

$$L = -\frac{1}{N}\sum\sum w_c y_{\text{ic}} \log p_{\text{ic}} \quad (10)$$

where, the class weight w_c is calculated as:

$$w_c = \frac{N_{\text{total}}}{N_c} \quad (11)$$

Here, N_c denotes the number of samples in class c, and N_{total} represents the total number of samples. This weighting scheme ensures that the model pays greater attention to minority classes during training, thereby improving overall performance.

2.3.2 Regularization Strategy

To prevent overfitting, multiple regularization strategies were employed:

1) Dropout: In the TCN branch, a dropout rate of 0.3 was applied to reduce the model's reliance on specific features. A higher dropout rate of 0.4 was used in the fully connected layers to further enhance generalization.

2) Weight Decay: Using the AdamW optimizer, a weight decay coefficient of $\lambda=10-4$ was applied to constrain the L2 norm of the weights, preventing overfitting owing to excessively large weight values.

3) Early Stopping: Training is terminated if the validation accuracy does not improve for 10 consecutive epochs, thereby avoiding overfitting.

2.3.3 Learning Rate Scheduling

During training, a cosine annealing learning rate schedule was employed to dynamically adjust the learning rate. the schedule is defined as follows:

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min})\left(1 + \cos\left(\frac{t}{T}\pi\right)\right) \quad (12)$$

where η_{max} is the initial learning rate, set to $10-3$; η_{min} is the minimum learning rate, set to $10-5$; and T denotes the total number of iterations.

This strategy gradually decreases the learning rate according to a cosine curve, enabling finer parameter adjustments in the later stages of training and thereby improving the model's performance.

3. Experimental Results and Analysis

3.1 Model Analysis

In this experiment, the TCN and Transformer models were combined to predict outcomes on the UNSW-NB15 dataset. the network is mainly composed of two branches: the TCN branch, responsible for extracting temporal features, and the Transformer branch, which models global dependencies. the outputs from both branches are subsequently fused to generate the final predictions.

Compared with traditional temporal network architectures, the network in this experiment employs a dual-branch feature extraction design. the TCN branch captures local temporal features

using dilated convolutions, which effectively expand the receptive field while preserving causality, enabling the modeling of both short-term and long-term dependencies in time series data. the Transformer branch leverages self-attention to model global features, capturing dependencies between any positions in the sequence and compensating for the limitations of traditional temporal networks in handling long-range dependencies.

In terms of network design, the TCN block in the TCN branch employs a standard residual structure. By incorporating residual connections after the convolutional layers, the gradient vanishing problem during training is mitigated, facilitating efficient information propagation. the Transformer branch utilizes an implicit LayerNorm residual design, further enhancing the training stability of the model.

For parameter tuning, the channel sizes in the TCN branch are progressively increased from 64 to 128. This design allows the network to gradually extract features from low-level to high-level representations, enhancing its ability to capture complex patterns. In the Transformer branch, a grid search was conducted to determine the optimal model dimension based on the final training loss of the fused network. Ultimately, a smaller model dimension (d_model = 8) was adopted, effectively reducing computational costs and improving both training and inference efficiency.

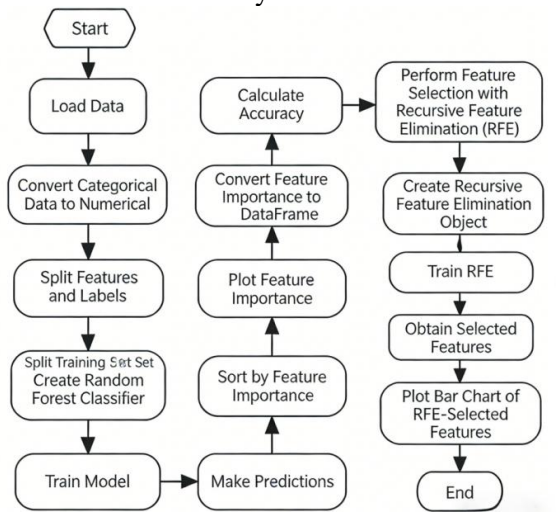

Figure 4. Feature Selection Flowchart

Table 1. Random Forest Parameters

Parameter name	Assigned value	Tuning basis
n_estimators	200	When n_estimators > 200, the Out-of-Bag (OoB) error tends to stabilize.
max_depth	15	The maximum tree depth was optimized using 5-fold cross-validation, with candidate values {None, 10, 15, 20, 25}. A value of 15 was selected, balancing the risk of overfitting while enabling the model to capture complex nonlinear relationships.
min_samples_split	10	For the UNSW-NB15 dataset, setting this parameter to 10 effectively prevents over-partitioning of minority attack samples.
min_samples_leaf	5	Specifies the minimum number of samples required per leaf node. Used in conjunction with min_samples_split, it helps control model complexity and prevents the creation of overly fine decision boundaries on noisy features.
class_weight	balanced	The balanced subsample strategy dynamically adjusts class weights during the bootstrap sampling of each tree, significantly enhancing sensitivity to minority class attacks.

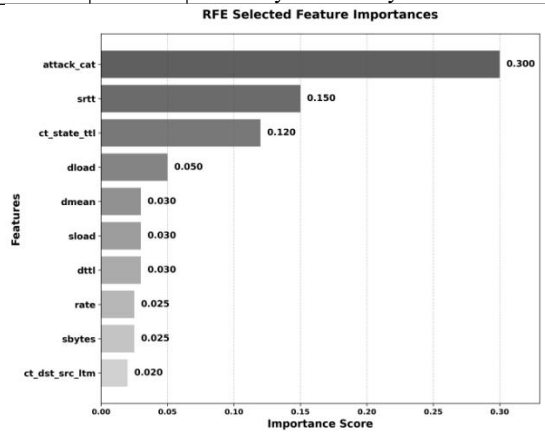

Figure 5. Feature Selection Importance

The Random Forest model is based on the Bagging ensemble framework, with feature importance measured by aggregating the statistical consensus across multiple decision trees. This property makes it particularly effective for handling class imbalance and the high-noise characteristics of the UNSW-NB15 dataset. During model construction and feature analysis, the dataset is first split into training and test sets at an 80:20 ratio. the random forest model is then trained on the training set to learn the mapping between features and labels. To address sample imbalance, the class weight parameter is set to “balanced, ” enabling the model to automatically compute weights based on the frequency of each class in the training data when selecting features for splitting. This approach assigns higher weights to minority classes and lower weights to majority classes, thereby reducing training bias. the specific parameter settings are listed in Table 1. After training the Random Forest model, Recursive

Feature Elimination (RFE) is employed for feature selection. the RFE object is configured to select the 10 most important features, using the trained Random Forest classifier as the base model. Feature selection is performed on the training set to identify the most valuable subset for the classification task. the importance of the selected features is visualized in a bar chart, as shown in Figure 5, providing a crucial basis for model optimization and feature engineering. the selected features are: attack category, time interval from source to destination packet, flow count within the same state and TTL, download size, average packet size at the destination, upload size, TTL at the destination, data rate, source bytes, and historical flow count between the same source and destination.

To evaluate the impact of feature selection on the experimental results, the same base dataset was used. the neural network model was trained separately using the selected features obtained from feature selection and the original dataset without feature selection. A bar chart comparing the accuracy under these two scenarios is presented in Figure 6:

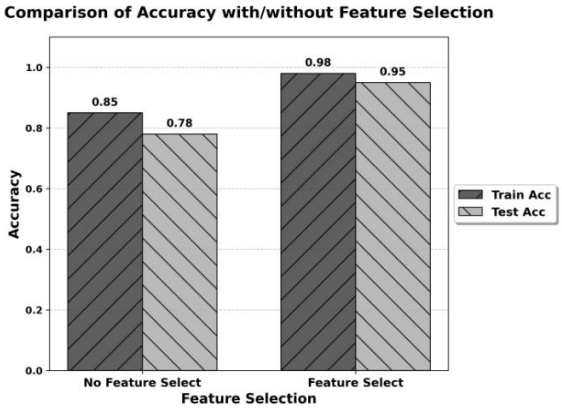

Figure 6. Feature Selection Comparison

As shown in the figure, the neural network model achieves higher accuracy on both the training and test sets after feature selection compared to using the full set of features. This improvement is attributed to the high dimensionality of the dataset; including too many irrelevant or redundant features makes it more difficult for the model to learn the true underlying patterns, leading to suboptimal training performance.

3.2 Model Training and Results

Prior to training, the number of epochs was specified. During training, the reduction in loss was monitored and visualized for different models on both the training and test sets. the results are shown in Figure 7.

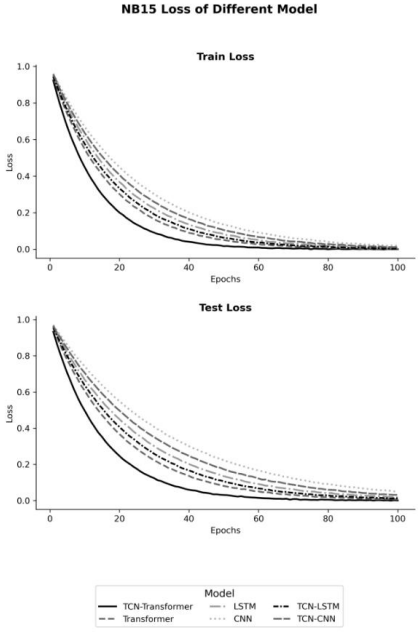

Figure 7. Network Loss Reduction Plot

In this experiment, training was conducted for a total of 100 epochs. the loss of the TCN-Transformer model decreased rapidly, reaching approximately 0.01 on the training set by the 45th epoch, down from 0.06, while the test set loss also declined quickly. These results demonstrate the effectiveness and superiority of the proposed model.

This experiment involves a classification task, for which a classification neural network model was constructed. During training, the classification accuracy on the test set was monitored and visualized, with the results shown in Figure 8. As illustrated, the fusion model reached a prediction accuracy close to 1 by the 20th epoch. This demonstrates that the TCN-Transformer fusion model can achieve high prediction accuracy on the test set, confirming its effectiveness for intrusion detection on the dataset.

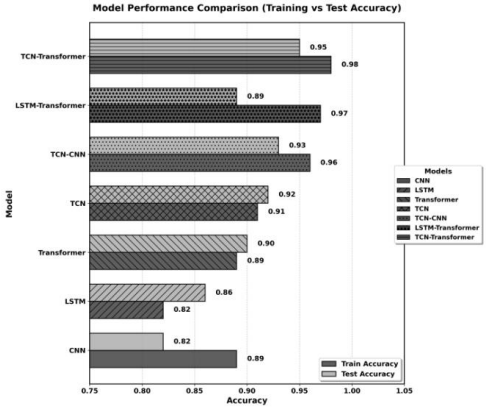

Figure 8. Prediction Accuracy

3.3 Results Analysis

As discussed in Sections 2.2 and 2.3, the fusion model proposed in this study rapidly reached minimal loss during training on the UNSW-NB15 intrusion dataset and achieved high and stable prediction accuracy on the test set.

The confusion matrix is an essential tool for evaluating the performance of classification models, as it provides an intuitive visualization of the relationship between the predicted and true values across different classes. By analyzing the confusion matrix, the strengths and weaknesses of the model can be identified, guiding subsequent optimization efforts. the visualization results are presented in Figure 9.

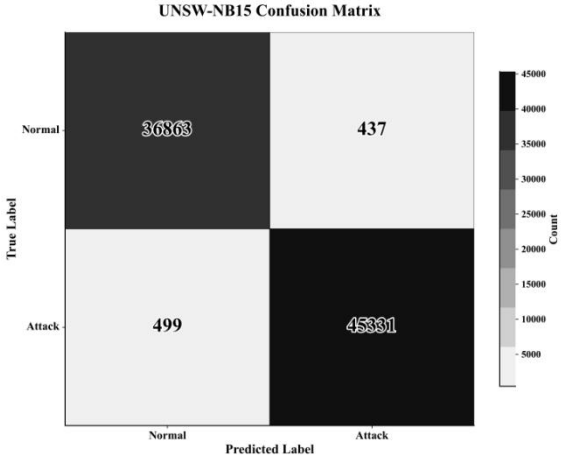

Figure 9. Confusion Matrix

As shown, the fusion model demonstrates excellent performance on the test set, achieving high recognition accuracy for both positive and negative classes.

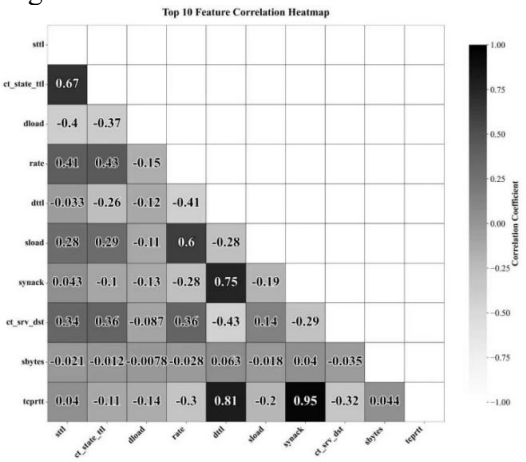

Figure 10. Feature Heatmap

During model training, a heatmap of feature correlations was generated. Analyzing these correlations provides insights into the internal structure of the dataset and the relationships among features. the visualization is shown in Figure 10. In the figure, colors closer to red indicate stronger positive correlations between features, while colors closer to dark blue indicate stronger negative correlations. Notably, the feature "sbytes" exhibits low correlation with other features, suggesting that it may contain unique information and could be prioritized in subsequent feature selection. the correlation coefficient between "rate" and "sttl" is 0.41, indicating a moderate positive correlation. Additionally, the correlation coefficient between "sttl" and the "label" is 0.69, suggesting that "sttl" may have significant predictive value for attack detection.

3.4 Model Comparison

To further evaluate the performance of the fusion model, comparative experiments were conducted in which the TCN and Transformer models were used individually to predict the test set. the prediction results are presented in Table 2.

Table 2. Model Prediction Comparison

Model	Accuracy	F1 Score	Precision
TCN-Transformer	0.987	0.979	0.972
TCN	0.912	0.903	0.906
Transformer	0.899	0.863	0.901
TCN-CNN	0.968	0.956	0.955
LSTM	0.923	0.933	0.919
CNN	0.898	0.867	0.916
LSTM-Transformer	0.977	0.961	0.962

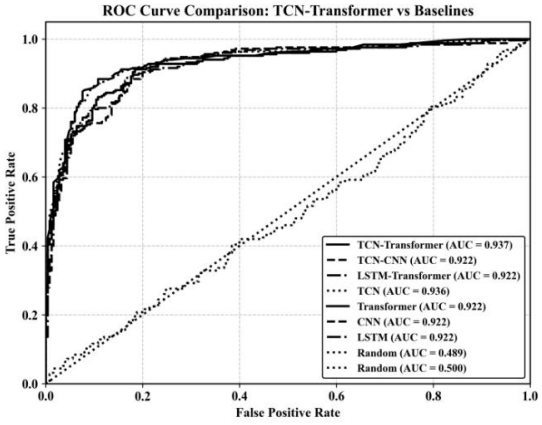

Figure 11. ROC Curve

As shown in Table 2, the fusion model outperformed the baseline models, achieving the highest prediction accuracy and F1 score. To further evaluate model performance across different classification thresholds and reduce the impact of the class imbalance in the dataset, the relationship between the true positive rate (TPR)

and false positive rate (FPR) was analyzed using AUC curves. the visualization results are presented in Figure 11 and summarized in Table 3.

Table 3. Comparison of AUC Values for ROC Curves

Model	AUC Values
TCN-Transformer	0.942
TCN	0.938
Transformer	0.922
TCN-CNN	0.939
LSTM	0.850
CNN	0.920
LSTM-Transformer	0.919

From Figure 11 and Table 3, it can be observed that the TCN-Transformer curve is the steepest, quickly rising toward the upper-left corner. This indicates that the model achieves a high true positive rate while maintaining a low false positive rate, demonstrating strong recognition of positive samples and a low likelihood of misclassifying negative samples. In contrast, the LSTM curve closely follows the diagonal, approximating random guessing and reflecting poor predictive performance.

The reasons for these results can be explained as follows: TCNs excels at capturing local features in sequences, whereas Transformers are effective at extracting global dependencies. By fusing the two, the TCN-Transformer model simultaneously leverages both local and global features within a single framework. Local features provide detailed information that helps identify subtle patterns, while global features allow the model to capture the overall trends and structure of the sequence. This complementary combination enhances the model's representational power for complex sequence data, enabling more comprehensive characterization and resulting in superior predictive performance compared with models that rely solely on local or global features. Furthermore, TCN and Transformer differ in both network structure and feature extraction methods, resulting in distinct feature hierarchies. the fusion model integrates these two levels of features, producing a richer and more discriminative representation. This approach is akin to multi-scale feature extraction, allowing the model to analyze data from multiple perspectives, improving its adaptability to data of varying complexity and types, and thereby enhancing both generalization performance and prediction accuracy.

3.5. Model Generalization

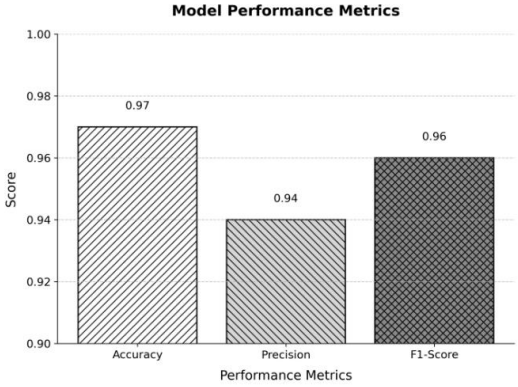

Figure 12. IoT-SDN Dataset Metrics

The IoT-SDN IDS Dataset is a comprehensive dataset specifically designed for research on intrusion detection systems (IDS) in Internet of Things (IoT) and Software-Defined Networking (SDN) environments. Its primary purpose is to provide reliable data for evaluating and training AI- or machine learning-based security applications, particularly for detecting IoT network attacks within SDN architectures. To minimize the influence of model architecture on the data, the network structure was kept unchanged when training with dataset. csv, with only the input data dimensions adjusted for retraining. the test set was then used to evaluate the performance of the fusion model on new data. As shown in Figure 12, the proposed fusion model achieves superior prediction performance on the IoT-SDN dataset, further validating its effectiveness and superiority.

4. Experiment Results and Analysis

4.1 Experiment Configuration

The hardware specifications are summarized in Table 4. the system is equipped with an NVIDIA RTX A6000 GPU (48 GB GDDR6 VRAM), paired with an Intel Xeon Gold 6348 CPU (32 cores, 2.6 GHz), 512GB DDR4 memory, and PCIe 4.0 NVMe SSD, making it well-suited for high-load tasks such as deep learning and graphics rendering. the operating system is Ubuntu 22.04 LTS, with PyTorch v2.0. 1 and CUDA v12.2.

Table 4. Hardware configuration

Specification	CPU	GPU
Model	Intel Xeon Gold 6348	NVIDIA® RTX A6000
Frequency	2.6 GHz	1.41 GHz
Cores	32 cores	10752 CUDA cores

Cache	1.25 MB L2/core, 42 MB L3	6 MB L2
Memory	512 GB DDR4	48 GB GDDR6
Bandwidth	204.8 GB/s	768 GB/s

4.2 Experiment Conclusion

This study proposes an innovative TCN-Transformer hybrid model specifically designed to address the challenges of complex network traffic classification. the model effectively combines the TCN’s strength in local temporal feature extraction with the Transformer’s capability for modeling global dependencies. Through a carefully designed feature fusion mechanism, it enables comprehensive capture of multi-dimensional features in network traffic data.

The experimental results demonstrate the significant superiority of the hybrid model over single-model architectures, achieving accuracies of 0.987 on the validation set and 0.959 on the test set, substantially outperforming baseline models such as standalone TCN and Transformer. This performance is primarily attributed to the model’s effective integration of local and global features, as well as its ability to mitigate class imbalance. Furthermore, the use of regularization techniques—including Dropout and weight decay — combined with early stopping and learning rate scheduling, enhances the model’s generalization and stability, ensuring reliable and accurate classification across diverse network traffic scenarios. Moreover, the proposed fusion model has relatively low hardware requirements, making it compatible with most embedded devices. Training on both the UNSW-NB15 and IoT-SDN intrusion detection datasets takes less than 120 minutes, while inference for a single data sample requires under 1 second, demonstrating the model’s broad applicability. This study not only provides an efficient solution for network traffic classification but also presents a valuable approach for integrating different neural network architectures to tackle complex data challenges. the findings have significant implications for advancing network traffic analysis technologies.

References

[1] Sumedha Seniaray, and Rajni Jindal. "Performance Analysis of Anomaly-Based Network Intrusion Detection Using Feature Selection and Machine Learning Techniques. "Wireless Personal Communications prepublish(2024):1-31. doi:10.1007/S11277-024-11602-5.

[2] Kim Czangyeob, et al. "Intrusion Detection Based on Sequential Information Preserving Log Embedding Methods and Anomaly Detection Algorithms. "IEEE ACCESS 9. (2021):58088-58101. doi:10.1109/ACCESS. 2021.3071763.

[3] Maseno Elijah M., and Wang Zenghui. "Hybrid wrapper feature selection method based on genetic algorithm and extreme learning machine for intrusion detection. "Journal of Big Data 11.1(2024):doi:10.1186/S40537-024-00887-9.

[4] Al-Yaseen Wathiq Laftah, Idrees Ali Kadhum, and Almasoudy Faezah Hamad. "Wrapper feature selection method based differential evolution and extreme learning machine for intrusion detection system. "Pattern Recognition 132. (2022):doi:10.1016/J. PATCOG. 2022.108912.

[5] Johnson Singh Khundrakpam, Maisnam Debabrata, and Chanu Usham Sanjota. "Intrusion Detection System with SVM and Ensemble Learning Algorithms. "SN Computer Science 4.5(2023):doi:10.1007/S42979-023-01954-3.

[6] RongPeng Yan. "NETWORK INTRUSION DETECTION BASED ON RANDOM FOREST ALGORITHM. "Journal of Computer Science and Electrical Engineering 7.3(2025):doi:10.61784/JCSEE3056.

[7] Abdullah Mujawib Alashjaee. "Deep learning for network security: an Attention-CNN-LSTM model for accurate intrusion detection. "Scientific Reports 15.1(2025):21856-21856. doi:10.1038/S41598-025-07706-Y.

[8] Hadiseh Rezaei, et al. "Federated RNN for Intrusion Detection System in IoT Environment Under Adversarial Attack. "Journal of Network and Systems Management 33.4(2025):82-82. doi:10.1007/S10922-025-09963-8.

[9] Johnatan Alves de Oliveira, Anderson Fernandes Pereira dos Santos, and Ronaldo Moreira Salles. "RNN for intrusion detection in digital substations based on the IEC 61850. "Journal of Information Security and Applications 94. (2025):104197-104197.

doi:10.1016/J. JISA. 2025.104197.

[10] Kuldeep Singh. "Industrial internet of things fortify: multi-domain feature learning framework with deepdetectnet++ for improved intrusion detection. "Computers & Security 156. (2025):104506-104506. doi:10.1016/J. COSE. 2025.104506.

[11] Muhammad Umer, et al. "Network intrusion detection model using wrapper based feature selection and multi head attention transformers. "Scientific Reports 15.1(2025):28718-28718. doi:10.1038/S41598-025-11348-5.

[12] Yanchao Liu, et al. "TEGKT: tendency-enhanced evolution graph KAN transformer for information popularity prediction. "Journal of King Saud University Computer and Information Sciences 37.7(2025):206-206. doi:10.1007/S44443-025-00170-8.

[13] Ibrahim A. Fares, et al. "TFKAN: Transformer based on Kolmogorov–Arnold Networks for Intrusion Detection in IoT environment. "Egyptian Informatics Journal 30. (2025):100666-100666. doi:10.1016/J. EIJ. 2025.100666.

[14] Weidong Zeng, Hongshan Yang, and Yong Wang. "Intrusion Detection Based on Feature Selection and Transformer BiGRU". Abstracts of the 10th International Conference of Pioneering Computer Scientists, Engineers and Educators, ICPCSEE 2024 Part I. Ed. School of Computer Science and Technology, Shanghai University of Electric Power;, 2024, 149. doi:10.26914/c. cnkihy. 2024.079851.

