


Article

Intrusion Detection System Based on Multi-Level Feature Extraction and Inductive Network

Junyi Mao ^{1,2,†}, Xiaoyu Yang ^{1,†} , Bo Hu ¹, Yizhen Lu ³ and Guangqiang Yin ^{1,3,4,*}

¹ School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 611730, China; jmiao0023@student.monash.edu (J.M.); yangxy@std.uestc.edu.cn (X.Y.); hubo@std.uestc.edu.cn (B.H.)

² Faculty of Information Technology, Monash University, Clayton, VIC 3800, Australia

³ Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China, Shenzhen 518110, China; 202152280911@std.uestc.edu.cn

⁴ Kashgar Regional Electronic Information Industry Technology Research Institute, Kashi 844000, China

* Correspondence: yingq@uestc.edu.cn

† These authors contributed equally to this work.

Abstract: With the rapid development of the internet, network security threats are becoming increasingly complex and diverse, making traditional intrusion detection systems (IDSs) inadequate for handling the growing variety of sophisticated attacks. In particular, traditional methods based on rule matching and manual feature extraction demonstrate significant limitations in dealing with small samples and unknown attacks. This paper proposes an intrusion detection system based on multi-level feature extraction and inductive learning (MFEI-IDS) to address these challenges. The model innovatively integrates Fully Convolutional Networks (FCNs) with the Transformer architecture (FCN-Transformer) for feature extraction and utilizes an inductive learning component for efficient classification. The FCN-Transformer Encoder extracts multi-level features from raw network traffic, capturing local spatial patterns and global temporal dependencies, significantly enhancing the representation of network traffic while reducing reliance on manual feature engineering. The inductive learning module employs a dynamic routing mechanism to map sample feature vectors into robust class vector representations, achieving superior generalization when detecting unseen attack types. Compared to existing FCN-Transformer models, MFEI-IDS incorporates inductive learning to handle data imbalance and small-sample scenarios. Experiments on ISCX 2012 and CIC-IDS 2017 datasets show that MFEI-IDS outperforms mainstream IDS methods in accuracy, precision, recall, and F1-score, excelling in cross-dataset validation and demonstrating strong generalization capabilities. These results validate the practical potential of MFEI-IDS in small-sample learning, unknown attack detection, and dynamic network environments.

Keywords: intrusion detection; inductive networks; small-sample learning; FCN-transformer; unknown attacks



Academic Editor: Aryya Gangopadhyay

Received: 9 October 2024

Revised: 29 December 2024

Accepted: 3 January 2025

Published: 5 January 2025

Citation: Mao, J.; Yang, X.; Hu, B.; Lu, Y.; Yin, G. Intrusion Detection System Based on Multi-Level Feature Extraction and Inductive Network. *Electronics* **2025**, *14*, 189. <https://doi.org/10.3390/electronics14010189>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

With the rapid development of information technology and the internet, the number of globally connected devices has increased significantly, highlighting the growing importance of cyberspace. However, this progress has also exacerbated the complexity and frequency of cyberattacks, threatening personal information, business data, and national security. Traditional intrusion detection systems (IDSs) have been widely deployed to safeguard

network security, and are broadly categorized into rule-based, signature-based, misuse-based, and anomaly-based systems [1]. While these systems perform well against known attacks, their reliance on predefined rule sets or manually extracted features imposes critical limitations when identifying novel or evolving threats.

In recent years, with the rapid advancement of deep learning technologies, researchers have applied them to the field of cybersecurity, attempting to improve intrusion detection through automated feature learning. For instance, convolutional neural networks (CNNs) excel in recognizing spatial patterns in network flows, while Recurrent Neural Networks (RNNs) are adept at capturing temporal dependencies. Deep learning methods perform remarkably well in scenarios with abundant labeled data but often struggle to generalize in environments with limited labeled data. For example, in real-world network traffic, most data are normal, while labeled malicious traffic data are sparse and diverse, making it challenging for traditional deep learning models to effectively capture the characteristics of malicious traffic [2]. Furthermore, as attack patterns evolve, models trained on past data quickly lose their effectiveness in real-world settings. For instance, traditional models have demonstrated insufficient robustness against supply chain attacks and Advanced Persistent Threats (APTs) emerging after 2020 [3].

Few-shot learning has emerged as a promising solution to address these challenges. Techniques such as Matching Networks and Prototypical Networks have shown potential in leveraging limited samples to detect unknown attacks. However, existing few-shot learning methods exhibit limitations in high-dimensional and dynamic network traffic scenarios. This is primarily due to their reliance on simple similarity measures (e.g., Euclidean distance), which perform poorly in multi-dimensional network traffic data, such as traffic data with temporal sequence features. For instance, these methods struggle to capture device-specific traffic patterns when detecting attack traffic targeting specific IoT devices [1,4]. Inductive learning offers an alternative approach by extracting class vectors from known samples and inferring the categories of unseen samples. This approach enhances classification performance through similarity-based measures and effectively addresses few-shot scenarios. Compared to traditional deep learning methods, inductive learning better handles data imbalance issues and enhances the detection of unknown impactful attacks through similarity awareness [5,6]. However, integrating inductive learning with advanced feature extraction techniques to improve robustness against unknown attacks remains an unresolved research challenge.

Moreover, the introduction of Transformer models has opened new possibilities for cybersecurity. Recently excelling in natural language processing and computer vision, Transformer mechanisms improve multi-head self-attention, effectively capturing long-range dependencies and complex patterns [7]. Zhang et al. combined CNNs and Transformers to learn local and global network traffic features, thereby improving model detection accuracy [8]. While Transformer models exhibit potential in handling complex temporal patterns in network traffic—for instance, the FlowTransformer framework achieves multi-dimensional feature extraction through modular design [7]—their performance significantly degrades in cross-dataset generalization tasks. This is especially evident when detecting new attack types, such as when training a model on the CIC-IDS 2017 dataset and transferring it to the ISCX 2012 dataset [9].

To address these key issues, this paper proposes a novel intrusion detection framework, MFEI-IDS, which is designed to tackle critical challenges in modern cybersecurity. The MFEI-IDS framework combines FCN and Transformer architectures to enhance the detection of unknown attacks by generating robust class vectors. For instance, in real-world environments, it enables the rapid detection of previously unseen network attacks, such

as malicious traffic exploiting zero-day vulnerabilities, using only a few labeled traffic samples. The main contributions of this study are as follows:

1. **Multi-Level Feature Extraction:** A multi-level network feature learning method (FCN–Transformer) is proposed to address network traffic feature extraction challenges. Fully Convolutional Networks (FCNs) are used as input Encoders for Transformers. FCNs effectively capture local spatial patterns in raw network traffic, minimizing reliance on manual feature engineering, while the self-attention mechanism of Transformers models long-range dependencies and complex temporal patterns. This hybrid architecture provides robust multi-scale feature representation, making it well-suited for high-dimensional and sequential network data.
2. **Inductive Learning Module:** An intrusion detection method based on inductive networks is proposed to tackle data imbalance and few-shot unknown attack detection issues. By introducing an inductive learning module equipped with a dynamic routing mechanism, robust class vectors are generated from limited samples, enabling effective classification of unknown attack types. Compared to traditional supervised learning methods, this module supports zero-shot and cross-dataset classification, significantly improving the applicability and generalization ability of the model in scenarios with insufficient labeled data.
3. **Modular and Explainable Design:** By utilizing the explainability of class vector representations and the modular structure of FCN–Transformer, MFEI-IDS provides insights into its decision-making process, addressing common criticisms regarding the lack of explainability in deep learning models.

Experiments on the ISCX 2012 and CIC-IDS 2017 datasets demonstrate that MFEI-IDS outperforms existing state-of-the-art methods in terms of accuracy, F1-score, and detection robustness. Notably, it exhibits stronger generalization in cross-dataset validation and zero-shot classification capabilities, enabling the detection of unknown attack types without retraining.

The remainder of this paper is organized as follows: Section 2 reviews related work. Section 3 elaborates on the system details. Relevant experiments are discussed in Section 4. Section 5 concludes the paper and outlines future research directions.

2. Related Work

In recent years, with the increasing frequency and complexity of cyberattacks, network intrusion detection systems have become a significant research focus in cybersecurity. Researchers have proposed various techniques to meet different intrusion detection needs, primarily including rule-based detection methods, machine learning and deep learning approaches, and few-shot learning techniques. This section primarily reviews the latest research achievements of deep learning and few-shot learning methods in IDS, highlighting the advantages, limitations, and gaps this study addresses.

2.1. Intrusion Detection System Based on Deep Learning

With the proliferation of the internet, network traffic has grown exponentially, and traffic data have become more complex. The challenge of designing a set of features that accurately represent traffic is a critical research issue for network anomaly detection methods due to the limitations of manual feature design. Deep learning, with its powerful feature extraction capabilities, has emerged as a promising solution to address these challenges. By automatically learning representative features directly from raw data, deep learning methods eliminate the reliance on manual feature engineering and provide robust solutions for anomaly detection.

Javaid et al. [5] proposed a self-learning intrusion detection technique that uses a sparse autoencoder to learn feature representations from large amounts of unlabeled data. Wang et al. [2] introduced a CNN-based malicious traffic classification method that transforms network traffic into grayscale images and uses CNNs to learn the intrinsic features of these images. Wei et al. [10] proposed a hierarchical spatiotemporal feature learning (HAST-NAD) method for network anomaly detection, utilizing CNNs to learn local spatial features and LSTMs to learn global temporal features from raw traffic. This approach effectively considers the spatio-temporal properties of network traffic, demonstrating superior classification accuracy and robustness, especially in high-dimensional data and dynamic scenarios. Deng et al. [11] proposed a method combining random forests and LSTMs for traffic anomaly detection. The random forest algorithm calculates feature importance scores to eliminate redundant features, while LSTMs identify abnormal traffic. The accuracy of detecting abnormal traffic in the CIC-IDS-2017 dataset reached 99%, although its generalization ability was limited by data diversity and scale. Alhaj et al. [12] introduced a deep learning-based traffic classification method with attention mechanisms, demonstrating that combining CNNs enhances feature extraction. However, this approach focuses primarily on performance optimization for labeled datasets, with limited exploration of few-shot learning and zero-shot generalization capabilities.

Many researchers introduced Transformers into network intrusion detection as Transformer models gained prominence in large language models and proved effective in capturing complex patterns and relationships in sequential data, including images, graphs, and speech [3,13]. For example, Vaswani et al. [7] first proposed the Transformer model, which excels in capturing global features and long-range dependencies. Luo et al. [13] and Chen et al. [14] explored the applications of Transformers in network traffic analysis, demonstrating their effectiveness in anomaly detection. However, these studies lack investigation into generalization capabilities in few-shot environments and do not leverage the local feature extraction capabilities of CNNs. Zhang et al. [8] introduced a dual-layer network structure combining CNNs and Transformers to learn local and global features, significantly improving detection accuracy and reducing training time. Manocchio et al. [4] proposed the FlowTransformer framework for Transformer-based intrusion detection systems. This framework allows flexible replacement of Transformer components for input encoding, classification, and feature learning, providing a modular approach for intrusion detection. However, most of these methods rely on supervised learning frameworks, failing to effectively address few-shot scenarios and cross-dataset generalization requirements. Additionally, the computational complexity of Transformer models for high-dimensional data remains unresolved.

2.2. Intrusion Detection System Based on Small-Sample Learning

Few-shot learning (FSL) is a technique that enables learning under conditions of limited samples. In recent years, it has garnered attention for its ability to generalize models with minimal data. FSL techniques utilize meta-learning, transfer learning, and other approaches to achieve generalization capabilities under limited data conditions. Typical FSL methods include Matching Networks and Prototypical Networks. In cybersecurity, few-shot learning has been applied to detect unknown attacks. Shu et al. [15] proposed an intrusion detection system based on Prototypical Networks, enabling the detection of unknown attacks by learning from a small number of known attacks. Zhao et al. [16] combined transfer learning and few-shot learning to achieve cross-domain attack detection. However, existing FSL methods face challenges in handling high-dimensional, temporally complex network traffic data. Vinyals et al. [17] introduced Matching Networks, optimizing the matching relationship between support and query samples through metric learning,

thereby improving classification accuracy. However, these methods primarily rely on simple similarity measures (e.g., Euclidean distance or cosine similarity), limiting their effectiveness in handling high-dimensional complex features.

Inductive learning extracts class vectors from known samples, effectively detecting unknown attacks. Metric learning enhances the distinction between different category samples by learning similarity measures. Xu et al. [18] proposed an intrusion detection method combining inductive and metric learning, achieving efficient detection in complex network environments by learning intra-class features and inter-class differences. Li et al. [19] introduced a multi-head self-attention mechanism, improving detection accuracy and robustness. However, existing studies have not fully explored the potential synergy between advanced feature extraction architectures (e.g., FCN and Transformer) and inductive learning, limiting their performance in addressing high-dimensional data and dynamic network environments.

2.3. Summary and Gaps

Despite significant progress in intrusion detection through deep learning and few-shot learning, several limitations persist, creating gaps between existing methods and the requirements of modern network security. Current studies often separate feature extraction and classification, such as feature extraction methods based on CNNs or Transformers that are not deeply integrated with inductive learning. This disconnect limits the performance of models in handling few-shot scenarios and cross-dataset generalization. While deep learning methods excel in automatic feature extraction and classification, their reliance on large, labeled datasets hampers their practicality in real-world scenarios with limited labeled data, especially in dynamic and evolving network environments. Additionally, although Transformer models excel in capturing long-range dependencies and complex patterns, their application in intrusion detection is still in its infancy and predominantly limited to supervised learning paradigms.

Few-shot learning offers an alternative for handling limited data scenarios but faces challenges in dealing with high-dimensional and sequential network traffic. Existing techniques (e.g., Prototypical Networks and Matching Networks) rely on simple similarity measures, which are insufficient for modeling complex relationships in intricate network environments. These methods often fail to detect unknown attack types effectively, particularly when facing heterogeneous traffic data. Furthermore, the computational complexity of Transformer models constrains their applicability in resource-limited environments. While inductive learning demonstrates the potential for enhancing generalization capabilities, its integration with advanced feature extraction architectures like FCN and Transformers remains underexplored, limiting its performance in high-dimensional and dynamic network environments.

To address these challenges, this study proposes the MFEI-IDS framework, which combines FCN–Transformer architecture with an inductive learning module equipped with dynamic routing. Unlike existing methods, MFEI-IDS leverages FCNs to extract local spatial features and Transformers to model global temporal dependencies, providing multi-scale feature representation suitable for high-dimensional and sequential traffic data. The inductive module maps sample features to compact, robust class vectors, effectively generalizing unknown attack types. This approach enhances adaptability through dynamic routing mechanisms, addressing the limitations of existing few-shot learning techniques. MFEI-IDS aims to adapt to few-shot scenarios, heterogeneous data, and dynamic attack patterns without requiring extensive retraining. Its modular design ensures scalability, making it suitable for practical deployment.

3. Method

MFEI-IDS integrates Fully Convolutional Networks (FCN), Transformer architecture, and an inductive learning module to address the challenges of intrusion detection in dynamic and few-shot scenarios. Its design prioritizes efficient feature extraction, robust classification, and adaptability across datasets. This section provides a detailed methodology overview, including architecture, preprocessing, feature extraction, and classification workflows.

3.1. Model Architecture Design

The proposed intrusion detection model incorporates few-shot learning principles using an FCN–Transformer Encoder and inductive network to enhance detection capabilities. As illustrated in Figure 1, the whole architecture consists of three main components: the Encoder, inductive module, and relational module. C represents the number of classes in the dataset, K represents the number of supported samples for each classes, and d represents the dimension of the feature embedding. These components work collaboratively to process raw network traffic, generate robust class vectors, and effectively classify query samples. This section elaborates on the methodology, detailing how input data are processed and utilized throughout the workflow.

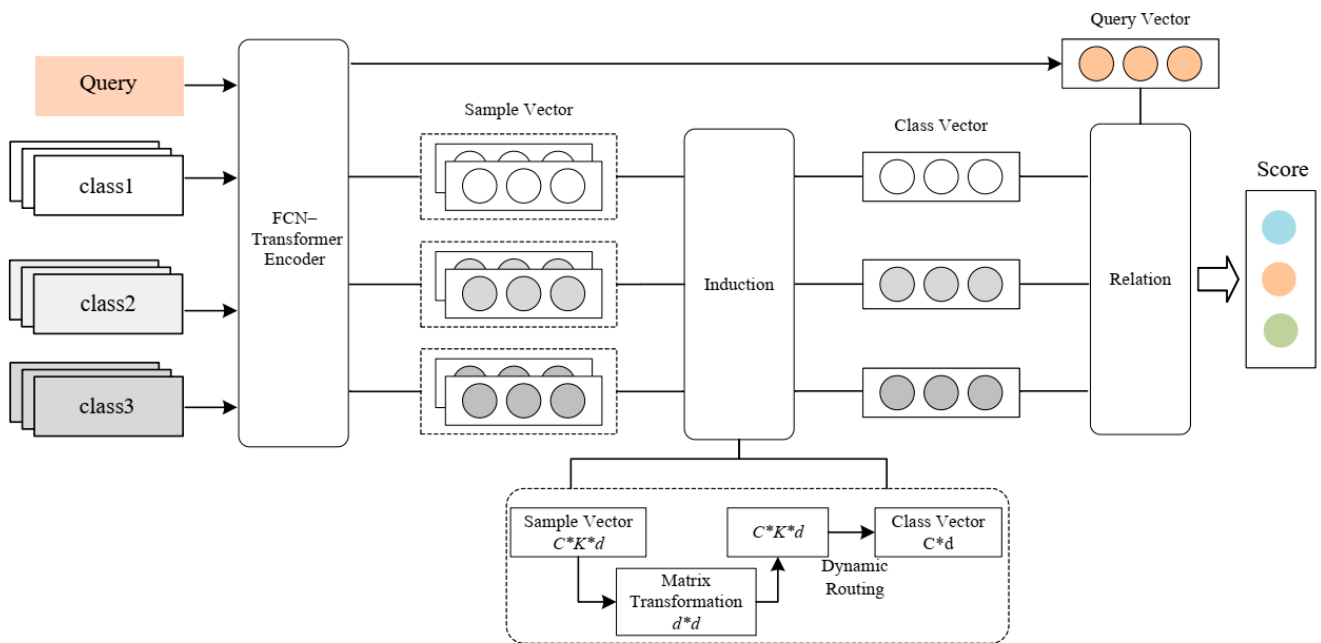


Figure 1. MFEI-IDS structure.

Encoder: The Encoder extracts features from raw network traffic data. This study employs a one-dimensional Fully Convolutional Network (1D FCN) combined with Transformer architecture, referred to as the FCN–Transformer model, to extract local spatial and global temporal features from traffic. The Encoder transforms raw network traffic into high-dimensional feature vectors, forming the basis for subsequent classification and detection.

Inductive Module: The inductive module learns from a small, labeled dataset (support set) to generate class vectors. These class vectors represent the feature space of specific attack types or normal traffic. The model infers the categories of unknown samples through these class vectors. This module utilizes the key principles of few-shot learning by learning features from a few labeled samples to infer new, unseen attack types.

Relational Module: This module, consisting of a simple neural tensor layer, calculates the relationship scores between query samples and class vectors. These scores reflect the similarity between query samples and different categories, determining their classification. Using a similarity-based classification method, this module enables precise traffic classification and detection without relying on many samples.

The model workflow consists of two primary stages:

Training Phase: The Encoder processes labeled session data to extract high-dimensional feature representations, which the inductive module then uses to generate compact class vectors.

Classification Phase: The Encoder processes unlabeled query samples, and the relational module classifies them based on their similarity to class vectors.

3.2. FCN–Transformer Encoder

The primary function of the Encoder is to extract multi-level spatial and temporal features from raw network traffic. This study employs a hybrid architecture combining FCN and Transformer. The Encoder architecture, as shown in Figure 2, consists of four main components: the data preprocessing module, the local spatial feature extraction module, the global temporal feature extraction module, and the output classification module.

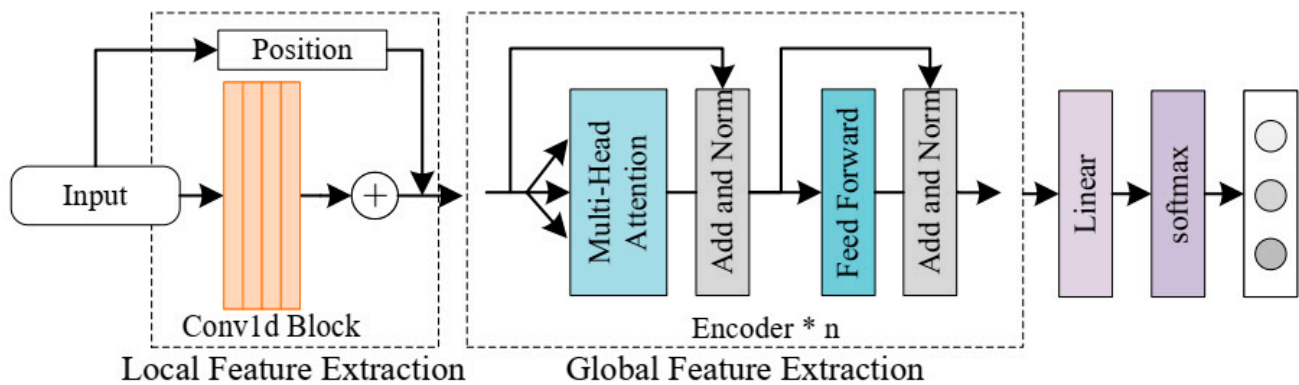


Figure 2. Encoder model architecture.

The data preprocessing module takes raw traffic data in PCAP format as input. Since the raw traffic contains redundant information and network security traffic datasets are typically captured in laboratory environments, this redundant information differs from that in real-world network environments. Therefore, it is necessary to clean this part of the data.

After inputting the data into the model, the model begins processing in the data encoding module, including input and positional encoding. The data encoding step transforms the input sequence into a representation that the model can process while preserving the sequence's information. Positional encoding employs a learnable approach to retain the sequence's positional information.

Next are the local spatial feature extraction module, the global temporal feature extraction module, and the classification module. These two modules form the core of the network. This study does not simply use FCN to extract the local spatial features of packets; instead, it uses a transformer to extract the temporal features of bidirectional traffic flows; as such, an approach processes data sequentially. The FCN network lacks parallel processing capabilities, which limits the parallelism of the Transformer. Instead, this study integrates the FCN network into the Transformer model, placing the FCN network after the input encoding stage. This approach mitigates the shortcomings of both FCN and Transformer and effectively leverages their respective strengths. It enables efficient

learning of spatiotemporal features from network traffic while improving the model's speed and accuracy.

After training, the FCN–Transformer network can map raw traffic to a multi-level spatiotemporal feature vector space. Finally, the learned feature vectors are passed through a fully connected layer and input into the classifier. The classifier uses the softmax activation function to classify the traffic to produce the final intrusion detection results.

The data preprocessing, local spatial feature extraction, and global temporal feature extraction are described below.

3.2.1. Data Preprocessing

The raw traffic is stored in PCAP file format, which is huge in size, so when using it for deep learning model training, the continuous traffic needs to be sliced into multiple discrete units according to a certain granularity.

Currently, there are six commonly used network traffic-slicing methods in network intrusion detection [20,21]: packet, connection, flow, session, service, and host. Researchers need to choose the appropriate data-slicing method according to the research needs, and more data at packet, stream, and session levels are currently used in related research. A packet is the basic data unit transmitted between computer network nodes, including the packet header, load, and activity information. A flow is defined as all packets with the same quintet (source IP, source port, destination IP, destination port, transport layer protocol). A session (a bidirectional flow) is defined as all packets where the source and destination IPs and ports are interchangeable. Packets provide only spatial characteristics of the traffic and cannot identify attacks that rely on sequential information to be detected. In high-speed network environments, the amount of packet-level data is enormous, and computational complexity is high. Suppose the analysis is based only on individual packets. In that case, it may result in different parts of the same session being dispersed into multiple batches, leading to the loss of traffic context. Session-level slicing, on the other hand, can aggregate related data and reduce the risk of misclassification. Stream and session data not only contain spatial characteristics of the traffic but also reflect the temporal information of the traffic. In contrast, the session data eliminate the possible confusion problem of the stream data, so this paper uses the session data as the input sequence of the model.

In raw data processing, this paper draws on the discretization method in Parsaei et al. [22] to internalize continuous features to improve the compatibility and robustness of data representation. To further improve the model's performance, we carry out the data processing flow on the traffic data, as shown in Figure 3. Data preprocessing primarily involves traffic segmentation and cleaning, along with an optional data truncation and padding step.

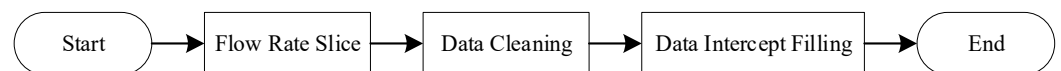


Figure 3. Data pre-processing process.

Traffic Segmentation: Complete traffic data are segmented and aggregated into session data based on the five-tuple information. This process involves dividing the entire traffic data into packets and then aggregating the packets into flow data. Subsequent operations are performed in the form of session data. It is important to note that PCAP format data are inherently binary.

Traffic Cleaning: At the session level, meaningless empty flows and duplicate flows are removed, as they essentially represent noise. Alternatively, these flows can be introduced as noise during training. To ensure the model focuses more on attack traffic patterns

rather than non-attacking IP addresses and to avoid overfitting caused by dataset-specific characteristics, randomly generated new data are used at the packet level to populate fields such as source IP, source port, destination IP, destination port, and MAC address.

Data Truncation and Padding: The proposed FCN–Transformer model can accept variable-length sequences of packets, making this step optional and primarily aimed at reducing training time. Data interception is used to limit the maximum length of packets and sessions, i.e., only the first m bytes of data are kept for each packet, and only the first n packets are kept for each session. At the same time, packets smaller than m bytes can also be padded with 0×00 and all 0×00 packets to bring the number of packets in the session up to n . To reduce the burden of model operations, the maximum length of the input sequence is limited to 200, and the packet length is reserved to 512 bytes.

3.2.2. Local Spatial Feature Extraction

Before learning the local features of the traffic data, the input data are first encoded and processed; this includes Position Embedding and Input Coding, as shown in Figure 4.

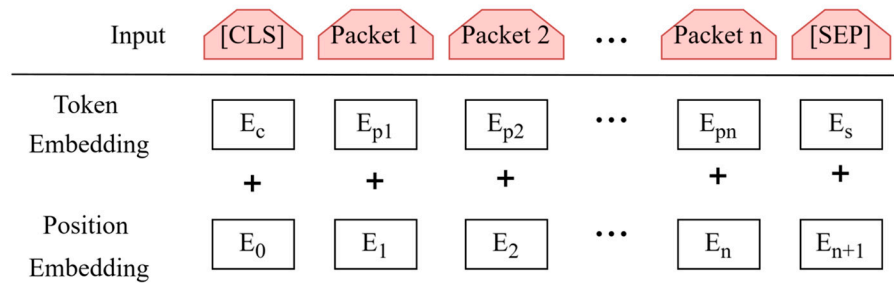


Figure 4. Input representation.

The multi-head self-attention mechanism in Transformer cannot capture the sequential relationships between positions within a text sequence. Compared to methods that process sequence inputs serially, this approach lacks order information for traffic sequences. Positional representation was introduced to address this issue. The original Transformer computes position encodings using a fixed formula, which is unsuitable for representing positions in traffic sessions. This is because packets are highly diverse and their positions are not fixed, as they are mainly related to network protocols and the connection, request, and response methods of application services.

Therefore, the FCN–Transformer model does not use a fixed transformation formula to calculate relative positional encodings. Instead, it adopts the positional encoding method of the BERT model [23], using a standard embedding approach to learn the position of each packet, resulting in position vectors.

The local features of network traffic represent the information carried by individual packets. Hence, local feature extraction networks can replace packet encoding. To extract local spatial features, an optimized one-dimensional fully convolutional neural network (1D-FCN) is used.

Different types of convolutional neural networks (CNNs) are suitable for processing data of different dimensions [24]. One-dimensional convolution kernels are well suited for handling sequential data and are widely used in the field of text processing. Similarly, traffic packets can be viewed as a special type of text. To handle variable-length packet data, this model employs a one-dimensional FCN network.

In CNNs, convolutional layers use multiple convolution kernels to extract features from the input. Each convolution kernel slides over the input using a small window, and the processing of the convolutional layers remains consistent regardless of the input size. Standard CNN networks typically require input sizes to be fixed because they include

Dense or fully connected (Dense) layers. These layers require a fixed input size because, after processing through convolutional and pooling layers, the flattened output must match the input size of the Dense layer. As a result, the input size ultimately determines the feature dimensions passed to the Dense layer. FCN, on the other hand, is a convolutional network without Dense layers, allowing it to handle inputs of varying sizes.

The 1D-FCN used in this study is an improved version of the standard FCN. Based on specific requirements, one-dimensional convolutional blocks (Conv1d Blocks) can be stacked, as illustrated in Figure 5.

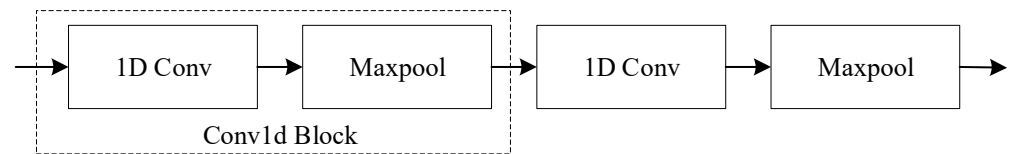


Figure 5. One-dimensional FCN structure diagram.

A convolution block consists of a convolutional layer and a max pooling layer. The computation of a one-dimensional convolution kernel is defined as shown in Equation (1).

$$y_j^l = \sum_{i \in X} y_i^{l-1} \times w_{ij}^l + b_i^l \tag{1}$$

From here, $f(x)$ represents the activation function, X denotes the set of input feature vectors, and y_j^{l-1} is the i feature vector of the output from the $l-1$ convolutional layer and is also the input to the l convolutional neural network. w_{ij}^l is the weight of the convolutional kernel in the l layer, b_i^l is the bias of the l layer, and y_j^l represents the output of the 1 convolutional neural network after convolution with the j kernel.

After position encoding and the 1D FCN local feature extraction network, the embedding vector Y_{FCN} of the data packet and the position encoding P of the session are obtained, and then the embedding vector and position encoding are fused according to Equation (2).

$$Y_L = Y_{FCN} \oplus P \tag{2}$$

3.2.3. Global Temporal Feature Extraction

The global features of network traffic represent the sequential information carried by bidirectional data flows. This paper constructs the global feature extraction module by stacking multiple Transformer Encoders, as shown in Figure 6.

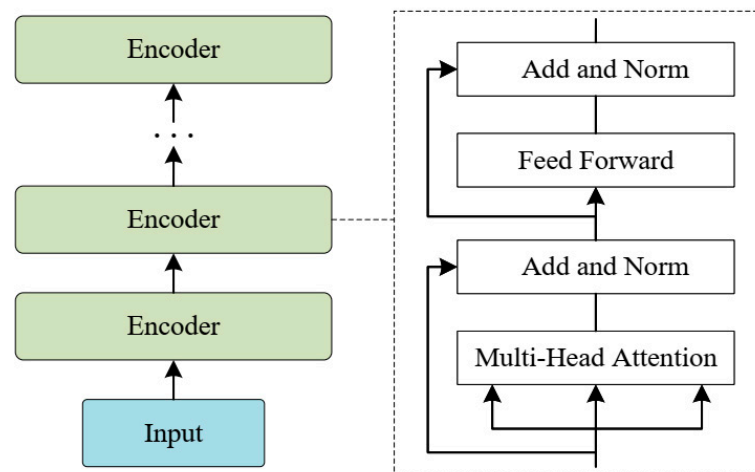


Figure 6. Global feature extraction module structure.

The Transformer Encoder adopts the multi-head self-attention mechanism to address the issue of long-term dependencies. By stacking multiple Encoders in series, it learns the dependencies within long sequences, enabling Transformers to have excellent capabilities in handling ultra-long sequences. This allows them to effectively detect complex network traffic patterns indicative of attacks. The vector Y_L , obtained from the fusion of the packet vector and session position encoding, is processed through the calculations in Equation (3), resulting in the self-attention output $MultiHead(Q, K, V)$.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (3)$$

The model then introduces a residual connection, adding the input before the multi-head self-attention mechanism to the output after it, which helps prevent the vanishing gradient problem in deep networks. Next, LayerNorm [25] is used as the normalization method, which can also speed up training convergence. The output representation obtained after residual joining and normalization is shown in Equation (4).

$$Y_{Norm} = LayerNorm(Y_L + MultiHead(Q, K, V)) \quad (4)$$

The feedforward layer consists of two linear fully connected layers connected by the activation function $ReLU$. The output of the feedforward layer is shown in Equation (5).

$$Y_{Forward} = Linear(ReLU(Linear(Y_{Norm}))) \quad (5)$$

Finally, the residual connection and LayerNorm described above are applied again, yielding the output of the first Transformer Encoder, as shown in Equation (6).

$$Y_{Encoder} = LayerNorm(Y_{Forward} + Y_{Norm}) \quad (6)$$

3.3. Inductive Model

In computer networks, due to different network protocols and environments, there are often different representations for the same category of network traffic. Simply summing or averaging the sample features to represent a category may accumulate interfering information that is not relevant to the classification, thus negatively affecting the final classification performance. The induction module simulates a nonlinear mapping process from sample vectors to category vectors to obtain an embedding vector that best represents the category. Therefore, the induction module is the core of the induction network. In this study, we draw on the insights of Hostiadi et al. [26] on efficient feature selection and utilize the induction model to extract the most representative features from a small number of samples, thus achieving the classification of unknown attack types [27,28]. The inductive module is described in detail below.

Let the sample vectors of the support set obtained after encoding through the Encoder be e^s and the query vectors obtained through encoding through the Encoder be e^q . The induction module exploits the dynamic routing concept of capsule networks to achieve a nonlinear mapping from low-level features to high-level features by iteratively optimizing the relational weights between the samples and the categories. In the induction model, this dynamic routing mechanism is redesigned to generalize the sample representation e_{ij}^s in each category to the representation c_i of the class vector, as shown in Equation (7).

$$\left\{e_{ij}^s \in R^{2u}\right\}_{i=1, \dots, C; j=1, \dots, K} \rightarrow \left\{c_i \in R^{2u}\right\}_{i=1}^C \quad (7)$$

In this process, the sample vectors in the support set are treated as input capsules. After a layer of dynamic routing transformation, the resulting output capsules are regarded as the feature representations of each class, as illustrated in Figure 7.

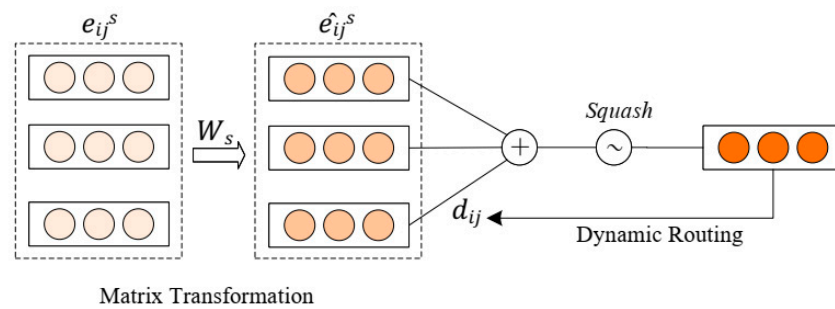


Figure 7. Dynamic routing algorithm of the inductive module.

The specific steps are as follows:

(1) Through matrix transformation, the initial features of all samples are converted from the sample-level semantic space to the category-level semantic space. In the original capsule network, different category sample representations use different transformation matrices, W . However, to support different sizes of C , the model uses the same transformation matrix W_s and bias b_s for all sample vectors in the support set, resulting in the predicted sample vector \hat{e}_{ij}^s , as shown in Equation (8). This allows the model to handle support sets of arbitrary sizes, meaning the model can adapt to any-way, any-shot scenarios.

$$\hat{e}_{ij}^s = \text{squash}(W_s e_{ij}^s + b_s) \tag{8}$$

The squash function is a nonlinear activation function used to normalize the magnitude of a vector, restricting its length within a specific range. This helps reduce noise and enhances the robustness of classification tasks, ensuring that features are represented consistently. Given an input vector x , the computation of the squash function is defined as shown in Equation (9).

$$\text{squash}(x) = \frac{\|x\|^2}{1 + \|x\|^2} \frac{x}{\|x\|} \tag{9}$$

(2) Next, the irrelevant information is filtered through dynamic routing, and a weighted sum of the transformed sample representations is computed to obtain the initial class representation. In each iteration of dynamic routing, the connection coefficients d_i between the upper and lower layers are dynamically adjusted using the softmax operation, ensuring that they sum to 1, as shown in Equation (10).

$$d_i = \text{softmax}(b_i) \tag{10}$$

Here, b_i is the logit value of the connection coefficients, initialized to 0 before the first iteration begins. For a given predicted sample vector \hat{e}_{ij}^s , each candidate class vector \hat{c}_i is the weighted sum of all predicted sample vectors within that candidate class, as shown in Equation (11).

$$\hat{c}_i = \sum_j d_j \hat{e}_{ij}^s \tag{11}$$

Then, the squash function is applied to ensure that the length of each class vector does not exceed 1, as shown in Equation (12).

$$c_i = \text{squash}(\hat{c}_i) \tag{12}$$

(3) Finally, the connection strength is adjusted using a routing protocol. The specific implementation is that when the dot product between the generated candidate class vector and a sample predicted vector is large, the connection strength between them is increased; conversely, it is decreased when the dot product is small, as shown in Equation (13).

$$b_{ij} = b_{ij} + \hat{e}_{ij}^S \cdot c_i \quad (13)$$

By modeling the nonlinear mapping process from sample vectors to category vectors through this dynamic routing mechanism, it effectively filters out interfering information and obtains category features. This method is referred to as dynamic routing induction, as outlined in Algorithm 1. The dynamic routing mechanism optimizes the generation process of category vectors, ensuring that they are highly representative of the features of each category within the support set.

Algorithm 1: Dynamic Routing Induction

Require:

Support Set Sample Vectors: e_{ij}^S

$b_{ij} = 0$

Ensure:

Class Vector: C_i

```

1:  for all samples  $j = 1, \dots, K$  in class  $i$  do
2:       $\hat{e}_{ij}^S = \text{squash}(W_S e_{ij}^S + b_S)$ 
3:  for iter iterations do
4:       $d_i = \text{softmax}(b_i)$ 
5:       $\hat{C}_i = \sum_j d_{ij} \hat{e}_{ij}^S$ 
6:       $C_i = \text{squash}(\hat{C}_i)$ 
7:  for all samples  $j = 1, \dots, K$  in class  $i$  do
8:       $b_{ij} = b_{ij} + \hat{e}_{ij}^S C_i$ 
9:  end for
10: Return  $C_i$ 

```

3.4. Relationship Model

After constructing the data pairs between the query sample and each category in the support set, obtaining the class vectors c_i for each category in the support set, and encoding the query sample into the query vector e^q , the next step is to use the relationship module to assess the correlation between e^q and c_i . The relationship module is a neural tensor layer [29], which models the interaction between each class vector and the query vector pair using a three-dimensional tensor:

$$v(c_i, e^q) = f\left(c_i^T M^{[1:H]} e^q\right) \quad (14)$$

Here, $M^{[1:H]} \in R^{2u \times 2u}$ is a slice of the tensor parameters, and $f(\cdot)$ is the ReLU activation function. The next step is to calculate the relationship score (also referred to as the similarity score) between the class vector and the query vector.

$$r_{iq} = \text{sigmoid}(W_r c(c_i, e^q) + b_r) \quad (15)$$

When evaluating the model, the relationship scores need to be regressed to the true labels, y_q . The induction network transforms the multi-classification problem into a 0–1 classification problem in the classification process. Specifically, it compares the relationship

scores between each query and the class pairs, where the score for matching classes and queries approaches 1, while the scores for non-matching pairs approach 0. Thus, the query vector is classified into the class corresponding to the maximum relationship score. The model is trained using the mean squared loss, and for each episode, given the support set S and query set Q , the objective function is defined as shown in Equation (16).

$$L(S, Q) = \sum_{i=1}^C \sum_{q=1}^n (r_{iq} - 1(y_q == i))^2 \quad (16)$$

After the training is completed, the model will not require fine-tuning when recognizing entirely new categories, as sufficient generalization capability has already been imparted to the model during the meta-training phase. Moreover, this capability will continue to accumulate with each iteration of the model.

4. Experiment

4.1. Dataset

We selected two datasets containing raw traffic in PCAP format, the ISCX 2012 dataset [30] and the CIC-IDS 2017 dataset [31], to conduct the experiments.

The ISCX 2012 dataset, released in 2012, contains traffic data representing seven days of network activity. The dataset includes normal traffic and four types of malicious traffic, totaling 1,520,158 session samples. Most of the data consist of normal traffic, with attack traffic accounting for less than 3%.

The CIC-IDS 2017 dataset, released in 2017, is an intrusion detection and prevention dataset comprising five days of captured data. It contains traffic data for normal behavior and 14 common attack types, with a total of 2,825,404 session samples, of which attack traffic constitutes 19.6%. Some attack types, such as Heartbleed, Infiltration, and SQL Injection, have only 11, 35, and 21 samples, respectively.

Both datasets were collected by the same laboratory using different methodologies and under varying hardware and software environments. The datasets, captured and published in different years, reflect the evolution of network environments to some extent. Over time, network traffic has become more complex, and attack types have grown more diverse, as evident from the differences in the composition of the ISCX 2012 and CIC-IDS 2017 datasets. Conducting experiments on these datasets helps evaluate the model's ability to handle both simple and complex data.

Both datasets were preprocessed using the steps described in Section 3.2.1, which include traffic sessionization, noise removal, and truncation. Session-level slicing was employed to preserve temporal and spatial dependencies, while data truncation ensured consistency and computational feasibility. These preprocessing steps were applied uniformly to all evaluated models, including baseline models such as LSTM, BiGRU, and standard Transformer architectures, to ensure a fair comparison.

To validate the proposed intrusion detection approach, a series of experiments were conducted. The experiments were run on a hardware platform equipped with an Intel processor, GTX 4060 GPU (8 GB VRAM), and Windows 10 operating system. The models were constructed using the PyTorch framework and Transformers library, with GPU acceleration employed for training. The experimental setup is detailed in Table 1.

We used evaluation metrics such as accuracy, precision, recall, and F1-score to evaluate the detection ability of the model. These metrics are calculated based on the confusion matrix, and the binary confusion matrix is shown in Table 2, where each row corresponds to the actual results and each column corresponds to the predicted results. True positives (TPs) are the number of samples correctly predicted as positive and are actually positive. False positives (FPs) are the number of samples incorrectly predicted to be positive or are

actually negative. False negatives (FNs) are the number of samples that were incorrectly predicted to be negative and that were positive. True negative (TN) is the number of samples correctly predicted to be in the negative sample category that were also negative. These metrics provide a comprehensive evaluation of the model's classification performance across different aspects.

Table 1. Experimental environment.

Hardware and Software Experimental Environment	Detail
CPU	Intel(R) Core(TM) i5-12400
Video Card	NVIDIA GeForce GTX 4060 8G
CUDA	11.4.2
RAM	16 GB
Operating System	Windows 10
Programming Languages	Python 3.7
Software Framework	Pytorch = 1.13.1, Transformers = 4.30.2
IDE	PyCharm 2023.3.2

Table 2. Confusion matrix.

Real Sample	Sample Projections	
	Positive Sample	Negative Sample
Positive sample	TP	FN
Negative sample	FP	TN

Accuracy refers to the number of samples correctly predicted by the model as a proportion of the total number of samples. It is one of the most intuitive metrics to assess, but it is not meaningful when the categories are severely imbalanced.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (17)$$

Precision refers to the proportion of samples that are truly positive among those predicted as the positive class by the model. It measures the accuracy of the model in predicting the positive class. Mathematically, it is defined as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (18)$$

Recall refers to the proportion of samples in the true positive category that the model predicts to be in the positive category. It measures the model's ability to find positive category samples.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (19)$$

The F1-score is the harmonic mean of precision and recall, providing a balanced measure that considers both false positives and false negatives. The F1-score ranges from 0 to 1, with higher values indicating better model performance. The formula for calculating the F1-score is as follows:

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (20)$$

4.2. FCN–Transformer Encoder Experiment

4.2.1. Parameter Experiment of FCN Transformer Encoder

We conducted a binary classification experiment on the ISCX 2012 dataset to investigate the impact of the number of Conv1d Blocks, Transformer Encoder units, and self-attention heads on the model’s performance. The objective was to determine the optimal parameter settings for subsequent experiments. During the experiment, all other model parameters were kept constant, except for those being tuned. The initial parameter settings were as follows: the 1D FCN network consisted of two stacked Conv1d Blocks, the number of Encoder units and self-attention heads was set to six, the batch size was 256, and the model was trained for a total of 60 epochs. The learning rate was set to 0.001, and the Adam optimizer was used. The results of the parameter tuning experiments are shown in Figure 8.

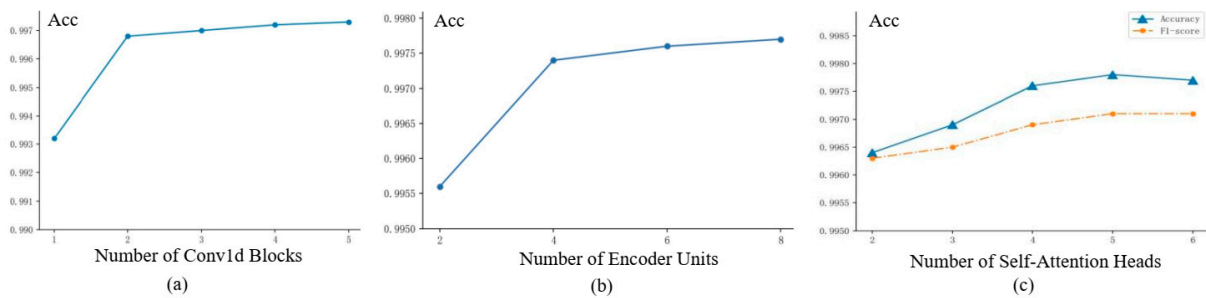


Figure 8. Impact of parameters on model performance: (a) number of Conv1d Blocks; (b) number of Transformer Encoder units; (c) number of self-attention heads.

When optimizing the model parameters, the experimental results reveal that increasing the number of Conv1d Blocks from one to two significantly improves the model’s spatial feature learning capability and accuracy. However, further increases in the number of blocks yield limited accuracy improvements while significantly increasing model complexity.

Due to hardware constraints, the range of Encoder units in the experiments was limited to two to eight. The results show that as the number of units increases, the model’s feature extraction capability improves, but the processing speed decreases.

For the multi-head self-attention mechanism, the experiments demonstrate that increasing the number of heads from two to six initially improves accuracy and steadily enhances the F1-score. However, beyond four heads, the improvement slows down, and accuracy even declines slightly at six heads.

In summary, blindly increasing the number of Conv1d Blocks or self-attention heads in pursuit of higher accuracy is not advisable, as this leads to higher computational complexity and slower model performance. Based on the experimental results, a set of optimal model parameters was determined to balance speed and accuracy, as shown in Table 3.

Table 3. FCN–Transformer model parameter settings.

Parameter	Value
Number of Conv1d Blocks	2
Number of Encoder Units	4
Number of Self-Attention Heads	4
Batch Size	128
Number of Training Iterations	60
Learning Rate	10^{-3}
Optimizer	Adam
Loss Function	CrossEntropy Loss

4.2.2. Comparison of FCN–Transformer with Other Models

To verify the effectiveness of the FCN–Transformer architecture in MFEI-IDS, we compared it with several baseline models widely used in intrusion detection. Based on the HAST-II model described in [10], we constructed the CNN-BiLSTM model and the CNN-BiGRU-Attn model. These, along with the LSTM, BiGRU, and Transformer models, formed the baseline comparison models.

All input data consisted of raw traffic data preprocessed in the same manner. The packet length was retained at 512 bytes, and the session length was set to 200, consistent with the configuration of the proposed model. The number of epochs for the training was set to 60 for all models. The parameters of each model are listed in Table 4, where the Attn suffix indicates the inclusion of an attention mechanism.

Table 4. Comparison of model parameter settings.

Model	Number of Hidden Layers and Units	Dropout	Learning Rate	Batch Size
LSTM	128	0.2	10^{-4}	128
BiGRU	128,128	0.2	10^{-4}	128
CNN-BiLSTM	CNN: 128@2*1, 256@2*1; 192@2*1, 320@2*1 GLobalMaxpool BiLSTM: 92, 92	0.1	10^{-3}	10
CNN-BiGRU-Attn	CNN: 128@2*1, 256@2*1; 192@2*1, 320@2*1 GLobalMaxpool BiGRU: 92, 92 Attention	0.1	10^{-3}	10
Transformer	Token Embedding Position Embedding Encoder: 6 Decoder: 6 Attention Head: 4	0.1	10^{-3}	128

We first conducted a binary classification experiment on the ISCX 2012 dataset, with the results presented in Table 5. From the results, it can be observed that the proposed FCN–Transformer architecture outperforms all other models across all evaluation metrics, demonstrating optimal performance.

Table 5. Binary classification comparison results on ISCX 2012 dataset.

Model	Accuracy	Precision	Recall	F1-Score
LSTM	94.19%	91.86%	95.98%	0.9837
BiGRU	95.06%	93.12%	97.04%	0.9504
CNN-BiLSTM	97.91%	95.16%	98.76%	0.9693
CNN-BiGRU-Attn	98.73%	96.27%	99.18%	0.9770
Transformer	97.47%	96.82%	98.86%	0.9783
FCN–Transformer	99.78%	98.79%	99.90%	0.9934

Among all the models, the two single-level models performed the worst, highlighting that single-level models are incapable of learning the complete multi-level spatiotemporal features of network traffic. In contrast, the dual-layer network models showed significant improvements in all aspects compared to single-level models. The CNN-BiGRU-Attn model outperformed the CNN-BiLSTM model in overall performance, indicating that the

attention mechanism enhances model performance. The Transformer model's performance fell between the CNN-BiLSTM and CNN-BiGRU-Attn models. This demonstrates that while Transformers are powerful, incorporating mechanisms such as attention and dual-layer structures can further boost performance.

We conducted a binary classification experiment on the more complex CIC-IDS 2017 dataset, and the results are shown in Table 6. Compared to the binary classification experiment on the ISCX 2012 dataset, all models experienced a decline in performance due to the increased complexity of the dataset.

Table 6. Results of binary classification comparison experiments on the CIC-IDS 2017 dataset.

Model	Accuracy	Precision	Recall	F1-Score
LSTM	92.59%	90.14%	95.28%	0.9264
BiGRU	93.84%	92.32%	96.05%	0.9415
CNN-BiLSTM	97.05%	94.53%	98.53%	0.9649
CNN-BiGRU-Attn	98.34%	95.85%	99.07%	0.9743
Transformer	97.29%	95.87%	98.32%	0.9708
FCN-Transformer	99.47%	98.52%	99.81%	0.9916

Despite this, the proposed FCN-Transformer model achieved an impressive 99.47% accuracy and an F1-score of 0.9916, demonstrating the best overall performance. These results indicate that the proposed model is highly effective in adapting to the temporal variations in network traffic, showcasing its robustness and ability to handle complex datasets.

Finally, a multi-class classification experiment was conducted on the CIC-IDS 2017 dataset, and the results are shown in Table 7. The proposed FCN-Transformer model also demonstrated the best overall performance in this multi-class classification task, with all metrics exceeding 99%, outperforming the other comparison models. The proposed model achieved optimal results across all evaluation metrics, highlighting its capability to represent the multi-level features of complex traffic effectively. Furthermore, these results suggest that the model is equally applicable to other traffic datasets, exhibiting excellent robustness and adaptability to different network traffic scenarios.

Table 7. Comparison results of different models for multi-categorization on the CIC-IDS2017 dataset.

Model	Accuracy	Precision	Recall	F1-Score
LSTM	91.8%	90.13%	93.37%	0.9172
BiGRU	92.79%	90.26%	94.07%	0.9213
CNN-BiLSTM	95.92%	93.87%	96.94%	0.9538
CNN-BiGRU-Attn	98.55%	95.83%	98.76%	0.9727
Transformer	96.33%	94.57%	96.43%	0.9549
FCN-Transformer	99.32%	99.03%	99.16%	0.9909

To illustrate the performance of FCN-Transformer under complex data, the detection effectiveness of the model for each category was analyzed, as shown in Table 8. It can be observed that the model can achieve a recall of more than 92% and an F1-score value of more than 0.99 for certain attacks, such as DoS Hulk, PortScan, DDoS, and DoS GoldenEye, because these attacks have enough samples to provide the model with training. As for the categories that lack training samples, the model cannot learn the features of these categories, so the detection performance is poorer; for example, attacks such as SQL Injection, XSS, and Infiltration, all of which have less than 50 total sample data, and are even more sparse after being partitioned into training and testing sets, show poorer performance, which illustrates how the unbalanced dataset seriously affects the model's performance. From

another perspective, the model’s recall for Infiltration, SQL Injection, and Heartbleed is not zero, and its precision rates reach 100%. This indicates that the model can detect the data for these attacks. However, the lack of sufficient samples prevents the model from adequately updating its parameters to learn all the representative features of these attacks. This observation demonstrates the powerful feature extraction capability of the FCN–Transformer model, despite the challenges posed by data imbalance.

Table 8. Multi-classification experimental results of FCN–Transformer on the CIC-IDS2017 dataset.

Traffic Category	Precision	Recall	F1-Score
Benign	99.85%	99.54%	0.997
DoS Hulk	99.29%	99.91%	0.996
PortScan	99.93%	99.89%	0.9991
DDoS	99.86%	98.92%	0.9939
DoS GoldenEye	99.12%	99.41%	0.9926
FTP-Patator	99.93%	99.68%	0.995
SSH-Patator	99.83%	98.39%	0.991
DoS slowloris	99.82%	99.23%	0.9953
Dos Slowhttptest	99.27%	99.09%	0.9918
Bot	80.72%	92.9%	0.8638
Brute Force	65.89%	98.75%	0.7904
XSS	97.94%	55.88%	0.7116
Infiltration	100.0%	42.89%	0.6
SQL Injection	100.0%	40%	0.5714
Heartbleed	100.0%	33.33%	0.5

To validate the model’s utilization of the Transformer multi-head self-attention mechanism for parallel computation on sequential data, we conducted a comparison experiment on training and testing times. Table 9 presents the training and testing time comparisons for binary classification on the ISCX 2012 dataset. The proposed model achieved a training time of 54 min and a testing time of 1.7 min on the entire ISCX 2012 dataset, achieving the best performance in both training and testing times. Even when compared to the structurally simpler CNN model, the results underscore the parallelism of the Transformer architecture and demonstrate the efficient processing capability of the model proposed in this chapter.

Table 9. ISCX2012 dataset binary classification training and testing time comparison.

Model	Training Time	Test Time
CNN	57 min	1.9 min
LSTM	76 min	2.7 min
BiGRU	71 min	2.6 min
CNN-BiLSTM	128 min	3.5 min
CNN-BiGRU-Attn	135 min	3.5 min
FCN–Transformer	54 min	1.7 min

Combining the above experimental results, the proposed FCN–Transformer architecture achieved excellent performance on both datasets. It demonstrated the ability to learn highly generalized deep spatiotemporal features of traffic data, outperforming both single models and traditional multi-level CNN-RNN models.

4.3. Analysis of Experimental Results

4.3.1. Comparison with Other Small-Sample Learning Models

To validate the efficiency of the MFEI-IDS model in few-shot learning and evaluate the impact of the number of support set categories and sample sizes on the training results,

we conducted comparative experiments with FC-Net [32], Induction-Network-Attention (IN-Attention) [33], and the proposed few-shot learning method.

In these experiments, IN-Attention employed the same encoding module as our model, but its attention mechanism served as the induction module, whereas the proposed model uses a dynamic routing induction method to construct the induction module. The experiments primarily used the CIC-IDS 2017 dataset, which has a finer-grained division of attack categories, while extracting Infiltration attack data from the ISCX 2012 dataset to expand the test set for this attack type.

The CIC-IDS 2017 dataset consists of 15 data types. However, the Heartbleed and SQL Injection attack categories, with only 11 and 25 samples, respectively, were excluded from the comparison experiments due to insufficient sample size.

For the experiments, the number of support set categories C was set to 2, 3, 5, and 7, with a fixed sample size of 10. Additionally, for each selected class, 20 random samples were chosen as the query set, and the test set size was set to 500 samples. The results are shown in Figure 9. From the experimental results, it can be seen that all three models are a bit sensitive to the number of classes, C . The FC-Net model shows an accuracy trend that initially increases and then decreases as the number of categories increases, achieving its best performance when $C = 5$. Similarly, the proposed model and IN-Attention also achieve optimal performance at $C = 5$. However, the performance for the three-way configuration is slightly lower than for the two-way configuration. These findings suggest that the number of categories in the support set significantly impacts few-shot learning performance and that balancing the number of categories and samples is crucial for optimal model results.

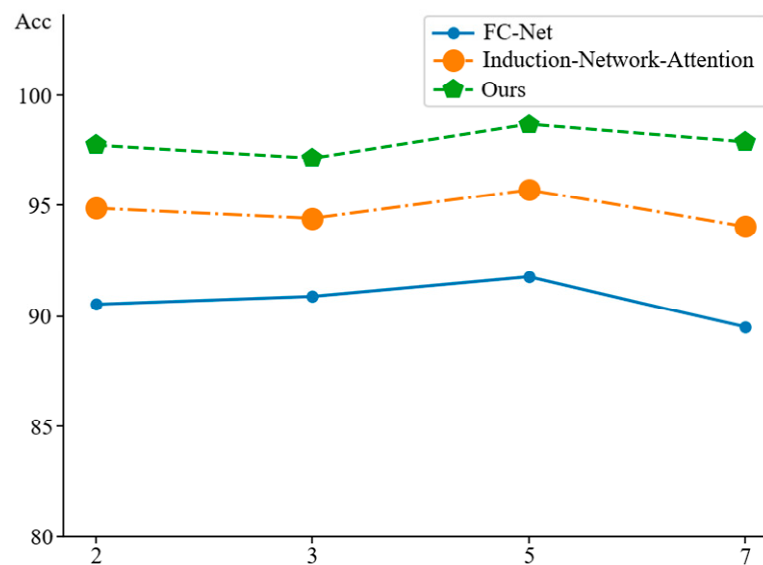


Figure 9. Accuracy–variation in the number of sample categories.

The number of experimental categories was set to five, and the support set sample size K was varied at 5, 10, 15, 20, and 25 to construct six groups for comparative experiments. During each iteration, 20 random samples from the selected classes were additionally chosen as the query set. The results are shown in Figure 10.

The accuracy of all three models improved as the number of samples increased. The improvement was most pronounced when K increased from 5 to 10, as both were in the few-shot learning scenario, but the sample size doubled, enabling the models to better learn the differences between categories.

From the accuracy trends across all models, it can be observed that few-shot learning differs from traditional supervised learning methods that require large sample sizes for

training. Few-shot learning methods exhibit significantly less dependency on sample size, demonstrating their robustness and efficiency in scenarios with limited data.

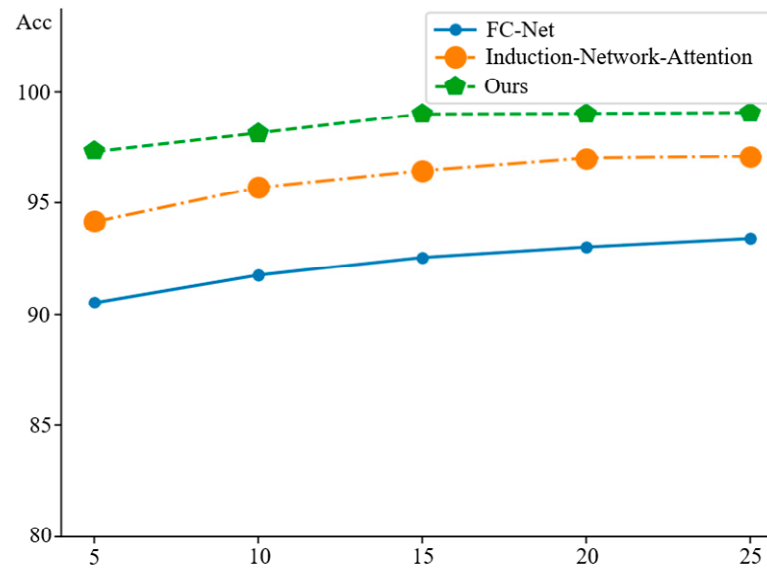


Figure 10. Accuracy–sample number variation.

From the above experiments, it can be observed that the proposed model achieves the best performance, while FC-Net performs the worst. This is because FC-Net uses a three-dimensional convolutional neural network (F-Net) for feature extraction and another CNN to measure sample similarity. While this approach captures differences between samples, it neglects the modeling of the mapping from sample representations to category representations. The IN-Attention model employs an attention mechanism as the induction module. However, since the FCN-Transformer used as the Encoder module has already extracted the temporal information of the traffic, IN-Attention fails to fully utilize its induction capability. Thus, in few-shot learning-based intrusion detection methods, MFEI-IDS demonstrates superior capability in mapping network traffic to attack types, resulting in better attack detection performance.

4.3.2. Comparison with Full Sample Learning

Based on the experimental parameters, the results of the MFEI-IDS model under the five-way 15-shot setting are presented in Table 10. These results are compared with the multi-class classification results on the CIC-IDS 2017 dataset (Table 8).

Table 10. Five-way 15-shot detection results for various attacks.

Traffic Category	Detection Rate	Recall Rate	F1-Score
Benign	93.52%	98.2%	0.958
DoS Hulk	99.39%	97.99%	0.9869
PortScan	97.98%	97.2%	0.9759
DDoS	96.84%	98%	0.9742
DoS GoldenEye	99.19%	97.6%	0.9838
FTP-Patator	99.8%	99.2%	0.995
SSH-Patator	99.2%	99%	0.991
DoS slowloris	98.79%	98.2%	0.985
Dos Slowhttpstest	96.4%	96.4%	0.964
Bot	96.18%	95.8%	0.9599
Brute Force	96.63%	97.4%	0.9701
XSS	99.8%	98.4%	0.9909

In terms of precision, the full-sample method and the few-shot method each demonstrate advantages in different cases. When sufficient data are available, the full-sample supervised learning method generally achieves better recall. However, for categories with only a small number of samples, the MFEI-IDS method consistently achieves better detection performance. This comparison highlights the strength of the MFEI-IDS model in handling imbalanced or limited-sample scenarios, making it particularly effective for detecting rare or infrequently encountered attack types.

4.4. Experimental Results Across Datasets

The ISCX 2012 and CIC-IDS 2017 datasets can both be considered network traffic data generated under different hardware and software environments. While the attack types in these datasets share some similarities, they also exhibit significant differences. Compared to ISCX 2012, the CIC-IDS 2017 dataset contains more diverse traffic types, more complex attack techniques, and a larger volume of attack data, reflecting the evolution of network traffic over time.

Using these two datasets for cross-dataset experiments not only evaluates the proposed model's adaptability to different network environments but also assesses its potential to detect future network attacks. For convenience, the Infiltration, Heartbleed, and SQL Injection categories, which have the fewest samples in CIC-IDS 2017, were excluded from the experiments. For each task, the experimental setup included $C = 3$ (number of categories), $K = 15$ (support set size), and a test set containing 500 samples per class.

The first cross-dataset experiment involved training on the CIC-IDS 2017 dataset and testing on the ISCX 2012 dataset. The results are shown in Table 11.

Table 11. Training at CIC-IDS 2017, testing experimental results at ISCX 2012.

Type of Flow	Detection Rate	Recall Rate	F1-Score
Benign	93.18%	95.6%	0.9437
BFSSH	98.79%	98%	0.9839
Infiltration	96.39%	96.2%	0.963
HttpDos	94.5%	92.8%	0.9364
DDoS	94.81%	95%	0.9491

The results reveal that among all categories, the model achieved the lowest precision for normal traffic, at 93.18%, indicating that a considerable amount of attack traffic was misclassified as normal traffic. This suggests that some attack data were not detected. However, it also demonstrates the model's accuracy in detecting attacks.

The recall for all traffic types exceeded 92%, showing excellent detection capabilities across all attack categories. These results clearly demonstrate the model's strong generalization ability, achieving impressive performance even in cross-dataset scenarios.

Then, there was a cross-dataset training experiment on the ISCX 2012 dataset and a testing experiment on the CIC-IDS 2017 dataset, and the experimental results are shown in Table 12. Compared with the experimental results (Table 11), where both training and testing are performed on the CIC-IDS 2017 dataset, the model's detection performance for each class of attack is slightly degraded, mainly due to the fact that the ISCX 2012 data have fewer classes and the data are slightly simpler, and the model did not fully learn the subtle differences in the mapping of the sample data of the similar attacks to the class space when training on the ISCX 2012 dataset, which resulted in more misclassifications when tested on the more complex CIC-IDS 2017 dataset. Even so, the model still maintains a high detection and accuracy rate.

Table 12. Trained using ISCX 2012, tested using CIC-IDS 2017 experimental results.

Type of Flow	Detection Rate	Recall Rate	F1-Score
Benign	86.02%	97.2%	0.9127
DoS Hulk	97.11%	94.2%	0.9563
PortScan	97.11%	94.2%	0.9563
DDoS	96.41%	96.6%	0.965
DoS GoldenEye	97.74%	95.19%	0.9645
FTP-Patator	98.78%	97.4%	0.9809
SSH-Patator	98.38%	97.19%	0.9778
DoS slowloris	95.53%	98.2%	0.9684
Dos Slowhttptest	95.12%	93.6%	0.9435
Bot	94.02%	91.38%	0.9268
Brute Force	90.94%	96.4%	0.9359
XSS	99.78%	90.4%	0.9486

From the two experiments, it is evident that the MFEI-IDS model maintained high precision and recall rates in both cross-dataset experiments, indicating that cross-dataset training and testing are feasible. These two datasets were collected by the same laboratory under different hardware and software environments using different methodologies. The model's ability to achieve excellent cross-dataset results on these datasets demonstrates that the MFEI-IDS model can be deployed in new environments and future network scenarios without requiring fine-tuning.

This also highlights the model's strong adaptability. Even when encountering data types during testing that were not present during training, the model could still classify them effectively. This suggests that the MFEI-IDS model learns to inductively map sample representations to class representations during training, rather than merely memorizing the features of the training samples.

4.5. Results of Zero-Sample Unknown Intrusion Experiments

In order to test the detection capability of MFEI-IDS for unknown attack detection, the experiment also constructs the experimental dataset, as shown in Table 13. In this experiment, seven types of data were selected from the CIC-IDS2017 dataset, including normal data and six types of attack data, and the test set was expanded with the Infiltration data from the ISCX 2012 dataset. The seven types of data were classified into two categories, one of which was the known traffic set containing normal traffic and four types of attack data. The other is the unknown attack traffic set, consisting of Heartbleed and SQL Injection attack types, which were not used in the model training process. The number of known traffic samples in the training set was 35, where the number of samples in the support set was 15, the number of samples in the query set was 20, and the number of samples in the test set was 100. This setup allowed the model to be evaluated on its ability to detect both known and unknown attacks, emphasizing its potential for generalization to unseen traffic types.

Table 13. Unknown attack detection dataset description.

Data Definitions	Data Types	Training Volume	Test Volume	Code
Known flow	Benign	35	100	iA
	DDoS	35	100	iB
	PortScan	35	100	iC
	DoS Hulk	35	100	iD
	Infiltration	35	100	iE
Unknown attack	Heartbleed	0	10	iF
	SQL Injection	0	25	iG

Table 14 presents the detection results for unknown attacks in a zero-shot scenario. Precision and recall were used as evaluation metrics for unknown attack types because the small sample size of unknown attacks makes other metrics less effective in assessing the model’s detection capabilities.

Table 14. Zero-sample unknown attack detection results.

Code	FC-Net		IN-Attention		MFEI-IDS	
	Detection Rate	Recall Rate	Detection Rate	Recall Rate	Detection Rate	Recall Rate
iA	88.99%	97%	91.67%	99%	95.24%	100%
iB	97.98%	97%	98.98%	97%	99%	99%
iC	90.62%	87%	92.93%	92%	96%	96%
iD	97%	97%	98%	98%	98.99%	98.99%
iE	90%	90%	93.94%	93%	97.94%	95%
iF&iG	93.55%	82.86%	100%	88.57%	100%	94.29%

For the CIC-IDS 2017 dataset, the MFEI-IDS model outperformed other models in detecting all types of attacks, achieving the best results. For the unknown attacks iFiFiF and iGiGiG, the detection rate reached 94.29%. However, the detection of unknown attack types is determined by a relation score threshold, meaning the model can identify unknown attacks but cannot further differentiate between specific unknown attack types.

The precision for unknown attacks reached 100%, indicating that any data predicted as unknown attacks genuinely belong to the unknown attack category. These experiments demonstrate that the proposed model has a high detection rate for unknown attacks and achieves exceptional accuracy in identifying them, highlighting its strong generalization capabilities for unseen attack scenarios.

To further demonstrate the superiority of the MFEI-IDS model, confusion matrices for the three models are presented in Figure 11. From the confusion matrices, it is evident that the MFEI-IDS model achieves a 100% recognition rate for normal traffic, with no instances of normal traffic being misclassified as attack traffic. This is of critical importance in practical applications, as it allows for detecting network attacks without disrupting normal operations.

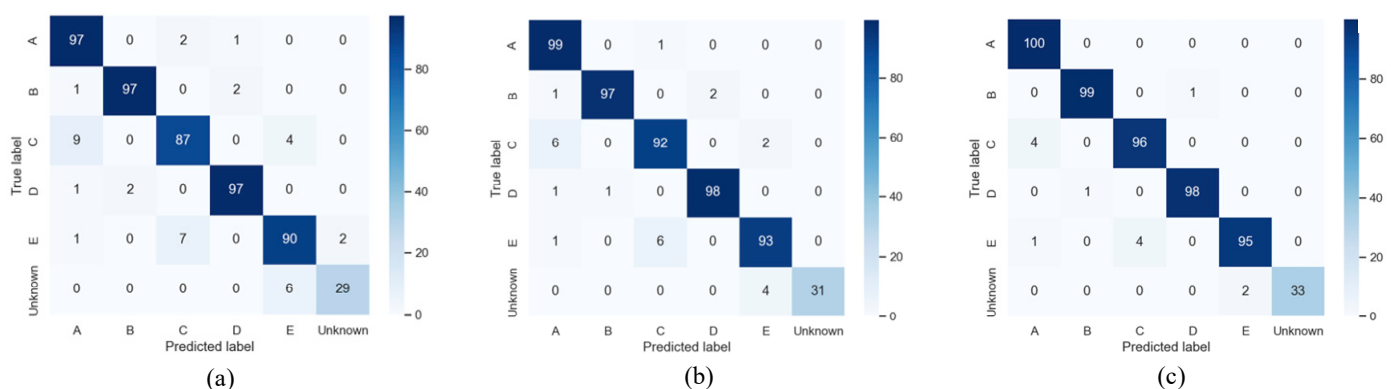


Figure 11. Confusion matrix: (a) FC-Net; (b) Induction-Network-Attention; (c) MFEI-IDS.

In contrast, The FC-Net and IN-Attention models tend to confuse PortScan with Infiltration and Infiltration with unknown attacks. This is because PortScan is part of the reconnaissance phase of an attack, and SQL Injection is inherently a form of web penetration, leading to misclassification between these categories. All three models occasionally

misclassify PortScan data as normal traffic. This is because reconnaissance activities, such as slow scans, may not involve overtly malicious operations and closely resemble normal traffic patterns.

The MFEI-IDS model clearly stands out, achieving the highest detection rate for unknown attacks and having the fewest misclassifications overall. This further highlights the model's exceptional inductive capability in distinguishing between categories, making it the most robust and reliable model among the three.

The above experiments demonstrate the performance and advantages of few-shot learning in detecting unknown attacks. By combining the excellent feature extraction capabilities of the FCN model with the inductive network's ability to map samples to categories, the MFEI-IDS model achieves strong generalization and robustness.

The model surpasses the other two methods in terms of the detection rate and precision for unknown attacks. This enables the proposed model to identify entirely new categories without requiring fine-tuning, making it adaptable to various network environments and future challenges.

5. Conclusions

This paper proposes a Multi-level Feature Extraction and Inductive Network-based Intrusion Detection System (MFEI-IDS), which combines a one-dimensional fully convolutional neural network (1D FCN) and Transformer architecture to achieve efficient feature extraction. The system utilizes an inductive learning module to map sample vectors into robust class representations, enabling effective detection of unknown attacks. The experimental results demonstrate that MFEI-IDS outperforms other models on the ISCX 2012 and CIC-IDS 2017 datasets. For instance, the model achieved an accuracy of 98.5% on the ISCX 2012 dataset and an F1-score of 0.92 on the CIC-IDS 2017 dataset. Notably, the model exhibited outstanding robustness and generalization capability in cross-dataset validation, few-shot learning, and unknown attack scenarios, validating its applicability and potential in complex traffic environments.

Despite its strong performance, MFEI-IDS has some limitations and areas for improvement worth exploring:

- (1) In terms of real-time inference and processing, the current design of MFEI-IDS is primarily tailored to batch processing. This approach requires the completion of data collection, slicing, cleaning, and preprocessing before performing intrusion detection. While effective for offline detection tasks, this batch analysis framework may face limitations in scenarios demanding real-time responses. For instance, existing session-level slicing methods efficiently aggregate contextual traffic information but rely on session completion prior to processing, which can introduce delays in dynamic and high-speed network environments. To overcome these limitations and meet the demands of real-time inference, several optimizations can be explored. One potential direction is the introduction of stream processing frameworks, utilizing sliding windows and stream-level slicing techniques to facilitate the real-time analysis of incomplete sessions, thereby reducing response time. Additionally, optimizing the data preprocessing module by simplifying data cleaning and filling processes can lower the complexity of preprocessing steps. Incorporating dynamic data truncation strategies may further reduce latency. Furthermore, enhancing the model architecture to support lightweight operation and incremental inference is essential. By refining the FCN and Transformer modules, the system can incrementally process incoming packets, achieving efficient real-time inference. In practical applications, the implementation of real-time inference requires a careful balance among model accuracy, response speed, and computational resources. While the current model is primarily designed

for offline detection, optimizing it for real-time inference represents an important avenue for future research, particularly in scenarios that require rapid detection of zero-day attacks and high-frequency traffic anomalies.

- (2) **Further Optimization of the Feature Extraction Model:** The experimental results demonstrate that bidirectional networks, such as BiGRU, outperform more complex models like LSTM in feature extraction, underscoring the importance of comprehensive feature extraction for model performance. This advantage likely arises from the ability of bidirectional networks to simultaneously capture both forward and backward sequence information, thereby providing a more complete representation of temporal dependencies and contextual features. To further optimize the feature extraction process, the introduction of the BERT model as the backbone for temporal feature extraction is a promising direction. BERT, a deep bidirectional model based on the Transformer architecture, is capable of capturing global dependencies in long sequences with high precision. Additionally, its pretraining and fine-tuning mechanisms enable the model to leverage extensive existing datasets, significantly enhancing generalization performance, particularly in few-shot scenarios. For the extraction of local spatial features, an improved WordPiece tokenization method could be employed for traffic packet analysis. Traditional tokenization methods may lead to the loss of fine-grained features within traffic data, whereas an enhanced WordPiece method can more precisely extract local features, such as protocol fields or application-layer data details. By incorporating this method, finer-grained embedding vectors can be constructed, enabling the model to exhibit stronger analytical capabilities when handling complex traffic patterns. Combining the BERT model with an optimized tokenization approach would not only strengthen the extraction of temporal and local features but also fully exploit the multi-scale characteristics of network traffic. This integration would further enhance the model's detection accuracy and robustness. Such an optimization strategy offers a robust theoretical foundation and practical potential for the application of the model in complex traffic environments.
- (3) **Memory Mechanism in Few-Shot Learning:** Current few-shot learning models face the challenge of catastrophic forgetting when switching between tasks in large-scale categorical datasets. Catastrophic forgetting reduces the ability of intrusion detection systems to generalize across tasks and retain prior knowledge, significantly impacting overall performance. This issue leads to instability in key performance metrics, such as accuracy and recall, thereby undermining the reliability of the system. Moreover, the loss of prior knowledge weakens the system's capability to detect unknown attacks, as it struggles to identify anomalies based on previously acquired information. Additionally, catastrophic forgetting increases the need for retraining, which elevates computational costs and decreases adaptability to emerging threats. To address this challenge, memory-enhanced modules based on Transformer architectures could be introduced. While few-shot learning models can effectively detect unknown attacks, they currently lack the capability to further classify these attacks. Future research could explore fine-grained classification mechanisms to improve the model's classification performance and enhance its ability to handle unknown attacks.
- (4) **Intelligent Detection with Knowledge Graphs:** The integration of knowledge graphs represents a critical future direction in network intrusion detection. By constructing semantic networks of attack paths, behavior patterns, and their interrelationships, knowledge graphs provide intuitive logical associations and contextual information for analyzing complex attack behaviors. This approach enables the integration of

multi-source heterogeneous data, uncovering potential attack paths and anomalous patterns, thereby equipping detection systems with more comprehensive analytical capabilities. The application of knowledge graphs in intrusion detection is primarily reflected in three areas. First, they facilitate the integration of attacker behavior characteristics, target system vulnerabilities, and attack techniques, providing a comprehensive understanding of attacks. Second, when combined with inference engines, knowledge graphs enable the real-time evaluation of anomalous traffic and rapid response, allowing for the inference of potential attack types and their consequences. Third, they enhance system explainability by visually presenting the reasoning processes and related cases, thereby improving the credibility of detection results. Nevertheless, constructing and maintaining high-quality knowledge graphs poses significant challenges, such as addressing data sparsity, dynamic changes, and complex entity relationships. Future research could explore automated knowledge graph construction techniques, such as extracting entities and relationships from security reports and logs. Furthermore, the introduction of dynamic updating mechanisms would ensure that knowledge graphs remain consistent with the latest attack patterns. Integrating knowledge graphs into intrusion detection not only enhances detection accuracy and real-time response capabilities but also provides profound insights into complex attack behaviors, offering robust support for combating modern cybersecurity threats.

In summary, MFEI-IDS provides an efficient solution to modern network security challenges through its innovative architecture and feature extraction mechanism. Future research will focus on optimizing feature extraction capabilities, expanding few-shot learning performance, and exploring the application of knowledge graphs in intrusion detection to further advance the field of network security.

Author Contributions: Conceptualization, J.M. and G.Y.; Data curation, B.H.; Formal analysis, X.Y. and Y.L.; Methodology, J.M., X.Y., and G.Y.; Resources, B.H.; Software, J.M., Y.L., and G.Y.; Supervision, B.H.; Validation, X.Y. and G.Y.; Visualization, Y.L.; Writing—original draft, J.M.; Writing—review and editing, B.H., Y.L., and G.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This paper was supported by Kashgar Science and Technology Plan: KS2023024.

Data Availability Statement: Data are contained within the article.

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

References

1. Spadaccino, P.; Cuomo, F. Intrusion detection systems for IoT: Opportunities and challenges offered by edge computing. *ITU J. Future Evol. Technol.* **2022**, *3*, 408–420. [[CrossRef](#)]
2. Wang, Z. Deep Learning-Based Intrusion Detection with Adversaries. *IEEE Access* **2018**, *6*, 38367–38384. [[CrossRef](#)]
3. Ferrag, M.A.; Shu, L.; Yang, X.; Derhab, A.; Maglaras, L. Security for Intelligent Autonomous Networks: A Survey. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 2282–2314. [[CrossRef](#)]
4. Manocchio, A.; Tinnirello, I. FlowTransformer: A Novel Approach for Intrusion Detection in IoT Environments Using Transformers. *IoT Secur. J.* **2021**, *9*, 141–158. [[CrossRef](#)]
5. Javaid, A.; Niyaz, Q.; Sun, W.; Alam, M. A Deep Learning Approach for Network Intrusion Detection System. In Proceedings of the 9th EAI International Conference on Bio-Inspired Information and Communications Technologies (Formerly BIONETICS), New York, NY, USA, 3–5 December 2015; pp. 21–26. [[CrossRef](#)]
6. Shu, J.; Zhang, Y.; Li, X. Few-shot Learning for Intrusion Detection: A Novel Prototype Network-based Approach. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 5036–5048. [[CrossRef](#)]

7. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is All You Need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5998–6008.
8. Zhang, W.; Lu, H.; Zhou, H.; Liu, X. FlowTransformer: A Hybrid CNN and Transformer Model for Network Traffic Analysis. *IEEE Access* **2021**, *9*, 88878–88890. [[CrossRef](#)]
9. Zhang, J.; Wang, H.; Sun, Y. FlowTransformer: A Transformer-based Framework for Network Intrusion Detection. *Appl. Intell.* **2023**, *56*, 2917–2970. [[CrossRef](#)]
10. Wei, G.; Wang, Z. Adoption and realization of deep learning in network traffic anomaly detection device design. *Soft Comput.* **2021**, *25*, 1147–1158. [[CrossRef](#)]
11. Deng, H.; Sun, W.; Luo, H.; Feng, D. Random Forest and LSTM-Based Network Anomaly Detection. *IEEE Access* **2022**, *10*, 22837–22847. [[CrossRef](#)]
12. Alhaj, T.; Al-Hadhrani, A.; Mohammad, R.; Al-Khanjari, Z. Deep Learning and Attention Mechanisms for Traffic Classification: Challenges and Future Directions. *J. Artif. Intell. Soft Comput. Res.* **2023**, *13*, 59–71. [[CrossRef](#)]
13. Luo, S.; Zhao, Z.; Hu, Q.; Liu, Y. A hierarchical CNN-transformer model for network intrusion detection. In Proceedings of the 2nd International Conference on Applied Mathematics, Modelling, and Intelligent Computing (CAMMIC 2022), Kunming, China, 25–27 March 2022. [[CrossRef](#)]
14. Chen, X.; Li, J.; Zhang, H. Transformer-Based Intrusion Detection System for Network Security. *J. Netw. Comput. Appl.* **2023**, *234*, 104527. [[CrossRef](#)]
15. Shu, Z.; Liu, Z.; Shen, C.; Wang, H. Few-shot learning for intrusion detection system. In Proceedings of the 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 10–13 December 2018; pp. 5473–5477.
16. Zhao, Y.; Chen, Y.; Wu, Y.; Zhang, X. Cross-domain few-shot learning for intrusion detection system. *IEEE Access* **2020**, *8*, 176872–176884.
17. Vinyals, O.; Blundell, C.; Lillicrap, T.; Kavukcuoglu, K.; Wierstra, D. Matching Networks for One Shot Learning. In Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS), Barcelona, Spain, 5–10 December 2016; pp. 3630–3638.
18. Xu, Y.; Liu, G.; Hu, Z.; Li, X. Inductive learning and metric learning for network anomaly detection. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 2893–2906.
19. Li, H.; Zhang, T.; Wang, Y.; Gao, S. Attention-based metric learning for network intrusion detection. *IEEE Access* **2021**, *9*, 48965–48978.
20. Dainotti, A.; Pescapé, A.; Claffy, K.C. Issues and future directions in traffic classification. *IEEE Netw.* **2012**, *26*, 35–40. [[CrossRef](#)]
21. Jing, X.; Yan, Z.; Pedrycz, W. Security data collection and data analytics in the Internet: A survey. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 586–618. [[CrossRef](#)]
22. Parsaei, M.; Taheri, R.; Javidan, R. Perusing The Effect of Discretization of Data on Accuracy of Predicting Naïve Bayes Algorithm. *J. Curr. Res. Sci.* **2016**, *1*, 457–462.
23. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Florence, Italy, 28 July–2 August 2019; Association for Computational Linguistics: Minneapolis, MN, USA, 2019; Volume 1 (Long and Short Papers), pp. 4171–4186.
24. Wang, W. Research on Network Traffic Classification and Anomaly Detection Methods Based on Deep Learning. Ph.D. Thesis, University of Science and Technology of China, Anhui, China, 2018.
25. Ba, J.L.; Kiros, J.R.; Hinton, G.E. Layer normalization. *arXiv* **2016**, arXiv:1607.06450.
26. Hostiadi, D.P.; Atmojo, Y.P.; Huizen, R.R.; Susila, I.M.D.; Pradipta, G.A.; Liandana, I.M. A New Approach Feature Selection for Intrusion Detection System Using Correlation Analysis. In Proceedings of the 2022 4th International Conference on Cybernetics and Intelligent System (ICORIS), Prapat, Indonesia, 8–9 October 2022; pp. 1–6. [[CrossRef](#)]
27. Bhuva, D.; Kumar, S. Securing Space Cognitive Communication with Blockchain. In Proceedings of the 2023 IEEE Cognitive Communications for Aerospace Applications Workshop (CCAAW), Cleveland, OH, USA, 20–22 June 2023; pp. 1–6. [[CrossRef](#)]
28. Bhuva, D.; Kumar, S. A novel continuous authentication method using biometrics for IoT devices. *Internet Things* **2023**, *24*, 100927. [[CrossRef](#)]
29. Socher, R.; Chen, D.; Manning, C.D.; Ng, A. Reasoning with neural tensor networks for knowledge base completion. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–8 December 2013; pp. 926–934.
30. Shiravi, A.; Shiravi, H.; Tavallaee, M.; Ghorbani, A.A. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Comput. Secur.* **2012**, *31*, 357–374. [[CrossRef](#)]
31. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In Proceedings of the ICISSP, Funchal, Portugal, 22–24 January 2018; pp. 108–116.

32. Xu, C.; Shen, J.; Du, X. A method of few-shot network intrusion detection based on meta-learning framework. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3540–3552. [[CrossRef](#)]
33. Geng, R.; Li, B.; Li, Y.; Zhu, X.; Jian, P.; Sun, J. Induction Networks for Few-Shot Text Classification. *arXiv* **2019**, arXiv:1902.10482. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.