


Article

A Network Intrusion Detection Method Based on Domain Confusion

Yanze Qu ¹ , Hailong Ma ^{2,*}, Yiming Jiang ² and Youjun Bu ²¹ Institute of Information Technology, PLA Information Engineering University, Zhengzhou 450003, China² National Digital Switching System Engineering and Technological Research Center, Zhengzhou 450001, China

* Correspondence: longmanclear@163.com

Abstract: Network intrusion detection models based on deep learning encounter problems in the migration application. The performance is not as good as expected. In this paper, a network intrusion detection method based on domain confusion is proposed to improve the migration performance of the model. A domain confusion network is designed for feature transformation based on the idea of domain adaptation, mapping the traffic data in different network environments to the same feature space. Meanwhile, a regularizer is proposed to control the information loss in the mapping process to ensure that the transformed feature obtains enough information for intrusion detection. The experiment results show that the detection performance of the model in this paper is similar to or even better than the traditional models, and the migration performance in different network environments is better than the traditional models.

Keywords: network security; deep learning; network intrusion detection; domain adaptation; transfer learning

1. Introduction

With the popularity of network infrastructure and the development of network-related technologies, such as network communication, electronic payment, cloud storage, and the Internet of Things, cyberspace has gradually become another major strategic space. At the same time, the occurrence frequency and harmfulness of malicious network behavior are increasing. How to efficiently maintain the stability and security of the network is one of the important research contents in the area of network security.

Network traffic data can effectively represent the nature of a network behavior. In addition, compared with other network data, network traffic data are easy to access and sufficiently diverse. It is feasible to judge whether network behavior is malicious or not based on network traffic data. In recent years, relevant technologies in the area of machine learning, especially deep learning, have gradually become the mainstream method for constructing a network intrusion detection model. Compared with traditional methods, a network intrusion detection model based on deep learning has advantages in the accuracy and speed of detection and offers broad development prospects. Many relevant studies based on public datasets have achieved a favorable performance [1].

However, the network intrusion detection model based on deep learning also faces some difficulties. SOMMER R et al. pointed out in reference [2] that although many academic studies have proved its good detection performance and development potential, the detection performance of a network intrusion detection model based on deep learning in practical deployment is much lower than expected, which the author attributed to the differences between network security scenarios and other application scenarios of deep learning.

The network environments are heterogeneous. Infrastructure equipment manufacturers, network scope, network users, etc., constitute various network environments, such as the Internet, national network, and enterprise network. We call the data space available for



Citation: Qu, Y.; Ma, H.; Jiang, Y.; Bu, Y. A Network Intrusion Detection Method Based on Domain Confusion. *Electronics* **2023**, *12*, 1255. <https://doi.org/10.3390/electronics12051255>

Academic Editor: Cheng-Chi Lee

Received: 15 February 2023

Revised: 3 March 2023

Accepted: 4 March 2023

Published: 6 March 2023



Copyright: © 2023 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/>).

observation and sampling in a certain network environment the Network Data Domain (NDD) of the network. An important reason why the performance of the same intrusion detection model fluctuates greatly in different networks is that the NDDs of different networks differ in distribution. Deep learning completes various tasks by learning potential patterns in data. Mathematically speaking, a deep learning model mainly completes the fitting of mapping from feature space to output space. In the context of network intrusion detection, an intrusion detection model is trained based on an NDD and has achieved excellent detection performance. When it is migrated to another NDD, its mapping function does not change, but its detection performance declines significantly. This is because different network environments obtain different NDDs.

To this problem, this paper proposes a network intrusion detection method based on domain confusion, which trains a domain confusion network based on Generative Adversarial Networks (GANs) [3]. At the same time, a regularizer is added to cope with the phenomenon of information loss in the process of feature transformation, so as to obtain a feature representation of network traffic data with domain-invariance and a sufficient amount of information. The contribution of this paper can be summarized as follows:

(1) Based on the idea of GANs, the adversarial training method of a domain confusion network is designed, so that the domain confusion network can effectively reduce the influence of the network's environment on network traffic data, so as to close the distance between different NDDs.

(2) Based on the idea of the Auto-Encoder (AE), the regular term of information loss is designed, which can effectively limit the information loss of network traffic data in the process of feature transformation. The regularizer can ensure that the subsequent feature representation obtains enough information for intrusion detection, thus improving the accuracy.

(3) A network intrusion detection method is proposed, which has strong robustness to network environmental noise. It is sensitive to the part of the network traffic data that is conducive to detection, while insensitive to the part related to the environment, providing a feasible solution for improving the migration application and practical deployment of the network intrusion detection model based on deep learning.

2. Related Work

Network intrusion detection is one of the important research topics in the area of network security. In recent years, with the rapid development of deep learning, several excellent works based on deep learning have emerged. In reference [4], BONTEMPS L et al. trained a network intrusion prediction model on the normal traffic data based on Long Short-Term Memory (LSTM) and then judged whether the network environment is in a normal state by comparing the distance between the predicted value of the model and the real value with a threshold value. Kim J et al. constructed a network intrusion detection model based on Multi-Layer Perceptron (MLP), which achieved 99.3% on accuracy and 0.12% on false alarm rate on 10%KDD CUP99 [5]. Reference [6] proposed a network intrusion detection model based on Non-Symmetric Deep Auto-Encoder (NDAE), which achieved 97.85% on accuracy in the five classification tasks of NSL-KDD. In addition, there are many studies with an outstanding performance carried out on CTU-13, UNSW-NB15, UGR'16, etc. [1]. Each of these datasets has its own purpose. For example, UNSW-NB15 focuses on packet content analysis and is suitable for identifying encrypted traffic. The success of network intrusion detection in various datasets also further proves its advantages.

The performance of a network intrusion detection model based on deep learning is relatively mature, but it is also faced with some problems. Reference [7] conducted a study of 30 papers from top-tier security conferences within the past decade and analyzed the pitfalls existing in the steps of constructing a security system based on deep learning, pointing out that most of the studies did not take the limitations of actual conditions into account, and the evaluation experiments were ideal. The problem mentioned in reference [2] that the network intrusion detection models developed a striking gap in terms

of actual deployments has not been solved. The research on the migration application of a network intrusion detection model is slightly insufficient.

In the classic scenario of deep learning, the data distribution during application is often assumed to be the same as that of training. However, in the actual scenario, the application environment is often different from the training one, which leads to the degradation of the model. In order to solve this problem, transfer learning comes into being, the core of which is to find the relationship between existing and new knowledge. In transfer learning, the data environment used for training is usually called the source domain, and the data environment for the application is called the target domain. Domain Adaptation (DA) is a method of transfer learning, which maps the features of different domains into the same space, thus reducing the differences between the source domain and the target domain [8] and finally obtaining the feature representation with domain-invariance.

GANs provide a method to effectively learn feature representation from samples without massive annotated data [9]. It achieves this goal by updating two adversarial neural networks through the back propagation algorithm. At present, the idea of generative adversarial learning has achieved refreshing achievements in many areas, such as image generation [10,11] and video generation [12]. In reference [13], the adversarial method is introduced into the study of DA, and Domain Adversarial Neural Networks (DANNs) were proposed. The authors added the domain error as a regularizer into the loss function and completed training with a gradient reversal layer, achieving a good migration effect in the area of image recognition. However, this work did not really involve GANs into training. Combining the above works, a domain confusion network is designed to generate the feature representation with domain-invariance.

The Auto-Encoder [14] is also one of the generation models. Its training goal is to ensure that the output of the model is as similar as possible to the input, obtaining meaningful feature representation by imposing constraints on the intermediate layers of the neural network, such as limiting the number of perceptrons or the sparsity. If the output and input of the model are close enough, the vector in the middle layer is considered as a feature representation that can effectively represent the original data sample. This algorithm inspired the design of the regularizer in this paper.

3. Method

3.1. Problem Modeling

The construction of network intrusion detection model based on deep learning requires the support of massive datasets with well-annotated labels. Deep learning is a process of approximating complex function through a large amount of data. The amount of training datasets and the accuracy of annotation play an important role in determining the performance of the model. The construction of qualified network security datasets is costly and requires a long development period. Training an intrusion detection model with high robustness to NDDs' transformation is a more valuable and promising method.

There is an NDD D_S from which a massive network dataset with well-annotated labels T_S can be obtained, corresponding to the training space for network intrusion detection model. There is an NDD D_T from which a massive network dataset without labels T_T can be acquired by tools, such as Wireshark, corresponding to the deployment environment of network intrusion detection model. A robust network intrusion detection model should pay enough attention to the part of the data that is conducive to classification, while neglecting other parts, such as environmental noise.

Network traffic data are one of the most commonly used network security data to construct network intrusion detection models. The experiment of this paper is based on it, and it is related to the network environment to a certain extent. Take network protocols as an example; the usage of network protocols varies in different network environments. The usage of encryption protocols on the Internet is usually higher than that on the Intranet. It is difficult to determine the correlation function between network traffic features and network environment. Traditional methods cannot effectively solve this problem. It is a feasible

scheme to transform network traffic features with deep learning, compressing environment-related information and obtaining feature representation with domain-invariance.

3.2. Domain Confusion Network

The network intrusion detection model based on domain confusion consists of two parts: domain confusion network, a feature extractor responsible for feature transformation, and a classifier responsible for identifying malicious network behaviors. The domain confusion network is responsible for mapping the traffic data of different NDDs into the same feature space and retaining enough information for the classification work of the classifier to ensure the classification accuracy.

Figure 1 shows the training process of the domain confusion model. In the figure above, two neural networks appear, namely Network Domain Confusion Network and Network Domain Discriminator. To map the traffic data of different NDDs into the same space, the training process of the two models is adversarial to some extent.

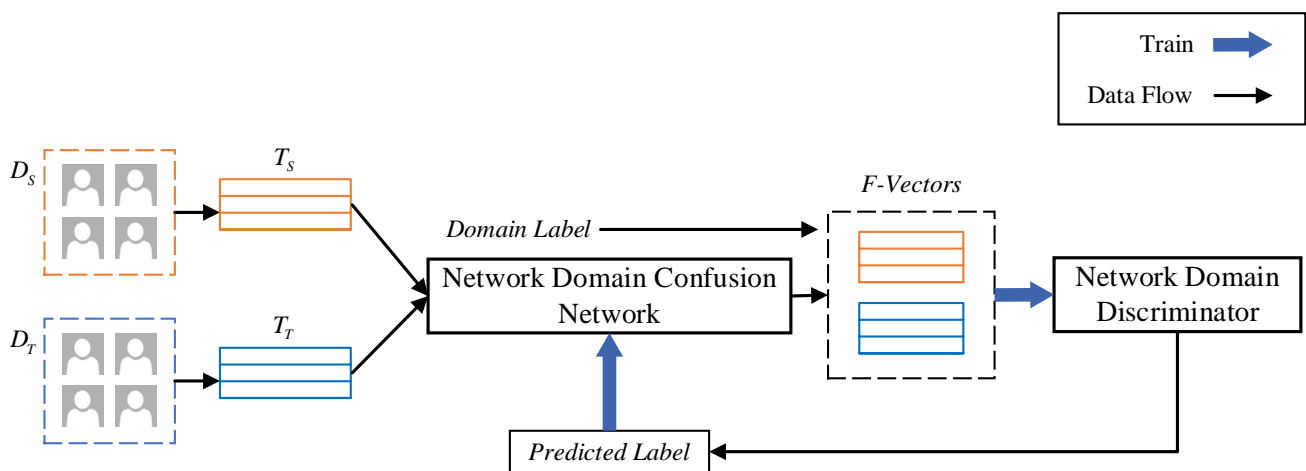


Figure 1. The Method to Train a Network Domain Confusion Model.

Domain confusion network maps network traffic data of D_S and D_T into F – Vectors to obtain their corresponding feature representation, as shown in Equation (1).

$$V_s, V_T = N_C(T_s, T_T; \theta_C) \quad (1)$$

The task of domain discriminator is to identify the NDD to where the feature representation V_* belongs. The output of domain discriminator is a scalar $N_D(V_*; \theta_D)$, representing the probability of feature representation V_* belonging to D_S . In the training process, we completed the training of domain discriminator by minimizing classification errors, as shown in Equation (2). On the contrary, domain confusion network was trained to maximize the classification error.

$$\min[\log(N_D(V_T; \theta_D)) - \log(N_D(V_S; \theta_D))] \quad (2)$$

Binary Cross Entropy (BCE) is often used as a loss function for binary classification problems, and its expression is shown in Equation (3). In the Equation, P represents the output of the classification model, and Y is the expected result. When training is processed in batches, y_i represents the category of the i -th sample, and $p(y_i)$ represents the i -th output of the model. With 0 indicating that the sample belongs to D_S and 1 indicating that the sample belongs to D_T , the loss functions of domain discrimination network and domain confusion network can be derived as Equations (4) and (5).

$$BCE_{Loss} = BCE(P, Y) = -\frac{1}{N} \sum_{i=1}^N [y_i \log(p(y_i)) + (1 - y_i) \log(1 - p(y_i))] \quad (3)$$

$$Loss_D = \frac{1}{2} [BCE(N_C(T_S; \theta_C), [0, \dots, 0]) + BCE(N_C(T_T; \theta_C), [1, \dots, 1])] \quad (4)$$

$$Loss_C = BCE(N_D(N_C(T_T; \theta_C); \theta_D), [0, \dots, 0]) \quad (5)$$

In the above training process, the samples of different NDDs are first mapped to the same feature space through domain confusion network, and then domain discriminator is trained based on this batch of data, so that it can effectively identify the feature representation of the traffic samples of NDDs. After this training, domain confusion network is trained based on the outputs of the domain discriminator, making the feature representation generated by domain confusion network more easily recognized by domain discrimination network as coming from D_S . After several rounds of the above steps, the loss value of the two neural networks will level off. At this point, the system can be considered to be in Nash equilibrium. In $F - Vectors$, the feature representation of the D_T 's samples will be closer to those of the D_S 's samples. Ideally, domain discriminator cannot complete the classification task. In this case, domain confusion network can be used to generate a feature representation that is independent of network environment for traffic data. With the output of domain confusion network as points in the feature space and domain discrimination network as a hyperplane, the training process can be shown in Figure 2.

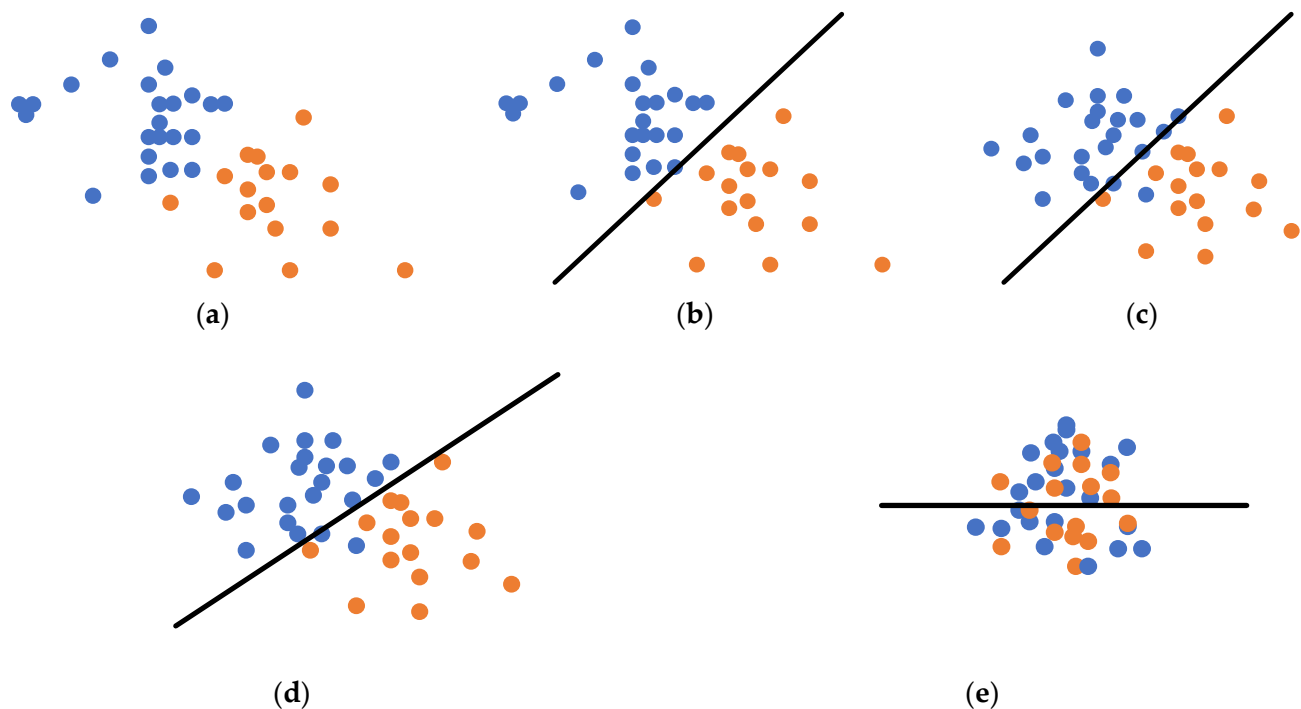


Figure 2. The Training Process. Traffic samples are mapped to the points in the feature space as (a), in which the orange ones represent these samples of source domain and the blue ones represent those of target domain. The discriminator is trained based on the dataset as (b). The parameters in domain confusion network are adjusted to make the points closer as (c). Then, the discriminator needs to be updated as (d). After enough iterations, as (e) shows, the discriminator cannot complete its task smoothly, and the probability that one point comes from source domain is about 0.5.

3.3. Information Loss Regularizer

When the above adversarial system reaches the equilibrium state, domain discriminator cannot accurately distinguish the feature representation of samples in different NDDs. It can be considered that the part related to network environment is greatly reduced, and the feature representation obtains domain-invariance. However, in the process of feature transformation, the information carried by samples is decreased. In addition to the expected reduction in the relevant part of network environment, the information of other parts is also

inevitably affected, thus affecting the performance of the model. In order to deal with this problem, it is necessary to add information loss regularizer to the loss function of domain confusion network, so as to ensure that the subsequent feature representation has enough information to judge whether the network behavior is malicious or not.

Considering the design of information loss regularizer, domain confusion network was designed based on the idea of Auto-Encoder as shown in Figure 3. Mean Square Error (MSE) was used to measure information loss, as shown in Equation (6).

$$R = MSE(X, Y) = \frac{1}{N} \sum_{i=1}^N (Y - X)^2 \quad (6)$$

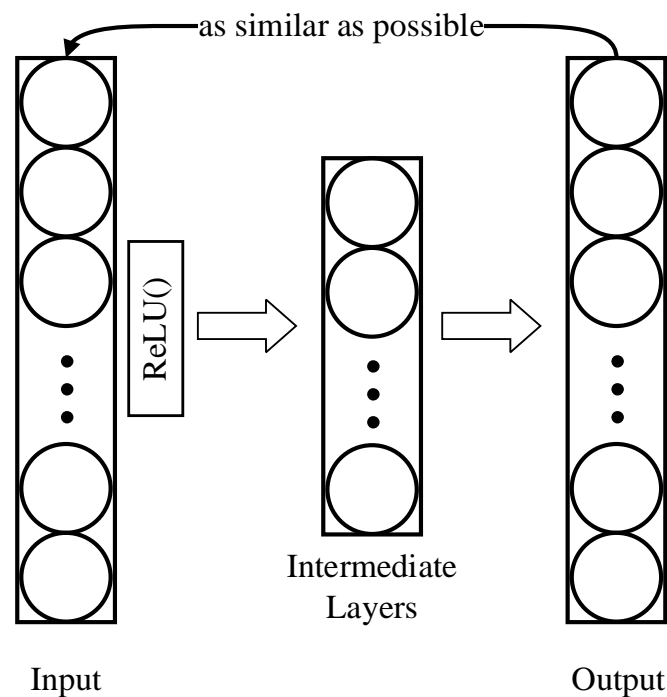


Figure 3. The Structure of Domain Confusion Network.

The final loss function of domain confusion network is shown in Equation (7). λ is a regularizer coefficient. If λ is too large, the model pays too much attention to the information loss during the feature transformation, making the model difficult to reach convergence.

$$Loss_C = BCE(N_D(N_C(T_T; \theta_C); \theta_D), [0, \dots, 0]) + \lambda MSE(T_T, N_C(T_T; \theta_C)) \quad (7)$$

The pseudocode of the training process is shown in Algorithm 1.

Algorithm 1 The Training Process of Domain Confusion Network

```

def get_DCN():
    # Load Data
    data_Source = get_Data(source)
    data_Target = get_Data(target)
    # Initialize the Neural Network
    DCN = base_network()
    Discriminator = base_classifier()
    # Multiple Epoches
    for epoch in range(num_epoch):
        # Multiple Batches
        for batch_Source, batch_Target in data_Source, data_Target:
            # Obtain Feature Representation
            F_Source, F_Target = DCN(batch_Source), DCN(batch_Target)
            # Train the Discriminator
            Pred_Source = Discriminator(F_Source)
            Pred_Target = Discriminator(F_Target)
            Preds = [Pred_Source, Pred_Target]
            Expecteds = [0 in len(Pred_Source), 1 in len(Pred_Target)]
            ## Loss of the Discriminator
            Loss_Discriminator = BCE(Preds, Expecteds)
            Loss_Discriminator.backward()
            ## Update the Discriminator
            Discriminator.weights.update()
            # Train the Domain Confusion Network
            Preds_DCN = Discriminator(F_Target)
            Expecteds_DCN = [0 in len(Pred_Target)]
            ## Loss of the Domain Confusion Network
            Loss_DCN = BCE(Preds_DCN, Expecteds_DCN) +  $\lambda$ *MSE(F_Target, batch_Target)
            ## Update the Domain Confusion Network
            DCN.weights.update()
        # Obtain the DCN
    return DCN

```

4. Experiment*4.1. Experiment Setup*

As mentioned above, the network intrusion detection model based on domain confusion requires two NDDs for construction. The experiment was carried out based on CICIDS2017 and CICIDS2018 [15], which were developed by the Communications Security Establishment (CSE) in collaboration with the Canadian Institute for Cybersecurity (CIC) for evaluating the detection performance of an intrusion detection system. In reference [15], the team evaluated eleven public datasets since 1998 and found that most of them were out of date, and some of the datasets suffered from low reliability, lack of diversity, and low volume. The CICIDS datasets followed eleven criteria for constructing baseline dataset proposed in reference [16], including benign network behaviors and most up-to-date attack behaviors, in which flow was taken as the basic unit. According to RFC3917 [17], a network flow is defined as a set of packets that have the same quintuple (source IP address, destination IP address, source port, destination port, and protocol number) within a certain period of time.

Another decisive reason for choosing CICIDS datasets as the baseline is the similarity between CICIDS2017 and CICIDS2018. The two datasets were obtained with the same form by the development team in 2017 and 2018, respectively, for similar purposes using similar configurations, which provided convenience for the experiment and eliminated the work of feature alignment. Due to the complexity of network data, datasets constructed by different research groups often have different purposes and use different collection configurations and different forms, which greatly weakens the comparability among public

datasets. Due to the privacy of network data, many public datasets do not provide complete traffic data, but only provide partial observations, which makes it difficult to carry out feature alignment. For example, KDD CUP'99 is collected from military networks and provides TCP connection features, time-based features, and host state features for different attacks. The similarities between CICIDS2017 and CICIDS2018 in terms of temporal and spatial characteristics and collection configuration provide necessary conditions for the implementation of the evaluation experiment.

Ten percent of CICIDS datasets was randomly selected for training, and then it was divided into a training set and a test set in a ratio of 4:1. Based on CICFlowMeter [18], the corresponding features were extracted from the traffic of a network flow. The shapes of each dataset in the experiment are shown in Table 1.

Table 1. The Shape of Experiment Datasets.

| Name | Shape |
|------------------|----------------|
| CICIDS2017_Train | [226, 231, 77] |
| CICIDS2018_Train | [659, 831, 77] |
| CICIDS2017_Test | [56, 557, 77] |
| CICIDS2018_Test | [164, 957, 77] |

The experiment took binary classification as the main task, CICIDS2017 as the traffic data of D_S , and CICIDS2018 as the traffic data of D_T . The traffic data of D_T were involved in the training process as unlabeled data. The parameter settings are shown in Table 2. How to select the value of λ will be discussed below.

Table 2. The Parameters During Experiments.

| Parameter | Value |
|-----------------------------------|--------|
| Optimizer | Adam |
| Learning Rate | 0.0001 |
| Batch Size | 128 |
| Epoch Number | 50 |
| Regularizer Coefficient λ | 0.1 |

It was necessary to process the data before the training. The pre-processing pseudocode is shown as Algorithm 2.

Algorithm 2 Data Pre-Processing

```
def pre_processing(data):
    # Load Data
    data = get_Data(dataset)
    # Feature Selection via CICFlowMeter with its default setting
    data = feature_selection_via_CICFlowMeter(data)
    # Data Cleaning, Drop Samples with Null
    data = clean_Data(data)
    # Obtain Feature and Label from Data
    features, labels = data.split()
    # Normalize Feature
    features = StandardScaler(features)
    # Label Encoding
    labels = LabelMapper(labels)
    # Obtain Data after Pre-processing
    data = [features, labels]
```

4.2. Domain Confusion Model

The experimental dataset is tabular data composed of 77 dimensional feature vectors. For tabular data, multi-layer perceptron has a high processing efficiency [5]. The domain confusion model was obtained based on the training method and the loss function Equation (7) proposed above. The changes in loss with different λ during training are shown in Figures 4–9.

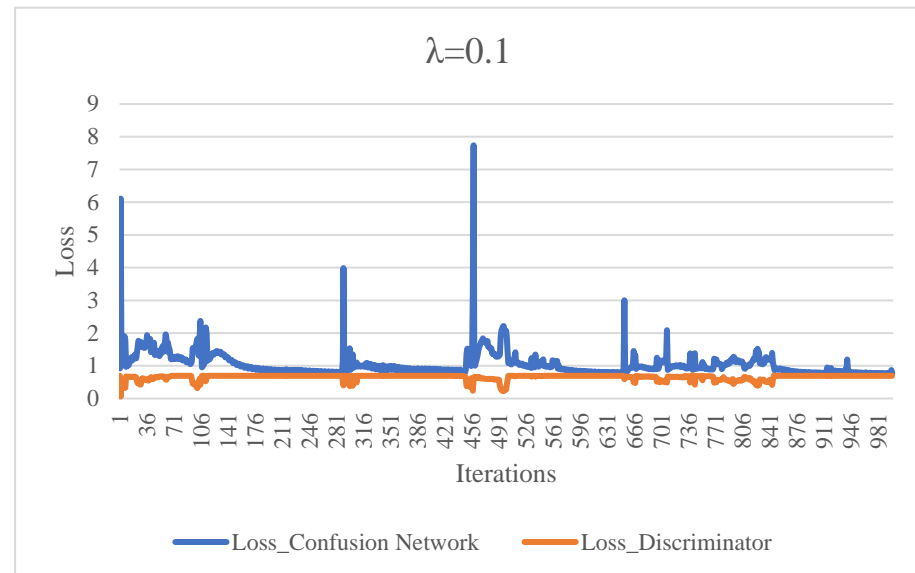


Figure 4. The Loss During Training ($\lambda = 0.1$).

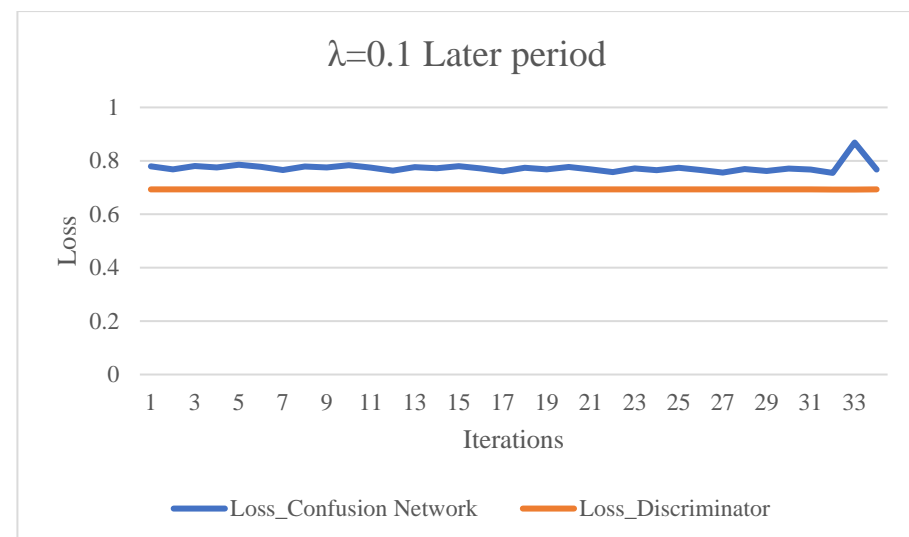


Figure 5. The Loss at The End of Training ($\lambda = 0.1$).

λ is the regularier coefficient in Equation (7), which to a certain extent represents the model's attention to information loss. Figures 4, 6 and 8 show the changes in loss of the domain confusion network and domain discrimination network in the whole training stage under different regularity coefficients. The blue line represents the loss of the domain confusion network, and the orange one represents the loss of the domain discrimination network. With the increase in iteration rounds, the loss of the two networks gradually becomes stable, which can be considered that the adversarial system reaches the equilibrium, and the training of domain confusion model is completed.

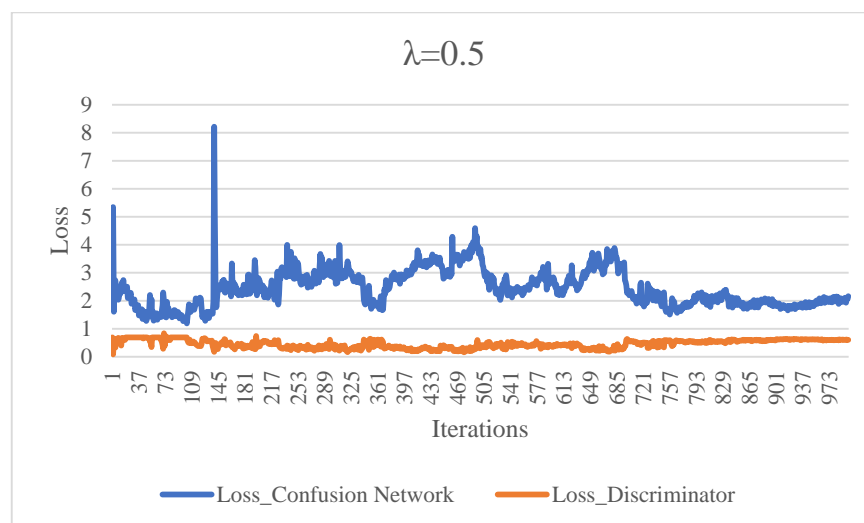


Figure 6. The Loss During Training ($\lambda = 0.5$).

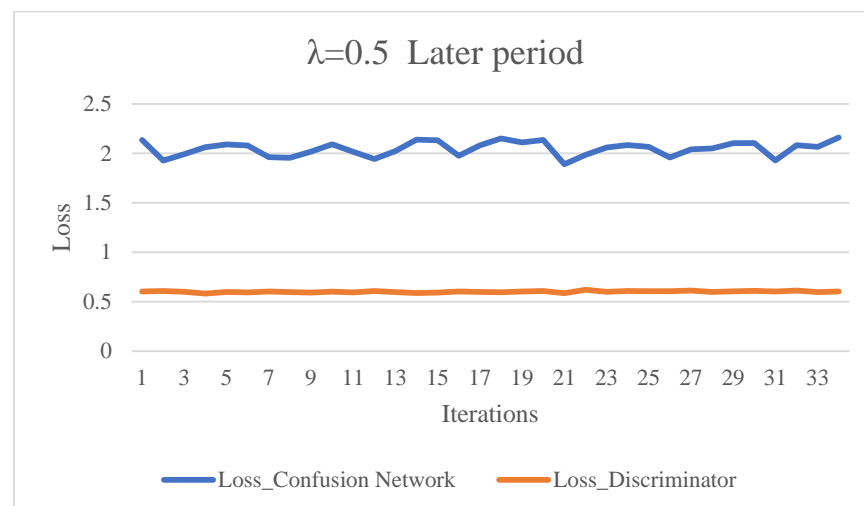


Figure 7. The Loss at The End of Training ($\lambda = 0.5$).

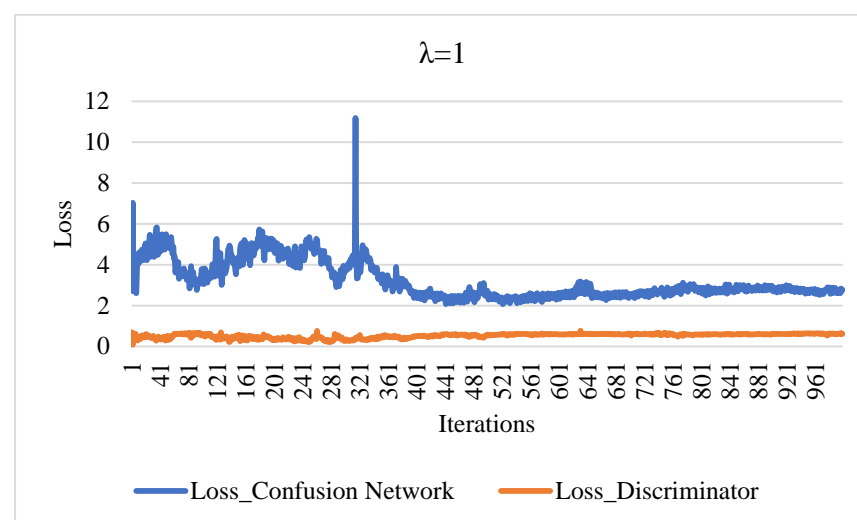


Figure 8. The Loss During Training ($\lambda = 1$).

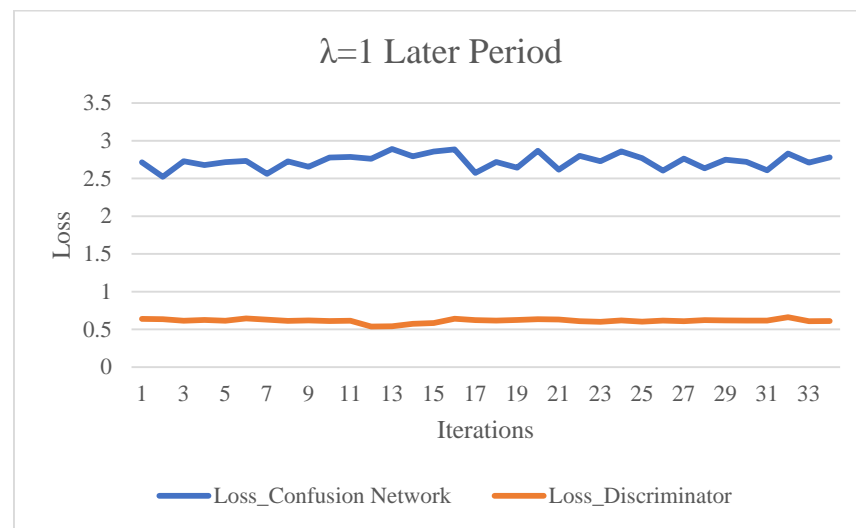


Figure 9. The Loss at The End of Training ($\lambda = 1$).

In addition, it can be seen from the above figures that the loss of the domain discrimination network reaches convergence relatively early. The training direction of the domain confusion network in the later period is mainly determined by the regularizer of information loss. In the case of sufficient iteration rounds, a smaller λ ($\lambda \neq 0$) can also be effective in accomplishing its tasks.

Figures 5, 7 and 9 show the fluctuation of losses under different regularity coefficients in the equilibrium state. The average loss of the domain confusion network and the domain discrimination network under different regularity coefficients in the equilibrium stage is shown in Table 3. The loss of the domain confusion network is proportional to λ , but the loss of domain discrimination is not, making it an effective reference for system stability.

Table 3. The Average Losses of Different λ .

| λ | Confusion Network Loss | Discriminator Loss |
|-----------|------------------------|--------------------|
| 0.1 | 0.7736 | 0.6931 |
| 0.5 | 2.7282 | 0.6143 |
| 1.0 | 2.0472 | 0.6012 |

The goal of the domain confusion network is that the domain discrimination network cannot accurately classify its output. The expected value given by the domain discrimination network is 50%, and the corresponding loss value is 0.6931. The closer the loss of the domain discrimination network is to 0.6931, the stronger the domain-invariance of the feature representation. Therefore, $\lambda = 0.1$ is preferred.

4.3. Migration Application

After the training of the domain confusion network, the network intrusion detection model CN-MLPM was constructed based on D_S with the domain confusion model as a feature extractor. The baseline models MLPM and 1D-CNNM were constructed based on the method in reference [5,19]. The evaluation experiment was conducted on the test set of D_S and D_T . The experiment results are shown in Tables 4–6.

Table 4. The Loss During Training.

| Model | Training Loss | Validation Loss | Validation Accuracy |
|---------|---------------|-----------------|---------------------|
| MLPM | 0.092 | 0.073 | 96.91% |
| 1D-CNNM | 0.110 | 0.070 | 96.76% |
| CN-MLPM | 0.105 | 0.088 | 95.14% |

Table 5. The Evaluation Results on CICIDS2017_Test.

| Model | Accuracy | Precision | Recall | F1-Score |
|---------|----------|-----------|--------|----------|
| MLPM | 96.13% | 99.45% | 91.94% | 0.9554 |
| 1D-CNNM | 92.51% | 98.13% | 91.03% | 0.9444 |
| CN-MLPM | 96.15% | 99.08% | 92.92% | 0.9590 |

Table 6. The Evaluation Results on CICIDS2018_Test.

| Model | Accuracy | Precision | Recall | F1-Score |
|---------|----------|-----------|--------|----------|
| MLPM | 71.10% | 75.05% | 91.29% | 0.8237 |
| 1D-CNNM | 72.91% | 74.31% | 92.56% | 0.8243 |
| CN-MLPM | 76.17% | 77.84% | 96.30% | 0.8609 |

As can be seen from the results, the detection performance of CN-MLPM on D_S is very close to that of MLPM in all indexes, which indicates that the network traffic data do not lose a lot of information that is helpful to distinguish the nature of network behavior in the feature transformation. On D_T , the detection performance of CN-MLPM is superior to MLPM, which indicates that the network intrusion detection model based on domain confusion obtains robustness to network environmental noise and can effectively improve the detection performance in the migration application.

In order to measure the degradation degree of different models, the $D - ratio$ was taken as an important indicator. Its calculation formula is shown in Equation (8), in which the $D - ratio$ refers to the degradation degree of different models, acc is the evaluation accuracy of the model on D_S , and acc' is the evaluation accuracy of the model on D_T .

$$D - ratio = \frac{acc - acc'}{acc} \quad (8)$$

With the work of reference [20], the migration performance of various methods is shown in Table 7.

Table 7. The Migration Performance of Various Methods.

| Model | Accuracy_ D_S | Accuracy_ D_T | $D - Ratio$ |
|------------------|-----------------|-----------------|-------------|
| PCA | 90.98% | 67.48% | 25.83% |
| Isolation Forest | 91.11% | 44.79% | 50.84% |
| Auto-Encoder | 94.26% | 70.97% | 24.71% |
| MLPM | 96.13% | 71.10% | 26.04% |
| 1D-CNNM | 92.51% | 72.91% | 21.19% |
| CN-MLPM | 96.15% | 76.17% | 20.78% |

The model proposed in this paper has better robustness to the noise of the network environment and can cope with the migration application of network intrusion detection model.

5. Conclusions

Although the network intrusion detection model based on deep learning has achieved remarkable performance, there is relatively little research on its migration application, which greatly affects the promotion and productive application of deep learning in the area of network security. This paper proposes a network intrusion detection method based on domain confusion. Based on the idea of GANs and Auto-Encoder, a domain confusion network is designed to process the traffic data, so as to reduce the part about network environment in the sample as much as possible and, at the same time, to retain the part that is helpful to distinguish the nature of network behavior. With the domain confusion model, an intrusion detection model with high robustness to network environmental noise

is constructed. Taking the network intrusion detection model as an example, this paper provides a feasible idea for the deep model in the area to improve the generalization performance, contributing to the actual deployment of network security systems based on deep learning.

Network traffic data are only a perspective to observe network behavior. In addition to the feature transformation method proposed in this paper, a more comprehensive way to characterize network behavior, such as modeling the packets of a network flow with the host state, is bound to further improve the generalization ability of deep models. Meanwhile, some works based on unsupervised learning methods have been proved to have better robustness, which is the direction of further research [21].

Author Contributions: Conceptualization, Y.Q. and H.M.; Data curation, Y.Q.; Formal analysis, Y.Q. and H.M.; Funding acquisition, Y.B.; Investigation, Y.Q.; Project administration, Y.B.; Resources, Y.B.; Software, Y.Q. and Y.J.; Supervision, H.M.; Validation, Y.Q. and H.M.; Visualization, Y.Q.; Writing—original draft, Y.Q.; Writing—review and editing, Y.Q. and Y.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Fund of China, grant number 62176264. The APC was funded by the National Natural Science Fund of China, grant number 62176264.

Data Availability Statement: The data used to support the findings of this study are available from reference [15].

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

References

1. Yang, Z.; Liu, X.; Li, T.; Wu, D.; Wang, J.; Zhao, Y.; Han, H. A systematic literature review of methods and datasets for anomaly-based network intrusion detection. *Comput. Secur.* **2022**, *161*, 102675. [CrossRef]
2. Sommer, R.; Paxson, V. Outside the closed world: On using machine learning for network intrusion detection. In Proceedings of the 2010 IEEE Symposium on Security and Privacy, Berkeley/Oakland, CA, USA, 16–19 May 2010.
3. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Bengio, Y. Generative adversarial networks. *Commun. ACM* **2020**, *63*, 139–144. [CrossRef]
4. Bontemps, L.; Cao, V.L.; McDermott, J.; Le-Khac, N.A. Collective anomaly detection based on long short-term memory recurrent neural networks. In Proceedings of the International Conference on Future Data and Security Engineering, Can Tho City, Vietnam, 23–25 November 2016; Springer: Berlin/Heidelberg, Germany, 2016.
5. Kang, M.J.; Kang, J.W. Method of intrusion detection using deep neural network. In Proceedings of the 2017 IEEE International Conference on Big Data and Smart Computing (BigComp), Jeju Island, Republic of Korea, 13–16 February 2017; pp. 313–316.
6. Shone, N.; Ngoc, T.N.; Phai, V.D.; Shi, Q. A deep learning approach to network intrusion detection. *IEEE Trans. Emerg. Top. Comput. Intell.* **2018**, *2*, 41–50. [CrossRef]
7. Arp, D.; Quiring, E.; Pendlebury, F.; Warnecke, A.; Pierazzi, F.; Wressnegger, C.; Rieck, K. Dos and Don'ts of Machine Learning in Computer Security. In Proceedings of the USENIX Security Symposium, Boston, MA, USA, 10–12 August 2022.
8. Wang, J.; Lan, C.; Liu, C.; Ouyang, Y.; Qin, T.; Lu, W.; Yu, P. Generalizing to unseen domains: A survey on domain generalization. *IEEE Trans. Knowl. Data Eng.* **2022**, *1*. [CrossRef]
9. Saxena, D.; Cao, J. Generative adversarial networks (GANs) challenges, solutions, and future directions. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–42. [CrossRef]
10. Karras, T.; Aila, T.; Laine, S.; Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. *arXiv* **2017**, arXiv:1710.10196.
11. Karras, T.; Laine, S.; Aila, T. A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 4401–4410.
12. Spampinato, C.; Palazzo, S.; D'Oro, P.; Giordano, D.; Shah, M. Adversarial framework for unsupervised learning of motion dynamics in videos. *Int. J. Comput. Vis.* **2020**, *128*, 1378–1397. [CrossRef]
13. Ganin, Y.; Ustinova, E.; Ajakan, H.; Germain, P.; Larochelle, H.; Laviolette, F.; Lempitsky, V. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.* **2016**, *17*, 1–35.
14. Bank, D.; Koenigstein, N.; Giryas, R. Autoencoders. *arXiv* **2020**, arXiv:2003.05991.
15. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp* **2018**, *1*, 108–116.
16. Gharib, A.; Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. An evaluation framework for intrusion detection dataset. In Proceedings of the 2016 International Conference on Information Science and Security (ICISS), Pattaya, Thailand, 19–22 December 2016; pp. 1–6.
17. RFC3917[EB/OL]. 2014. Available online: <http://www.ietf.org/rfc/rfc3917.txt> (accessed on 10 December 2022).

18. Lashkari, A.H.; Draper-Gil, G.; Mamun, M.S.I.; Ghorbani, A.A. Characterization of tor traffic using time based features. In Proceedings of the ICISSP, Porto, Portugal, 19–21 February 2017; pp. 253–262.
19. Wang, W.; Zhu, M.; Wang, J.; Zeng, X.; Yang, Z. End-to-end encrypted traffic classification with one-dimensional convolution neural network. In Proceedings of the 2017 IEEE International Conference on Intelligence and Security Informatics (ISI), Beijing, China, 22–24 July 2017; pp. 43–48.
20. Verkerken, M.; D’hooge, L.; Wauters, T.; Volckaert, B.; De Turck, F. Towards Model Generalization for Intrusion Detection: Unsupervised Machine Learning Techniques. *J. Netw. Syst. Manag.* **2021**, *30*, 1–25. [[CrossRef](#)]
21. Qu, Y.; Ma, H.; Jiang, Y. CRND: An Unsupervised Learning Method to Detect Network Anomaly. *Secur. Commun. Netw.* **2022**, *2022*, 9509417. [[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.