Investigating Domain Adaptation for Network Intrusion Detection

1st Hamza Alami
College of computing
Mohammed VI Polytechnic University
Ben Guerir, Morocco

2nd Meryem Janati Idrissi
College of computing
Mohammed VI Polytechnic University
Ben Guerir, Morocco

3rd Abdelkader El Mahdaouy
College of computing
Mohammed VI Polytechnic University
Ben Guerir, Morocco

4th Abdelhak Bouayad
College of computing
Mohammed VI Polytechnic University
Ben Guerir, Morocco

5th Zakaria Yartaoui
College of computing
Mohammed VI Polytechnic University
Ben Guerir, Morocco

6th Ismail Berrada
College of computing
Mohammed VI Polytechnic University
Ben Guerir, Morocco

***Abstract*—With the ever-increasing network intrusion techniques, the effectiveness of conventional Network Intrusion Detection Systems (NIDS) solutions has become limited. In response, machine learning-based NIDS are proposed as an alternative to handle new designed intrusions. However, building NIDS based on machine learning techniques requires high-quality labeled datasets which can cost large amount of time and resources. In this paper, we propose and evaluate a domain adaptation method designed for NIDS. First, we use raw packets and NFStream to represent flows as images. Then, we rely on convolutional neural networks to extract features, and gradient reversal to perform domain adaptation. Hence, leveraging a labeled dataset (source domain) to build models that perform well on unlabeled dataset (target domain). Finally, we leverage suitable intrusion detection metrics and three publicly available datasets, including USTC-TC2016, CIC-IDS2017, and CUPID to perform performance evaluation. The obtained results show that we need to further investigate domain adaptation techniques for NIDS.**

***Index Terms*—network intrusion, domain adaptation, CIC-IDS2017, gradient reversal, convolutional neural network**

I. Introduction

In recent years, the continuously growing number of cyberattacks has introduced new challenges to the security of computer networks [1]. Network Intrusion Detection Systems (NIDS) have arisen as an important component in safeguarding sensitive information, critical infrastructures, and the integrity of data transmission in both private and public sectors [2], [3]. These systems provide the capability to detect and respond to malicious activities in real time. The development of successful NIDS based on machine learning depends heavily on high-quality labeled datasets. These datasets play a crucial role in training machine learning models and assessing their performance under various conditions. However, collecting and annotating such labeled datasets presents a great challenge due to several factors [1]. First, the dynamic nature of cyberattacks demands datasets that encompass a wide array of attack types and normal network behaviors, making it tough to curate comprehensive and representative NIDS datasets. Next, the manual annotation of network traffic data by domain experts is a time-consuming and resource-intensive process. Additionally, the sensitivity and privacy concerns surrounding real-world network traffic limit the sharing and distribution of large-scale labeled NIDS datasets. Consequently, the scarcity and costliness of high-quality labeled datasets are significant bottlenecks in the advancement of NIDS research. Thus, researchers have explored alternative solutions such as weak-supervised and domain adaptation methods to alleviate these challenges.

Domain adaptation techniques [4] are a promising solution to address the scarcity of NIDS labeled datasets. These methods aim to bridge the gap between labeled source domains (e.g., publicly available datasets) and target domains (e.g., private or real-world networks) by leveraging knowledge transfer and adaptation mechanisms. This approach effectively circumvents the need for extensive labeled datasets in the target domain while still improving the performance of NIDS in cross-domain scenarios. In addition, a machine learning model optimized using a NIDS dataset performs poorly when tested with another NIDS dataset [5]. Hence, domain adaptation may help models catch invariant features and improve network intrusion detection performances.

In this paper, we propose and evaluate a domain adaptation method for NIDS. Our technique consists of presenting flows as images based on raw network traffic and NFStream [6]. Then, we use a 1D convolutional neural network to extract features from raw flows. We rely on the gradient reversal layer [7] to perform domain adaptation and focal loss [8] to optimize our network attack classifier. We performed various experiments to evaluate the performance of domain adaptation, we also evaluated the performance of optimized models with unseen datasets. We use three publicly available datasets, including USTC-TFC2016, CIC-IDS2017 [9], and CUPID [10]. The obtained results show that we need further investigations to build effective NIDS models based on domain adaptation. The rest of the paper is organized as follows: Section 2 presents some works that applied domain adaptation in the field NIDS; Section 3 describes our domain adaptation method for network intrusion detection; Section 4 gives all the

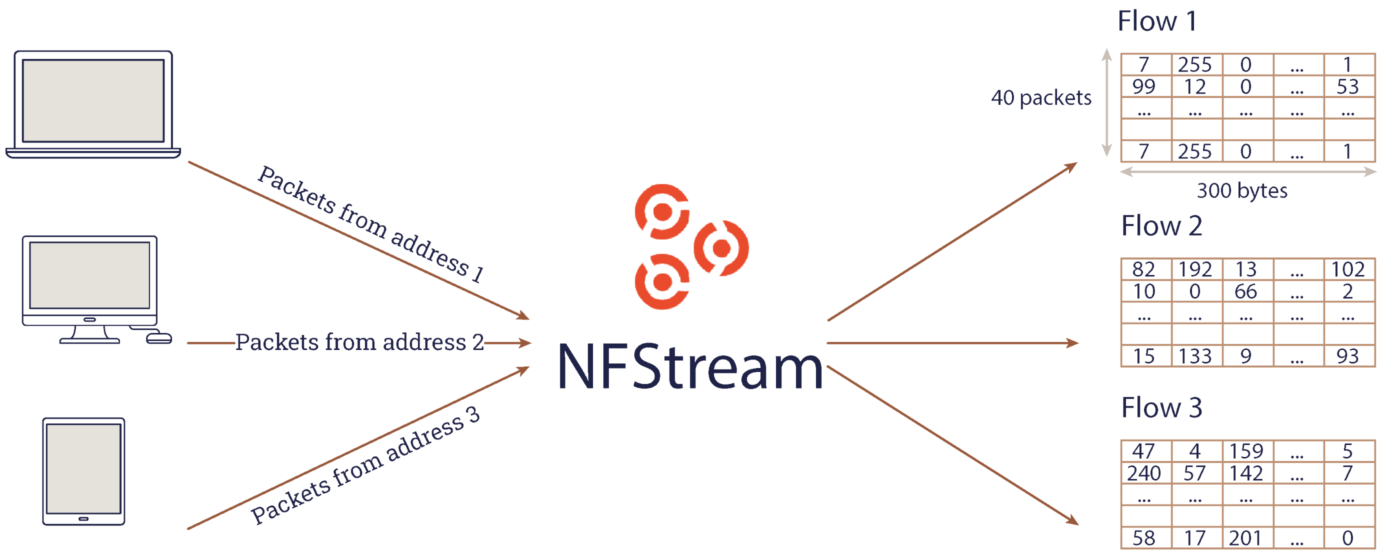

Fig. 1. Extracting raw flow features based on packets bytes and NFStream

necessary information about the experimental evaluation; and Section 5 presents the conclusion and perspectives.

II. Related Work

Few works addressed the challenge of transfer learning between different NIDS domains. In this section, we describe various works that applied unsupervised domain adaptation methods for NIDS. In the paper [5], the authors built various network flow classifiers based on machine learning methods, namely decision tree (DT), random forest (RF), logit, k nearest neighbor (k-NN), artificial neural network (ANN), long short term memory (LSTM), and gated recurrent unit (GRU). They found that when the training and testing are sampled from the same dataset the models reach high performances. However, when the training and testing datasets are different (sampled from two different datasets) the models score very bad performances. They used the Kyoto+2006, NSL-KDD, and gureKDD datasets with a 80% training and 20% testing split to evaluate their methodology. The authors used accuracy and F1-score as the main performance measures. In [11], the authors aim at detecting attacks within smart grid networks. They leverage adversarial training to adapt optimized models on new data collected at different times. The paper shows that domain adaptation techniques improve the model performances when compared to conventional ML-based NIDSs. The main metric for evaluation was the F1-score. Another work [12] proposed a NID framework based on a weighted adversarial nets-based partial domain adaptation method. The idea consists of mapping two domains (source and target) to a domain-invariant feature space. They use a publicly labeled dataset (CIC-IDS2017) to build a model able to detect attacks within an IoT unlabeled dataset. Similarly, the authors [13] proposed and evaluated adversarial domain adaptation methods to handle the lack of labeled data. They evaluated their proposal according to two scenarios: 1) Same feature space where the source and target datasets have the same feature space; 2) Different feature space where the target dataset has a different feature space. Various experiments were performed using the NSL-KDD [14] and the UNSW-NB15 [15] datasets. The metrics used to measure the model performances are the accuracy and F1-score. The last paper [3] introduced a method which consists of using an unsupervised technique for intrusion detection based on domain invariant features extracted by an adversarial domain adaptation method from different network domains. The main idea is to extract invariant features by training an adversarial neural network on source domains then train a one-class SVM model using these invariant features. They performed extensive experiments with both NFv2-CIC-2018 and NFv2-UNSW-NB15 datasets [16]. Their method achieved good cross-domain performance. The main metric used in this work is the F1-score.

All the discussed works have contributed to the NIDS domain adaptation challenge, all these methods measured the performance of models with the target domain, however they never measure its performance on unseen dataset. In addition, the presented works used the accuracy and F1-score as the main metric for performance evaluation. In our work, we fill these gaps by performing evaluations with unseen dataset and we use suited metrics for intrusion detection, including detection rate and false detection rate.

III. Method

In this section, we present the two main steps we performed to build a domain adaptation method for NIDS. First, we extract raw features from packets using the concept of the flow and NFStream [6], we named this step *Data preprocessing*. Then, we design the neural network architecture and the objective function to perform domain adaptation, this step is called *Domain Adaptation for NIDS*.

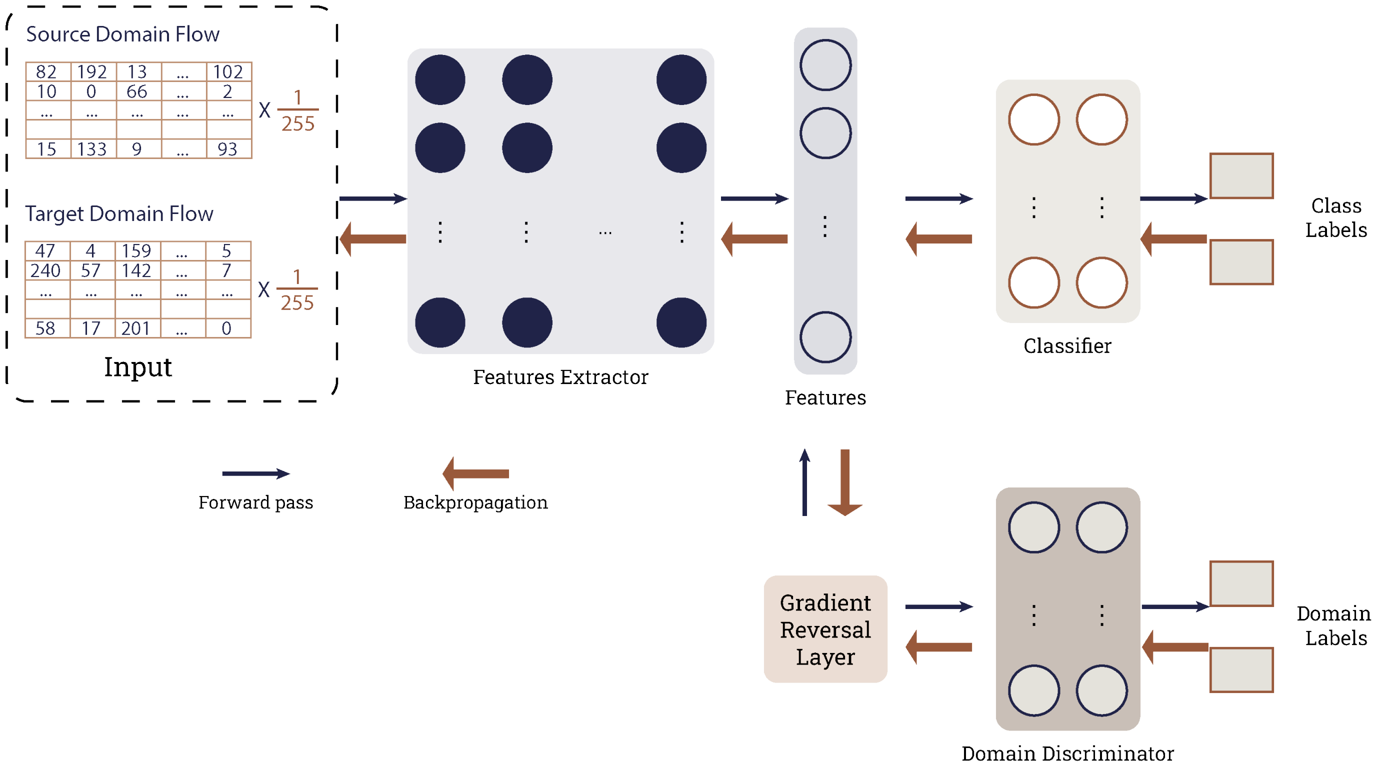

Fig. 2. Domain adaptation technique for NIDS

A. *Data preprocessing*

We compute raw flow features based on network packet bytes and NFStream [6]. The process involves analyzing the packets' information to identify and group them into coherent flows. These flows represent the communication between different hosts or endpoints on a network. Figure 1 illustrates the process of extracting raw flow features from network packets.

First, network packets are captured from the network using NFStream [6]. This tool intercepts the packets passing through a specific network interface or captures them from network traffic logs. Next, we build network flows by grouping packets according to the 5 tuples fields (source and destination addresses, source and destination ports, and the protocol attribute). Finally, we represent each flow by concatenating its first 40 packets bytes. If the flow contains less than 40 packets we perform 0 padding. In addition, we consider only the first 300 bytes from each packet, if the packet size is less than 300 we perform also 0 padding. Thus, each flow is represented as a matrix of bytes with the shape (40, 300). Figure 1 presents samples of raw flow features extracted from packets bytes. We divide raw flow features by 255 to normalize features since the values of bytes are between 0 and 255.

B. *Domain Adaptation for NIDS*

We employ the DANN [7] to transfer knowledge from a source domain (source NIDS domain) to a target domain (target NIDS domain), even when the two domains have different characteristics. Figure 2 depicts the overall architecture of the proposed domain adaptation method for NIDS. It consist of three main module, including *features extractor*, *gradient reversal layer*, and *class and domain classifiers*.

Features Extractor: Our features extractor leverages convolutional neural networks to extract valuable features from the raw flow representation. It consist of 3 convolutional layers. Each layer applies the following pipeline: 1D convolutional operation, Rectified Linear Unit (ReLU), and Max Pooling 1D. In the first place, we apply a 1D convolution operation to the raw flow representation. Let $p_i \in \mathbb{R}^{300}$ the normalized packet bytes representation of the i-th packet within a flow. A flow is then represented by the following equation:

$$F = p_1 \oplus p_2 \oplus \cdots \oplus p_{300} \tag{1}$$

where F and $\oplus$ are the raw flow representation and the concatenation operator. Given a raw flow representation F of size (40,300) and a kernel w of size (300, K), the first 1D convolution operation can be defined as follows:

$$y[n] = (F * w)[n] = \sum_{k=0}^{K-1} F[n+k] \cdot w[k] \tag{2}$$

where $y[n]$ is the output at position n, $x[n]$ represents the flow's packet at position n, and $w[k]$ denotes the filter coefficient vector at position k. The other convolution operations consider only a 1D signal as input, consequently, the filter size is K. This process is repeated for each position n in the raw flow representation, resulting in an output sequence of length $40 - K + 1$. Next, we apply the ReLU activation function to introduce non-linearity and capture complex patterns in the data. ReLU is defined by the next equation:

$$f(x) = \max(0, x) \tag{3}$$

Finally, we perform a max pooling operation to reduce the spatial dimensionality of the feature maps while retaining the most important features. It divides the feature maps into non-overlapping regions and takes the maximum value within each region. The max pooling operation is defined as follows:

Max pooling is a downsampling operation commonly used in convolutional neural networks (CNNs) to reduce the spatial dimensionality of feature maps while retaining the most salient features. Let's describe the MaxPooling1D operation mathematically:

Given an input sequence or feature map of length N, the Max Pooling 1D operation divides the input into non-overlapping regions or windows of a specified size and takes the maximum value within each window. The operation can be defined as follows:

$$y[i] = \max(x[i \cdot s : i \cdot s + k]) \tag{4}$$

where $y[i]$ represents the output value at position i in the downsampled feature map, s is the stride, which represents the step size for moving the window, and k is the size of the window.

Gradient Reversal Layer: Gradient Reversal Layer (GRL) is proposed in [7], it does not modify the input data in the forward propagation. However, in the backpropagation GRL takes the gradient from the subsequent layer and flips its sign (by multiplying the gradient with -1) before passing it to the preceding layer. Our objective is to extract features from source and target instances such that these features appear to come from the same distribution. In other words, we want to extract features that are not affected by the difference in domains between the source and target instances. This means that features should have the same distribution in both domains. We achieve this by maximizing the domain classification loss for the features extractor while minimizing the domain classification loss for the domain classifier.

Class and Domain Classifiers: Our objective function minimizes both the label classifier loss and the domain classifier loss. In order to compute the label classifier loss, we use the focal loss proposed in [17]. We define the label classifier loss by the following equation:

$$\begin{aligned} &focal\_loss(x, y) = \\ &\frac{1}{N}\sum_{n=1}^{N} \alpha \times (1 - exp(-CE_n))^{\gamma} \times CE_n \end{aligned} \tag{5}$$

where x is the predicted probabilities, y is the target, α and γ are tunable parameters, N is the batch size, and CE is the unreduced cross entropy loss defined as follows:

$$\begin{aligned} &CE(x, y) = \{CE_1, CE_2, ..., CE_N\}^T, \\ &CE_n = -\sum_{c=1}^{C} log\left(\frac{exp(x_{n,c})}{\sum_{i=1}^{C} exp(x_{n,i})}\right) y_{n,c} \end{aligned} \tag{6}$$

TABLE I
CONVOLUTIONAL LAYERS HYPER-PARAMETERS

	Layer 1	Layer 2	Layer 3
#neurons	64	128	256
kernel size	3	3	3
stride	1	1	1
padding	1	1	1
pool kernel size	2	2	2
pool stride	2	2	2

where C is the number of class labels.

Similarly, we define the domain classifier loss as the binary cross-entropy loss since we consider only two domains (source and target). It is defined as the following:

$$\begin{aligned} &domain\_loss = \\ &-\frac{1}{N}\sum_{n=1}^{N} y_n \times log(x_n) + (1 - y_n) \times log(1 - x_n) \end{aligned} \tag{7}$$

Consequently, we define the global loss function as the sum of the labels classification loss and the domain classification loss. The next equation provides the global loss:

$$global\_loss = focal\_loss + domain\_loss \tag{8}$$

IV. Experimental Results

In this section, we evaluate the performance of our domain adaptation method for NIDS. We present all the necessary information, including datasets, experimental settings, and performance evaluation.

A. Datasets

We used three datasets to evaluate domain adaptation for NIDS, namely the two well-known datasets USTC-TFC2016[1] and CIC-IDS2017 [9], and CUPID [10] a new dataset that contains both scripted and human-generated attack traffic. We split the datasets into three sets training (66,67%), validation (16,67%), and testing (16,67%). We divide all datasets into train validation and test set. Since the datasets are designed for multiclass classification, we map the problem into a binary classification task by considering all attacks as one-class attack. We should notice that the authors of these datasets provide pcap files, thus we perform flow features extraction leveraging NFStream.

B. Experimental Settings

In this work, we use Python and its machine learning libraries including PyTorch, PyTorch Lightning, Numpy, Pandas, and Transfer Learning Library [18], [19] to perform all experiments. All these experiments were run by the African SuperComputing Center HPC service, supported by Mohammed VI Polytechnic University[2].

Our features extractor is composed of three convolution layers that contain 64, 128, and 256 neurons, respectively. Table I presents all the hyper-parameters concerning the convolutional

[1]https://github.com/yungshenglu/USTC-TFC2016
[2]https://ondemand.hpc.um6p.ma/

TABLE II
PERFORMANCE EVALUATION OF THE SAME MODEL TRAINED ON THE SOURCE DATASET AND TESTED WITH TWO DIFFERENT DATASETS

	Test Dataset											
	USTC-TFC2016				CIC-IDS2017				CUPID			
Source Dataset	Accuracy	F1-score	DR	FDR	Accuracy	F1-score	DR	FDR	Accuracy	F1-score	DR	FDR
USTC-TFC2016	99.99%	99.99%	99.99%	0.00%	20.21%	**29.97%**	98.12%	**82.31%**	23.70%	**34.38%**	94.70%	**79.00%**
CIC-IDS2017	55.96%	**3.13%**	1.59%	**6.86%**	99.57%	98.76%	97.64%	0.09%	81.90%	**43.70%**	33.28%	**36.38%**
CUPID	54.11%	**0.14%**	0.07%	**97.53%**	86.14%	**56.29%**	51.26%	**37.60%**	94.34%	84.50%	73.18%	0.02%

TABLE III
PERFORMANCE EVALUATION OF DOMAIN ADAPTATION METHOD WHERE THE SOURCE DATASET IS USTC-TFC2016 AND THE TARGET DATASETS ARE CIC-IDS2017 AND CUPID. THE UNSEEN DATASET EVALUATION PRESENTS THE MEASURES OBTAINED WHEN TESTED ON A DIFFERENT DATASET WHICH IS NOT SOURCE OR TARGET.

	Source dataset evaluation				Target dataset evaluation				Unseen dataset evaluation			
Target Dataset	Accuracy	F1-score	DR	FDR	Accuracy	F1-score	DR	FDR	Accuracy	F1-score	DR	FDR
CIC-IDS2017	99.88%	99.86%	99.84%	0.12%	63.29%	**46.70%**	92.42%	**68.75%**	25.34%	32.29%	84.36%	80.03%
CUPID	99.84%	99.82%	99.82%	0.17%	21.38%	**29.89%**	96.28%	**82.31%**	47.05%	29.91%	53.53%	79.25%

layers. We represent features in a 1024-dimension space and we use the Adam optimizer [20] with a 0.001 learning rate to optimize the model parameters. We also set the focal loss parameters γ and α to 2 and 0.25 according to the original paper results [8]. It is important to mention that we keep the default parameters of NFStream except for the idle timeout which we have changed to 180 seconds.

We evaluated the domain adaptation method for NIDS according to measures that are well-suited for cybersecurity, including Accuracy, F1-score, Detection Rate (DR), and False Detection Rate (FDR). We describe these measures as the following:

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

where TP are true positives, TN are true negatives, FP are false positives, and FN are false negatives.

$$Precision = \frac{TP}{TP + FP}$$
$$DR = \frac{TP}{TP + FN}$$
$$F1 - score = \frac{2 \times Precision \times DR}{Precision + DR}$$
$$FDR = \frac{FP}{TP + FP}$$

the FDR measure presents the ratio of security operators' time misspent in tracking false alerts which can consume a large amount of time and resources.

C. Performance Evaluation

Prior to presenting the results of our domain adaptation method for NIDS, we evaluate the performance of our label classifier on unseen datasets. Table II presents the performance of the same model trained without adaptation and evaluated with data sampled from the same training distribution and data sampled from different distributions. We highlighted the $F1 - score$ and FDR on unseen datasets, it is clear that the model performs poorly on unseen datasets.

Next, we evaluate the performance of the domain adaptation method. In order to measure the performance of our method according to various scenarios, we designed a specific evaluation template. This template consists of considering one dataset as the source domain, one dataset as the target domain, and one dataset as the unseen domain. For instance, the USTC-TFC2016 dataset is the source domain, the CIC-IDS2017 dataset is the target domain, and the CUPID dataset is the unseen domain.

Table III presents the obtained results when considering USTC-TFC2016 as the source domain. First, we notice that the performances improve slightly for both target domains CIC-IDS2017 (from 29.97% to 46,70% F1-score and from 82.31% to 68.75% FDR) and CUPID (from 34.38% to 56.29% f1-score and from 79.00% to 37.60% FDR). However, the performances related to the unseen datasets didn't change.

In the next evaluation, we train the model with the CIC-IDS2017 source domain. Table IV gives the scored measures in both cases where we use the other two datasets as target domains, independently. The performances did not improve for both target domains USTC-TFC2016 (from 3.13% to 2.56% F1-score and from 6.86% to 71.02% FDR) and CUPID (from 43.70% to 56.29% f1-score and from 36.38% to 37.60% FDR). However, the performances related to the unseen datasets didn't change. Thus, the domain adaptation in this scenario (CIC-IDS2017 as source domain) didn't work.

In the last evaluation, we set the CUPID dataset as the source domain. Table V depicts the obtained performance of the domain adaptation method. The performances improved for the USTC-TFC2016 target domain (from 0.14% to 36.74% F1-score and from 97.53% to 12.47% FDR). However, the performances slightly dropped for the CUPID target domain (from 56.29% to 53.87% f1-score and from 37.60% to 46.82% FDR). In addition, the performances related to the unseen datasets didn't increase.

Given the results presented in the Tables III, IV, and V, we show that domain adaptation based on gradient reversal layer can improve in some cases the performance. However, this improvement is still poor and we need to study further

TABLE IV

PERFORMANCE EVALUATION OF DOMAIN ADAPTATION METHOD WHERE THE SOURCE DATASET IS CIC-IDS2017 AND THE TARGET DATASETS ARE USTC-TFC2016 AND CUPID. THE UNSEEN DATASET EVALUATION PRESENTS THE MEASURES OBTAINED WHEN TESTED ON A DIFFERENT DATASET WHICH IS NOT SOURCE OR TARGET.

	Source dataset evaluation				Target dataset evaluation				Unseen dataset evaluation			
Target Dataset	Accuracy	F1-score	DR	FDR	Accuracy	F1-score	DR	FDR	Accuracy	F1-score	DR	FDR
USTC-TFC2016	99.48%	98.48%	97.68%	0.70%	54.44%	**2.56%**	1.34%	**71.02%**	79.81%	10.56%	5,65%	18.91%
CUPID	99.34%	98.11%	98.89%	2.66%	81.85%	**39.56%**	28.15%	**33.49%**	56.44%	5.22%	2.68%	4.68%

TABLE V

PERFORMANCE EVALUATION OF DOMAIN ADAPTATION METHOD WHERE THE SOURCE DATASET IS CUPID AND THE TARGET DATASETS ARE USTC-TFC2016 AND CIC-IDS2017. THE UNSEEN DATASET EVALUATION PRESENTS THE MEASURES OBTAINED WHEN TESTED ON A DIFFERENT DATASET WHICH IS NOT SOURCE OR TARGET.

	Source dataset evaluation				Target dataset evaluation				Unseen dataset evaluation			
Target Dataset	Accuracy	F1-score	DR	FDR	Accuracy	F1-score	DR	FDR	Accuracy	F1-score	DR	FDR
USTC-TFC2016	93.85%	83.03%	71.24%	0.50%	64.22%	**36.74%**	23.25%	**12.47%**	82.78%	59.58%	72.91%	49.63%
CIC-IDS2017	94.12%	83.81%	72.17%	0.07%	83.73%	**53.87%**	54.58%	**46.82%**	52.25%	1.80%	0.98%	88.89%

methods for knowledge transfer in the context of NIDS. The results also reveal that our domain adaptation method for NIDS didn't catch invariant features since it failed to generalize to unseen domains. These findings approve that building a model for various domains related to NIDS is still a challenging task.

V. Conclusion and future work

In conclusion, this paper contributes valuable insights into the potential of domain adaptation in the field of NIDS. First, we represented flows as images based on raw packets and NFStream. Next, we extracted features using 1D convolutional neural networks. Then, we performed domain adaptation and label classification using gradient reversal and focal loss. Finally, we performed various evaluations to measure the performance of the proposed method. We leverage three publicly available datasets USTC-TFC2016, CIC-IDS2017, and CUPID, and well-suited metrics (DR and FDR) for network intrusion detection. The results show the need for new studies and works that investigate domain adaptation for building NIDS. By addressing the limitations of traditional supervised learning paradigms and embracing domain adaptation techniques, we believe that NIDS can forge ahead as a robust and proactive line of defense against the ever-evolving landscape of cyber-attacks, safeguarding critical networks and ensuring the integrity of sensitive information.

In future work, we plan to continue our journey and evaluate new domain adaptation methods for NIDS. Moreover, we project to use different features and study their impact on domain adaptation-based NIDS.

References

[1] B. Seraphim, S. Palit, K. Srivastava, and E. Poovammal, "A survey on machine learning techniques in network intrusion detection system," in *2018 4th International Conference on Computing Communication and Automation (ICCCA)*, 2018, pp. 1–5.

[2] T. T. Huong, T. P. Bac, D. M. Long, T. D. Luong, N. M. Dan, L. A. Quang, L. T. Cong, B. D. Thang, and K. P. Tran, "Detecting cyberattacks using anomaly detection in industrial control systems: A federated learning approach," *Computers in Industry*, vol. 132, p. 103509, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0166361521001160

[3] S. Layeghy, M. Baktashmotlagh, and M. Portmann, "Di-nids: Domain invariant network intrusion detection system," *Knowledge-Based Systems*, vol. 273, p. 110626, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705123003763

[4] G. Wilson and D. J. Cook, "A survey of unsupervised deep domain adaptation," *ACM Trans. Intell. Syst. Technol.*, vol. 11, no. 5, jul 2020. [Online]. Available: https://doi.org/10.1145/3400066

[5] S. Al-Riyami, F. Coenen, and A. Lisitsa, "A re-evaluation of intrusion detection accuracy: Alternative evaluation strategy," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 2195–2197. [Online]. Available: https://doi.org/10.1145/3243734.3278490

[6] Z. Aouini and A. Pekar, "Nfstream: A flexible network data analysis framework," *Computer Networks*, vol. 204, p. 108719, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128621005739

[7] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. March, and V. Lempitsky, "Domain-adversarial training of neural networks," *Journal of Machine Learning Research*, vol. 17, no. 59, pp. 1–35, 2016. [Online]. Available: http://jmlr.org/papers/v17/15-239.html

[8] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017.* IEEE Computer Society, 2017, pp. 2999–3007. [Online]. Available: https://doi.org/10.1109/ICCV.2017.324

[9] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy, ICISSP 2018, Funchal, Madeira - Portugal, January 22-24, 2018*, P. Mori, S. Furnell, and O. Camp, Eds. SciTePress, 2018, pp. 108–116. [Online]. Available: https://doi.org/10.5220/0006639801080116

[10] H. Lawrence, U. Ezeobi, O. Tauil, J. Nosal, O. Redwood, Y. Zhuang, and G. Bloom, "Cupid: A labeled dataset with pentesting for evaluation of network intrusion detection," *Journal of Systems Architecture*, vol. 129, p. 102621, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762122001515

[11] Y. Zhang and J. Yan, "Semi-supervised domain-adversarial training for intrusion detection against false data injection in the smart grid," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–7.

[12] Y. Fan, Y. Li, H. Cui, H. Yang, Y. Zhang, and W. Wang, "An intrusion detection framework for iot using partial domain adaptation," in *Science of Cyber Security*, W. Lu, K. Sun, M. Yung, and F. Liu, Eds. Cham: Springer International Publishing, 2021, pp. 36–50.

[13] A. Singla, E. Bertino, and D. Verma, "Preparing network intrusion detection deep learning models with minimal data using adversarial domain adaptation," in *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, ser. ASIA CCS '20.

New York, NY, USA: Association for Computing Machinery, 2020, p. 127–140. [Online]. Available: https://doi.org/10.1145/3320269.3384718

[14] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6.

[15] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6.

[16] M. Sarhan, S. Layeghy, and M. Portmann, "Towards a standard feature set for network intrusion detection system datasets," *Mobile Networks and Applications*, vol. 27, no. 1, pp. 357–370, nov 2021. [Online]. Available: https://doi.org/10.1007%2Fs11036-021-01843-0

[17] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[18] J. Jiang, Y. Shu, J. Wang, and M. Long, "Transferability in deep learning: A survey," 2022.

[19] J. Jiang, B. Chen, B. Fu, and M. Long, "Transfer-learning-library," https://github.com/thuml/Transfer-Learning-Library, 2020.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980