Computers and Electrical Engineering 108 (2023) 108692

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Explainable Cross-domain Evaluation of ML-based Network Intrusion Detection Systems☆

Siamak Layeghy *, Marius Portmann

School of ITEE, University of Queensland, Brisbane 4072, QLD, Australia

ARTICLE INFO

Keywords:
Network intrusion detection
Supervised learning
Unsupervised learning
Cross-domain evaluation
Domain specific evaluation
Shapely analysis

ABSTRACT

Many of the proposed machine learning (ML) based network intrusion detection systems (NIDSs) achieve near perfect detection performance when evaluated on synthetic benchmark datasets. However, there is no record of if and how these results generalise to other network environments. In this paper, we investigate the cross-domain performance of ML-based NIDSs by extensively evaluating eight supervised and unsupervised learning models on four recently published benchmark NIDS datasets. Our investigation indicates that none of the considered models is able to generalise over all studied datasets. Interestingly, our results also indicate that the cross-domain performance has a high degree of asymmetry, i.e., swapping the source and target domains can significantly change the classification performance. Our investigation also indicates that overall, unsupervised learning methods perform better than supervised learning models in our considered scenarios. We further used SHAP values to explain the observed cross-domain performance results. They show a high correlation between a good model performance and a correspondence between feature distributions/values and Attack/Benign classes.

1. Introduction

A quick search of the academic literature in network intrusion detection systems (NIDSs) reveals that there are hundreds of proposals based on machine learning (ML) algorithms. In most of these proposals, the performance is evaluated using a publicly available benchmark NIDS dataset such as UNSW [1], CIC-IDS [2], and KDD99 [3]. For this purpose, like in other fields that use machine learning, the benchmark dataset is divided into the training and test subsets, the classification algorithms (ML models) are trained on the training subset and evaluated on the test subset. In some of these studies, such as [4,5], multiple benchmark dataset are used in this way. That is, the classification algorithm is trained and tested on each benchmark dataset separately.

While the accuracy, detection rate and other performance metrics reported in many of these studies are near perfect, and the studies are designed and performed rigorously, these excellent results have unfortunately not translated into ML-based NIDSs in practical network deployments with close to 100% detection performance.

The *cross-domain* performance of a machine learning model is defined as the model performance when there is a distribution shift between the training and test/evaluation datasets [6]. We believe that an evaluation of the cross-domain performance of ML-based NIDSs is an important, but so far under-investigated step towards bridging the gap between the excellent results achieved by the academic research community, and the practical impact of the research.

☆ This paper is for regular issues of CAEE. Reviews were processed by Associate Editor Dr. Yang Li and recommended for publication.

* Corresponding author.
E-mail address: siamak.layeghy@uq.net.au (S. Layeghy).

https://doi.org/10.1016/j.compeleceng.2023.108692
Received 14 October 2022; Received in revised form 8 March 2023; Accepted 16 March 2023
Available online 28 March 2023

In particular, we assume that the setup/environment in which benchmark NIDS datasets are created/collected varies among the different available benchmark datasets, and most likely also varies significantly from the characteristics of potential real-world deployment networks.

Applying machine learning for cross-domain network intrusion detection, i.e., using separate datasets for the training and evaluation, has been explored in a few previous works such as [7,8] that examine domain adaptation and transfer learning in the field of ML-based NIDSs. The classification of attack classes of the same datasets not present during training has also been explored in a few previous works such as [9–11] that investigate zero-day and unseen attack detection. To the best of our knowledge, the only previous works that partially or indirectly investigate the cross-domain performance of ML-based NIDSs are [12] and [13], in which the cross-domain evaluation of a single classification algorithm has been explored for eight NIDS datasets. Apart from these studies, which are discussed in more detail in the following section, we have not been able to find other works considering this subject.

Our work aims to address this gap by taking advantage of four recently published NIDS benchmark datasets [14], which have been converted from their original format into a common NetFlow-based format, with an identical feature set. The availability of these NIDS benchmark datasets with a common feature set and common data representation is critical for enabling our investigation. We evaluate the cross-domain performance of ML-based NIDSs and compare their *domain specific* and cross-domain performance across these four datasets. Our evaluation includes both the supervised and unsupervised learning models.

We first calculate the performance of NIDSs for *domain-specific* evaluation, in which each benchmark dataset is used for both training and evaluation. Then, we calculate their *cross-domain* performance in which each NIDS is trained on one dataset and evaluated against the other three benchmark datasets. Then we compare their performance in domain specific evaluation with their cross-domain performance. We further apply Shapley values [15] on the data and models in different scenarios, and provide an explanation of the achieved results. In summary, the main findings of this paper are:

- None of the considered models in this study is able to generalise over all studied datasets.
- The cross-domain performance among the considered datasets and models has a high degree of asymmetry, i.e., swapping the source and target domains can significantly change the classification performance.
- Overall, unsupervised learning methods perform better than supervised learning models in our considered scenarios.
- The Shapley analysis indicates that the cross-domain performance in the studied scenarios is correlated with the absence/presence of correspondence between the distribution/values of one or more features and Attack/Benign classes.

2. Related works

While there are no previous works that systematically evaluate the cross-domain performance of ML-based NIDSs, the two previous studies [12,13] investigate the cross-domain performance of a single ML-based NIDS model.

In [12] it is shown that the near perfect domain-specific performance of a simple supervised ML-based NIDS considerably drops when assessed in cross-domain evaluation. This study uses the Kyoto+ [16], gureKDD [17] and NSL-KDD [18] NIDS datasets which are published in 2006, 2008 and 2009 respectively. However, the Kyoto+ dataset is not included in the cross-domain evaluation, due to the difference of the feature set with the other datasets. While this study discusses cross evaluation of ML-based NIDSs, it does not investigate the factors for the poor performance of the ML-models in cross-domain evaluation. In addition, the results presented in [12] are based on two old benchmark datasets, which makes it hard to draw conclusions about the generalisability of ML-based NIDSs in current network environments. Furthermore, the study only includes a single supervised learning model and does not include an evaluation of unsupervised learning models, as provided in this paper.

The main focus of [13], the other work, is proposing XeNIDS, a framework for cross evaluation of ML-based NIDSs. In this framework 10 series of combinations/scenarios of the Benign and Attack classes of two different datasets are defined for using in the training and evaluation of the NIDSs. They consider their combination number 4 as a scenario for testing the generalisation capability. In this scenario, the Benign and Attack classes of the same dataset are used for training, but the Benign and different classes of Attacks from different datasets can be used for the evaluation. Though, the Benign samples for the training and evaluation should be selected from the same dataset. Then paper proposes an ensemble ML-based NIDS, and shows its detection performance for the different types of attacks extracted from eight different NIDS datasets. This shows a significant drop for those attacks that have not been included in the training dataset.

While the two above works partially consider the cross-domain evaluation of ML-based NIDSs, it is not their main focus, and hence there are limitations in a number of aspects. In particular, these works do not consider the asymmetry of cross-domain performance, as we do in this paper, by considering both directions of cross-domain evaluation by swapping the source and target datasets. As our results show, this is critical. More importantly, the generalisation experiments discussed in [13] cannot be considered as an evaluation on a fully new domain, since the Benign traffic is selected from the same domain in both the training and evaluation phases. Finally, neither of these works consider cross-domain performance in the context of unsupervised learning, as we do in this paper.

Table 1
Summary information of NetFlow datasets used in this paper along with their class distributions.

Dataset	No. of records	Classes	No. of records	Class (%)
NFv2-BoT-IoT	37,763,497	Benign	135,037	0.36
		DDoS	18,331,847	48.54
		DoS	16,673,183	44.15
		Reconnaissance	2,620,999	6.94
		Theft	2,431	0.01
NFv2-CIC-2018	18,893,708	Benign	16,635,567	88.05
		DDOS-HOIC	1,080,858	5.72
		DoS-Hulk	432,648	2.29
		DDoS-LOIC-HTTP	307,300	1.63
		Bot	143,097	0.76
		Infilteration	116,361	0.62
		SSH-Bruteforce	94,979	0.50
		DoS-GoldenEye	27,723	0.15
		FTP-BruteForce	25,933	0.14
		DoS-SlowHTTPTest	14,116	0.08
		DoS-Slowloris	9,512	0.05
		Brute Force-Web	2,143	0.01
		DDOS-LOIC-UDP	2,112	0.01
		Brute Force-XSS	927	0.01
		SQL Injection	432	0.01
NFv2-ToN-IoT	16,940,496	Benign	6,099,469	36.01
		Scanning	3,781,419	22.32
		XSS	2,455,020	14.49
		DDoS	2,026,234	11.96
		Password	1,153,323	6.81
		DoS	712,609	4.21
		Injection	684,465	4.04
		Backdoor	16,809	0.1
		MITM	7,723	0.05
		Ransomware	3,425	0.02
NFv2-UNSW-NB15	2,390,275	Benign	2,295,222	96.02
		Exploits	31,551	1.32
		Fuzzers	22,310	0.93
		Generic	16,560	0.69
		Reconnaissance	12,779	0.53
		DoS	5,794	0.24
		Analysis	2,299	0.10
		Backdoor	2,169	0.09
		Shellcode	1,427	0.06
		Worms	164	0.01

3. Datasets

In order to compare the performance of an ML-based NIDS on different datasets, they all need to have the same feature set. This is specially the case for the evaluation of cross-domain performance of NIDSs as defined in this paper, the difference between domain-specific and cross-domain performance. In the case of the cross-domain evaluation, a model is initially trained on one dataset, then the same model, is applied on other datasets which requires the exact set of feature to be available on the training and evaluation datasets.

However, the existing datasets prior to [14] are created using different tools resulting in datasets with various number of different features, with only 4 common features among the four benchmark NIDS datasets as shown in [4]. As such, availability of NIDS Benchmark datasets with a common feature set was a prerequisite for the cross-domain evaluation performed in this paper. This was realised through conversion of four publicly available NIDS benchmark datasets from their original formats. Table 10 shows the list of all NetFlow features available in the converted datasets along with a brief explanation of each feature and the indication if the feature is used in this study.

Table 1 shows the summary information of the four datasets used in this study, all in NetFlow (NF) format. These datasets, which include NFv2-UNSW-NB15, NFv2-CIC-2018, NFv2-ToN-IoT and NFv2-BoT-IoT [14], are converted from the pcap files of their original formats published as UNSW-NB15 [1], CIC-2018 [2], ToN-IoT [19] and BoT-IoT [20] respectively. The first version of the NetFlow format datasets (NFv1) with 20 features has been published in [4], and the second version, that is used in this work and includes 43 features, is discussed in [14]. The procedure for converting the original format into NetFlow and the labelling of the converted flows are also explained in [14].

The three original datasets UNSW-NB15, ToN-IoT and BoT-IoT are published by the same research group. Since the network setup used for the traffic generation for each dataset is very different, they represent different network environments, i.e., different

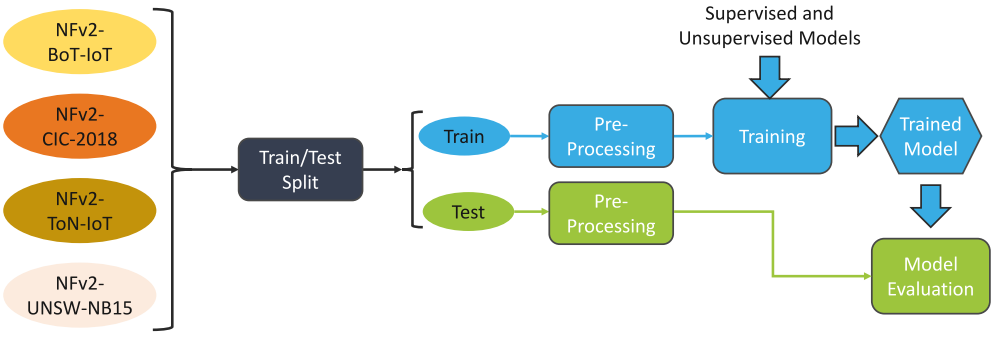


Fig. 1. Domain-specific model evaluation procedure.

domains, and represent a valid basis for the evaluation of the cross-domain performance of NIDSs. This is also the case for CIC-2018, the fourth considered dataset, which has been generated by a different research group in a completely different network setup.

Since the sizes of these datasets are different, e.g. NFv2-UNSW-NB15 has 2,390,275 flows and the BoT-IoT dataset has 37,763,497 flows, we used a stratified (with regards to classes) sampling strategy with a size of 1,000,000 flows and we ran all the experiments on the sampled datasets.

As can be seen, all these datasets come with a Benign class and various numbers of different attack classes. Since the set of attack classes is different for each dataset, we therefore focus on binary classification only, i.e. Benign vs Attack traffic in our analysis. Consequently, all attack classes of each dataset are aggregated under a single class called Attack. Hence, the datasets used in this study have two classes, i.e. Benign and Attack.

Initially, when we ran the experiments related to unsupervised learning algorithms, we noticed that the performance (F1-Score) of the models was very poor when trained and evaluated on the following two datasets, NFv2-BoT-IoT and NFv2-ToN-IoT, in both domain-specific or cross-domain evaluations. This was mainly due to the unrealistically high imbalance of the datasets, i.e. the ratio of the Attack to Benign flows, as can be seen in Table 1. Accordingly, we created and used a balanced version of all four datasets (via down sampling) in terms of Attack-Benign labels, i.e. equal number of samples from Attack and Benign classes by maintaining the ratio of individual attack classes to all attacks, and used for the unsupervised learning models experiments. These datasets, extended with a "-b" suffix such as NFv2-BoT-IoT-b, indicate the balanced version of the original, imbalanced (e.g. NFv2-BoT-IoT) dataset.

4. Domain-specific evaluation

In *domain-specific* evaluation, which is the common method for the evaluation of NIDSs, the same dataset is used for both the training and evaluation. Fig. 1 shows the procedure for the domain-specific evaluation of the models. As can be seen, in this method, a publicly available benchmark NIDS dataset is divided into the training and test/evaluation subsets, and each subset is separately pre-processed. The classification algorithms is trained on the training subset and the trained model is tested/evaluated against the test subset of the same dataset. In some cases, such as [4,5], more than one dataset is used for the domain-specific evaluation. However, the classification algorithm is trained and tested/evaluated against each dataset separately. Since the training and test/evaluation data are both collected from the same environment, this approach still is referred to as domain-specific evaluation.

4.1. Supervised learning

Initially, we evaluate supervised learning methods via the domain-specific evaluation approach. There are many previous studies which achieve a very high detection performance using this approach, based on the same benchmark NIDS datasets that we consider in this paper, but in their original (non NetFlow) format, in particular UNSW-NB15 [1] and CIC-2018 [2]. Since the NetFlow version of these datasets have been published relatively recently, there are only a few studies such as [4,5] and [21], that have used them. However, we cannot use their results in our comparisons because we need to evaluate the same model later in the cross-domain setup, and these studies do not provide such evaluation. As such, we use the result of our experiment, even for the domain-specific evaluation, which might be found in the literature.

The experimental setup used to implement the models, train and evaluate them includes a virtual machine running on VMware ESXi v6.7 platform using 2 virtual CPUs of 4 GHz, 32 Gb of RAM and Mint Linux distribution with a kernel version of 5.4. All the models are implemented, trained and evaluated in Python. Since the main focus of this study is the evaluation of ML-based NIDSs in a cross-domain setup, and the models utilised in this research are simple and small scale, the computational complexity is not considered in detail in this paper. A thorough study of computational cost of similar models used for the implementation of NIDSs is available in [22].

We chose four supervised learning models including two deep and two shallow learning methods. For the deep learning we chose the same number of neural network layers and nodes from two different architectures. The first model is a simple Feed Forward neural network and the second model is a Long Short-Term Memory (LSTM) network with the same number of layers and nodes

Table 2
The model parameters for two supervised and one unsupervised deep learning-based NIDSs along with the other parameters for training and evaluation of the models.

	Feed Forward	LSTM	AutoEncoder
No. Hidden Layers	4	4	3-1-3
No. Nodes (each layer)	10	10	32-16-8-4-8-16-32
Learning Ratio	0.0001	0.0001	0.0001
Dropout Ratio	0.2	0.2	–
Batch Size	512	512	512
Validation Split	0.3	0.3	0.3
No. of Folds	5	5	5

Table 3
The model parameters for two shallow learning-based NIDSs along with the other parameters for training and evaluation of the models.

	Random Forest	Extra Tree
ccp_alpha	0.001	0.001
Batch Size	512	512
Validation Split	0.3	0.3
No. of Folds	5	5

Table 4
Domain-specific performance (F1-Score (%)) of 4 supervised (2 shallow and 2 deep) learning methods.

Source/Target	Extra Tree	Random Forest	Feed Forward NN	LSTM NN
NFv2-BoT-IoT	99.82%	99.82%	99.76%	99.92%
NFv2-CIC-2018	84.62%	95.44%	46.27%	90.17%
NFv2-ToN-IoT	77.63%	77.33%	93.98%	76.38%
NFv2-UNSW-NB15	91.73%	92.17%	90.63%	92.82%

on each layer. The model parameters are shown in Table 2. For the shallow learning methods, we used a Random Forest and an Extra-Tree classifier, which allow us to easily implement our NIDSs without much effort to tune hyperparameters. The parameters used in these models are mostly the default parameters of the Python library except *ccp_alpha* which defines the number of nodes pruned and its value is shown along with the rest of parameters in Table 3.

It is possible to search and fine tune the hyperparameters of the classification algorithms to achieve the best possible performance on a training dataset. However, this may increase the chance of over-fitting to the training dataset and reducing the performance on other datasets, not seen during the training [23]. Since in the next steps of this study we are going to evaluate the performance of these models via the cross-domain approach, hyperparameter fine tuning might bias the performance results towards the training dataset, i.e., the domain-specific evaluation. Accordingly, in order to avoid over-fitting to the training datasets, we mostly used the default hyperparameters for the different models, as provided by the corresponding software libraries, i.e. scikit-learn (for the shallow learning methods) [24] and TensorFlow (for the deep learning methods) [25].

The main performance metric used in this evaluation is *F1-Score*, which is the harmonic mean of the two main performance metrics, *precision* and *recall*

$$F1 = 2\frac{precision \times recall}{precision + recall} \tag{1}$$

where precision is $\frac{TP}{PP}$ and recall is $\frac{TP}{P}$ in which PP is the total number of samples predicted positive and P stands for the number of positive samples. F1-Score is a more common metric for the cases where datasets are imbalanced. Table 4 shows the classification performance (F1-Score) for the four supervised learning models in domain-specific evaluation. As can be seen, the performance on each dataset are mostly consistent across all four ML-based NIDSs (1% to 5% variations), except the Feed Forward NN model, which its performance is up to 50% different, in one case, from the rest of models. It is noticeable that even these simple models, without fine tuning of their hyperparameters, are able to achieve a very high classification performance in most cases. For instance, all these simple models have been able to achieve a F1-Score above 99% on the NFv2-BoT-IoT dataset, and for the rest of the datasets there is at least one NIDS model which achieves a F1-Score greater than 92%.

4.2. Unsupervised learning

While the cross-domain performance of a single supervised learning model has partly been investigated for the NIDS application in a previous study [12], there are no such results for unsupervised learning methods in the context of NIDS, to the best of our knowledge. As such, we have included four unsupervised learning models in our investigation. These models, which include the *Isolation Forest (IsolationForest)* [26], *One-Class Support Vector Machines(oSVM)* [27], *Stochastic Gradient Descent one-Class Support Vector Machines (SGD-oSVM)* [28], and AutoEncoder are used for the purpose of anomaly/attack detection in our context.

Table 5
Domain-specific performance (F1-Score (%)) of 4 unsupervised (semi-supervised) learning methods.

Source/Target	IsolationForest	oSVM	SGD-oSVM	AutoEncoder
NFv2-BoT-IoT-b	87.58%	72.62%	72.70%	28.17%
NFv2-CIC-2018-b	85.05%	65.62%	65.60%	39.29%
NFv2-ToN-IoT-b	57.15%	54.65%	54.64%	30.04%
NFv2-UNSW-NB15-b	73.67%	76.26%	76.25%	40.61%

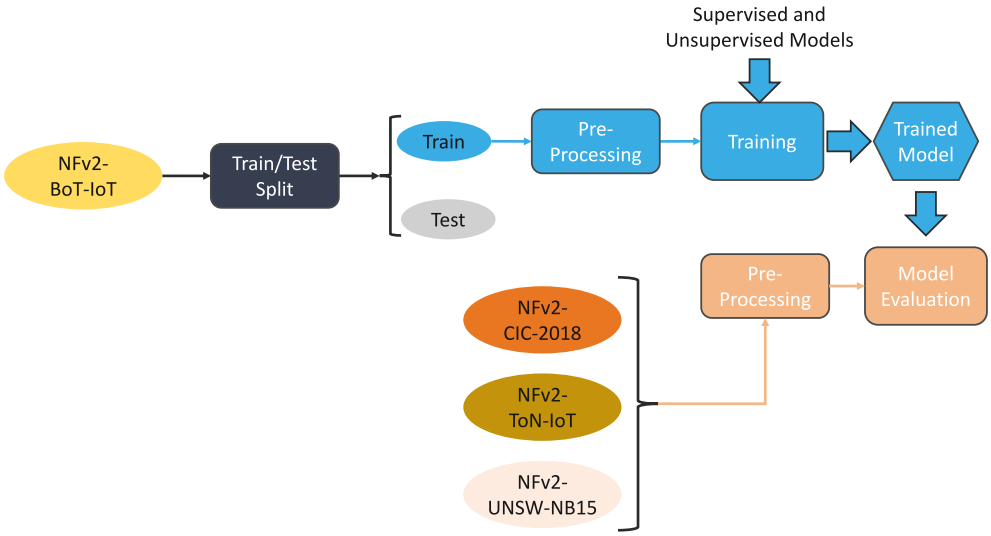


Fig. 2. Cross-domain model evaluation procedure.

Although these models are typically considered unsupervised learning algorithms, since they are very sensitive to anomalies/outliers in the training set [27], we have used them in a semi-supervised manner. This means the models are exposed to one class of data during the training phase, but are tested against both classes. To achieve this, each benchmark dataset is divided into the training and test subsets. Then, the Attack samples are removed from the training subset. Hence, the models are trained only on normal/Benign data samples and evaluated against the test subset, which includes samples of both the Benign and Attack classes.

Similar to the case of the supervised learning models, we did not fine tune the hyperparameters of the models, for achieving the best possible performances, and mostly used the default values provided in the scikit-Learn package. The parameters for AutoEncoder are shown in Table 2. Table 5 shows the results of applying these four NIDS models on the four considered benchmark datasets. The main conclusion from this table can be summarised in two points. First, all models have lower performance on NFv2-ToN-IoT, which was also the case for the supervised learning models (except the Feed Forward model). Secondly, the Isolation Forest model has higher performances across all the four datasets.

5. Cross-domain evaluation

In the previous section we evaluated the performance of both supervised and unsupervised learning NIDS algorithms via a domain-specific approach, which is the current de-facto standard in the NIDS literature. As it was shown, we were able to get close to the high performance values reported in the literature, using our simple non-fine-tuned models, for most of the cases in both the supervised and unsupervised learning methods.

While having a high performance in a domain-specific evaluation is a necessary condition for any NIDS targeting the real-world application, there are other conditions to be met as well. One such condition is the cross-domain performance, i.e., the ability to generalise and translate the high performance on a domain-specific evaluation to cross-domain evaluations.

Fig. 2 shows the cross-domain model evaluation procedure. As can be seen, in order to evaluate the cross-domain performance of an ML-based NIDS model, we need to evaluate the performance of the model in different network environments, referred to as *domains* in this context and properly represented by the selected NIDS benchmark datasets. The first domain/NIDS dataset, which is called the *source domain*, is used for the training, and the second domain/NIDS dataset, which is referred to as the *target domain*, is used for the evaluation. In this way, the NIDS model is only trained on the source domain and it does not see the target domain during training.

Table 6
Cross-domain performance (F1-Score) of supervised learning-based NIDSs.

Target	Source	Classifier			
		Extra Tree	Random Forest	Feed Forward	LSTM
NFv2-BoT-IoT	NFv2-CIC-2018	0.14%	12.91%	91.83%	54.74%
	NFv2-ToN-IoT	24.42%	46.75%	0.56%	0.04%
	NFv2-UNSW-NB15	94.83%	7.61%	71.38%	81.78%
NFv2-CIC-2018	NFv2-BoT-IoT	22.32%	22.32%	22.32%	28.15%
	NFv2-ToN-IoT	44.83%	4.50%	57.55%	21.82%
	NFv2-UNSW-NB15	17.47%	7.70%	34.89%	14.20%
NFv2-ToN-IoT	NFv2-BoT-IoT	85.50%	85.50%	85.50%	83.65%
	NFv2-CIC-2018	62.25%	30.28%	69.01%	48.03%
	NFv2-UNSW-NB15	73.39%	0.00%	3.82%	81.40%
NFv2-UNSW-NB15	NFv2-BoT-IoT	4.90%	4.90%	4.90%	4.41%
	NFv2-CIC-2018	0.57%	0.84%	0.05%	9.63%
	NFv2-ToN-IoT	0.00%	0.25%	0.00%	0.40%

5.1. Supervised learning

Here we evaluate the cross-domain performance of the same four supervised learning models considered in the domain-specific evaluation. We ran a set of three experiments for each NIDS model in which the model trained in each domain-specific experiment was evaluated against the other three benchmark datasets without any further training. Hence, for each target domain/dataset we have three different results for the same NIDS model, each indicating the performance of the model when trained on a different source domain/dataset. Table 6 shows the results of these experiments, ordered by the target domain and source domain, to allow easy comparison with the domain-specific results shown in Table 4. As can be seen, each supervised learning-based NIDS model is evaluated for the 12 different source/target domain combinations.

Fig. 3 provides a more visual representation of both the domain-specific and the cross-domain evaluation results, with each sub-figure showing the F1-Score results for a different supervised model. The used colour map indicates a very high F1-Score (100%) in dark blue, and a very low value (0%) in light yellow, with the in-between values as indicated. The source datasets are indicated on the vertical axis, and the corresponding target datasets are shown on the horizontal axis.[1] On the diagonal, we observe the domain-specific results, where a model is trained and evaluated on the same dataset. The off-diagonal results show the cross-domain results.

The mostly dark colouring on the diagonal indicates a generally high performance in the domain-specific evaluation of all the four supervised models. However, the mostly lighter colours, and hence lower F1-Scores, indicate a generally poor ability of the models to generalise from a source dataset to a different target dataset. There are some exceptions though. For example, for the Extra-Tree model (Fig. 3-a), UNSW-NB15 as the source domain generalises well to the BoT-IoT dataset as the target domain, with an F1-Score of 94.83%.

Interestingly, this result is highly asymmetrical. If we swap the source and target domain, and use BoT-IoT as the source and UNSW-NB15 as the target, the Extra-Tree model only achieves 4.90%. While this is the most prominent example, we observe a generally high degree of asymmetry of the cross-domain performance across different source/target domain pairs and supervised learning models.

If we compare the cross-domain performance results across the four different classification algorithms, we observe some consistent patterns, but we also notice some significant differences. The Random-Forest model seems to perform quite differently from the other three models, in particular for the UNSW-NB15 dataset as the source domain (bottom row).

Finally, we also observe significant differences among the datasets. Most strikingly, we see that if the UNSW-NB15 dataset is chosen as the target domain, the results are very poor for any of the other datasets chosen as the source domain (rightmost column). This is consistent across all four classification algorithms.

Table 7 shows the average performance decay per model for the cross-domain evaluation compared to the corresponding domain-specific evaluation. Each column indicates a supervised learning model and each row indicates a source/training dataset. For instance, the cell in the first row and first column shows the average decay of the performance of the Extra-Tree model trained on NFv2-BoT-IoT when tested on NFv2-CIC-2018, NFv2-ToN-IoT and NFv2-UNSW-NB15 datasets, compared to its performance when tested on NFv2-BoT-IoT dataset. Investigating the performance decays presented in Table 7 indicates that:

(a) There is no supervised learning-based NIDS generalising over all combination of source-target domains.
(b) Deep learning-based NIDSs generalise better than shallow learning-based NIDSs in average.
(c) There is an average performance decay of 56.28% for the cross-domain cases compared to their corresponding domain-specific cases.

[1] In order to make the names of datasets readable in this figure, we used a larger font size, that necessitated to remove the "NFv2-" prefix from the name of all datasets to make them fit to the spaces.

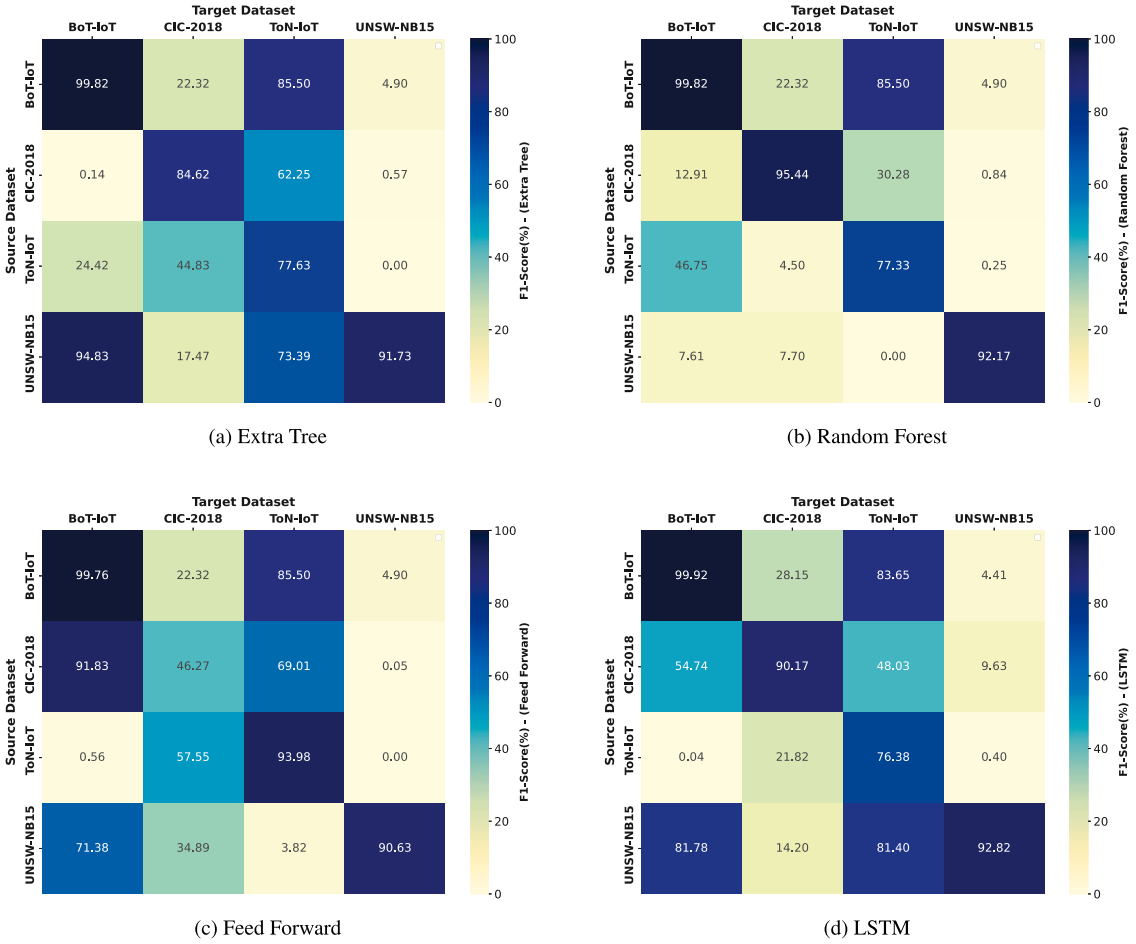


Fig. 3. Comparing the domain-specific and cross-domain performance (F1-Score (%)) of supervised learning-based NIDSs for (a) Extra Tree, (b) Random Forest, (c) Feed Forward, and (d) LSTM models. The diagonal entries show the domain-specific and off-diagonal values indicate the cross-domain performance.

Table 7
Cross-domain performance decay compared to domain-specific performance (average performance (F1-Score) decay) per model and source domain for the supervised learning models.

Source/Training dataset	Extra Tree Decay (avg.)	Random Forest Decay (avg.)	Feed Forward Decay (avg.)	LSTM Decay (avg.)	Average Decay per Source
NFv2-BoT-IoT	62.24%	62.25%	62.19%	61.18%	**61.96%**
NFv2-CIC-2018	63.63%	80.77%	−7.36%	52.70%	**47.43%**
NFv2-ToN-IoT	54.55%	60.16%	74.60%	68.96%	**64.57%**
NFv2-UNSW-NB15	29.84%	87.07%	53.93%	33.70%	**51.13%**
Average Decay per Model	**52.57%**	**72.56%**	**45.84%**	**54.13%**	**56.28%**

These conclusions are based on the average performance of models rather than individual model performance, and a higher number of evaluation datasets would further increase the validity of these conclusions.

5.2. Unsupervised learning

Similar to the supervised learning algorithms, we evaluated the cross-domain performance of unsupervised learning-based NIDSs using the same four benchmark datasets as we used for the domain-specific evaluation. As per the domain-specific evaluation, training is performed using only the normal/Benign class of the source domain, while the trained model is exposed to both the Benign and Attack classes of the target domain for the evaluation of its cross-domain performance.

Table 8 shows the results of the cross-domain evaluation of the unsupervised learning algorithms. Similar to supervised learning algorithms, each model is trained on one dataset and evaluated on the other three datasets that were not used for training. For

Table 8
F1-Score (%) of unsupervised learning-based NIDSs for the cross-domain evaluation.

Target	Source	Unsupervised learning algorithm			
		IsolationForest	oSVM	SGD-oSVM	AutoEncoder
NFv2-BoT-IoT-b	NFv2-CIC-2018-b	49.35%	77.87%	77.87%	55.80%
	NFv2-ToN-IoT-b	60.52%	80.38%	80.86%	71.19%
	NFv2-UNSW-NB15-b	34.33%	0.23%	0.23%	57.06%
NFv2-CIC-2018-b	NFv2-BoT-IoT-b	0.18%	56.86%	58.71%	65.58%
	NFv2-ToN-IoT-b	0.25%	56.24%	58.12%	65.62%
	NFv2-UNSW-NB15-b	13.84%	56.66%	57.74%	62.50%
NFv2-ToN-IoT-b	NFv2-BoT-IoT-b	66.75%	66.75%	29.20%	30.45%
	NFv2-CIC-2018-b	62.12%	22.23%	22.17%	28.25%
	NFv2-UNSW-NB15-b	0.00%	7.10%	7.10%	28.37%
NFv2-UNSW-NB15-b	NFv2-BoT-IoT-b	0.00%	76.29%	76.28%	76.28%
	NFv2-CIC-2018-b	26.22%	64.80%	64.80%	65.94%
	NFv2-ToN-IoT-b	0.00%	64.85%	64.85%	76.28%

Table 9
Cross-domain performance decay compared to domain-specific performance (average performance (F1-Score) decay) per model and source for the unsupervised learning models.

Source/Training Dataset	IsolationForest	oSVM	SGD-oSVM	AutoEncoder	Avg. per source
NFv2-BoT_IoT	−65.27%	−18.51%	−17.55%	29.24%	**−18.02%**
NFv2-CIC_2018	−39.16%	−10.65%	−10.65%	10.71%	**−12.44%**
NFv2-ToN_IoT	−36.89%	12.51%	13.30%	40.99%	**7.48%**
NFv2-UNSW_NB15	−57.61%	−54.93%	−54.56%	8.71%	**−39.60%**
Avg. per Model	**−49.73%**	**−17.90%**	**−17.36%**	**22.41%**	**−15.65%**

each of the four datasets used as the source domain, we consider the other three as the target domain, resulting in 12 target/source domain combinations that are the basis for the cross-domain performance evaluation.

Fig. 4 visualises the results for the four considered unsupervised algorithms, and also includes the domain-specific results on the diagonal, corresponding to Fig. 3. As can be seen, while the performance of the models is not as high as the supervised learning algorithms, in the domain-specific experiments, there are many cases in which the cross-domain performance is equal or higher than for domain-specific performance.

Table 9 quantifies these outcomes in terms of the performance (F1-Score) decays. It shows the average performance decay for each unsupervised learning model in cross-domain evaluation compared to the corresponding domain-specific evaluation. Each column indicates an unsupervised learning model, each row indicates a source domain, and each value indicates the average of three F1-Score values. Excluding AutoEncoder for which the performance significantly changes with the anomaly threshold, investigating these performance decays indicates that

(a) There is no unsupervised learning model that generalises over all combination of source-target domains.
(b) The two unsupervised algorithms oSVM and SGD-oSVM are better generalised than Isolation Forest, even though their domain-specific performance is weaker.
(c) There is an average performance decay of 28.33% for the cross-domain cases compared to domain-specific cases.

5.3. Supervised vs unsupervised learning models

Comparing the results shown in Tables 7 and 9 indicates

(a) The unsupervised learning models generalise better with an overall average performance decay of 28.33%, compared to the supervised learning models with an overall average performance decay of 56.28%.
(b) The superiority of unsupervised learning models is consistent across all source domains, i.e., the average per source domain decay (last column of Tables 7 and 9) is significantly lower for unsupervised learning models for all the four datasets.

6. Explainable cross-domain performance

Explaining the behaviour of a machine learning model usually requires to investigate the impact of the features of input data on the output. There are a range of tools and techniques to study and estimate the feature importance and how much each feature impacts the model output. *SHapley Additive exPlanations (SHAP) values* [29] is one of the recent trends in explaining and interpreting the output of the AI/ML models in terms of the features of the datasets. It provides a value for each feature in the train/test datasets,

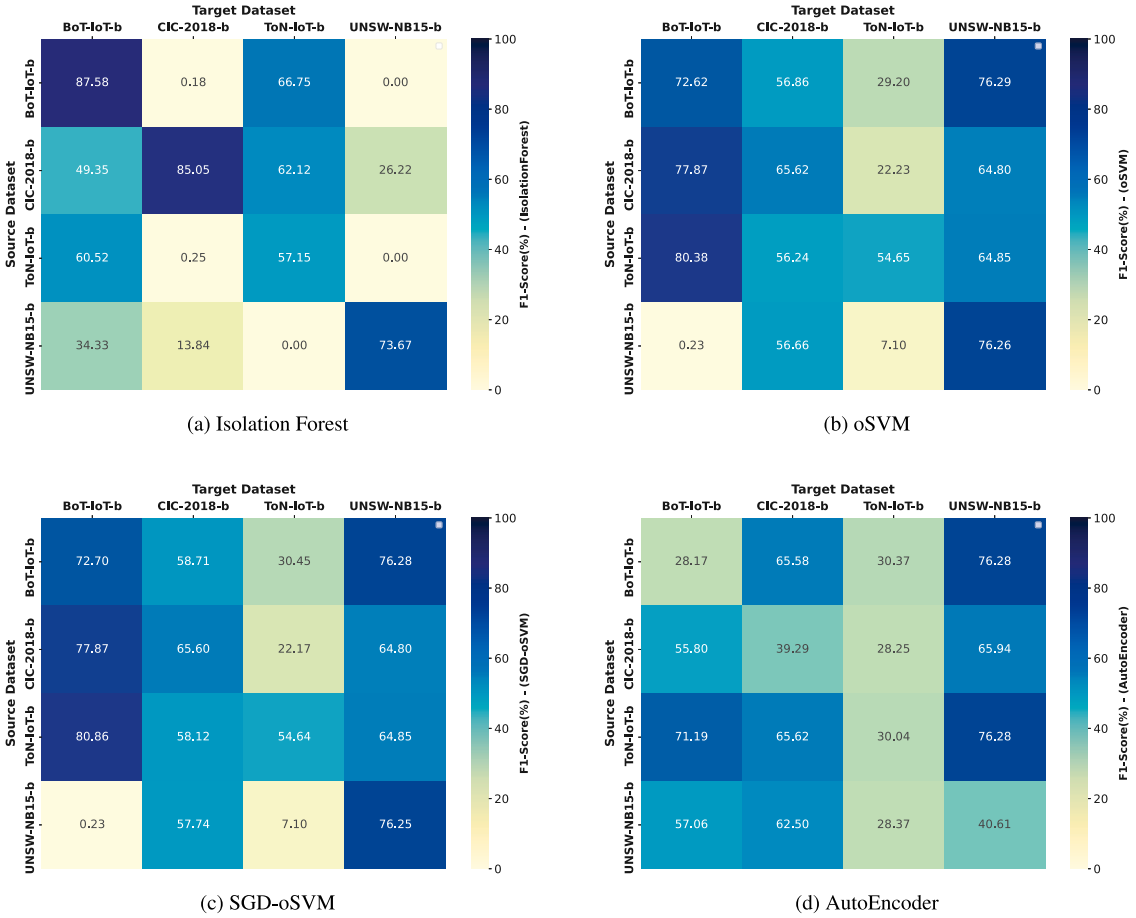


Fig. 4. Comparing the domain-specific and cross-domain performance (F1-Score (%)) of unsupervised learning-based NIDSs for (a) Isolation Forest, (b) oSVM, (c) SGD-oSVM models and (d) AutoEncoder. The diagonal entries show the domain-specific evaluation and off-diagonal values indicate the cross-domain evaluation.

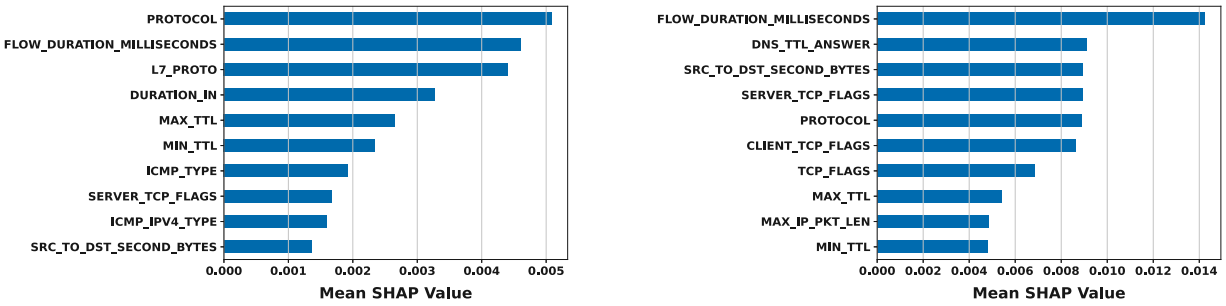
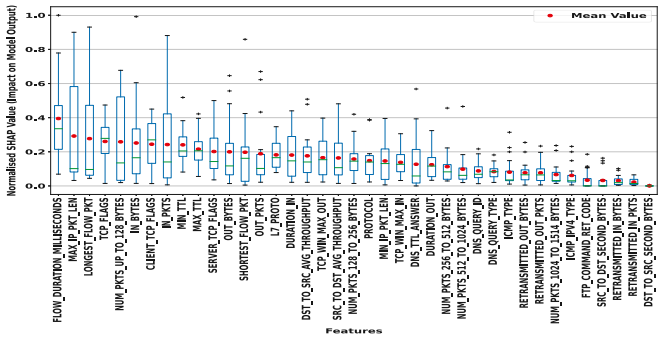


Fig. 5. Feature importance (Mean absolute SHAP value) of top 10 features for the Feed Forward model trained on NFv2-BoT-IoT dataset and evaluated on(a) NFv2-BoT-IoT and (b) NFv2-UNSW-NB15 datasets.

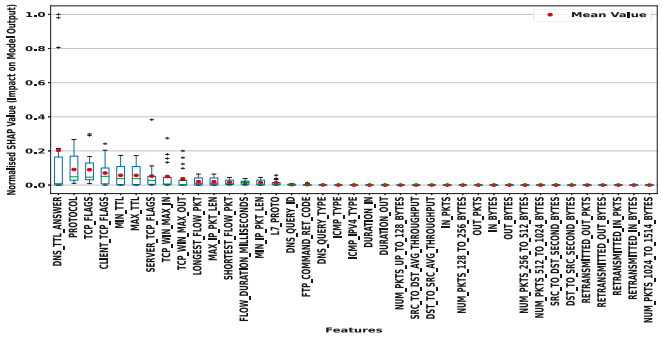
which indicates how much a feature has contributed to the generated output. Hence, these values depend on the training dataset, the classification algorithm (ML model), and the evaluation dataset.

Fig. 5 shows the mean absolute SHAP value for the top ten features where the Feed forward model is trained on NFv2-BoT-IoT and evaluated on (a) NFv2-BoT-IoT and (b) NFv2-UNSW-NB15. These figures, called the feature importance plot, show the features in descending order (of their mean of absolute SHAP values) on the vertical axis and the mean SHAP value on the horizontal axis.

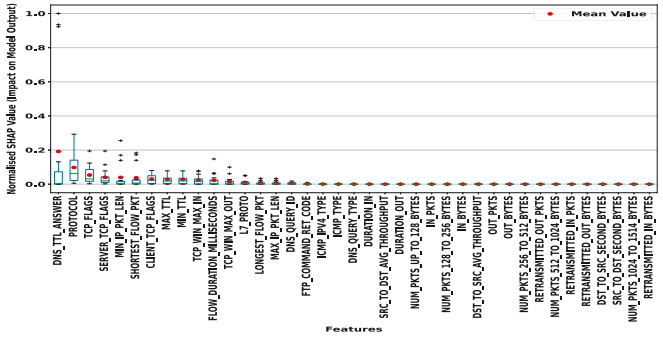
Comparing the two feature importance plots clearly shows that feature orders and the mean SHAP values of features significantly varies between the two. For instance, in Fig. 5-(a), PROTOCOL is the most important feature with a mean SHAP value of 0.005, while



(a) Isolation Forest



(b) oSVM



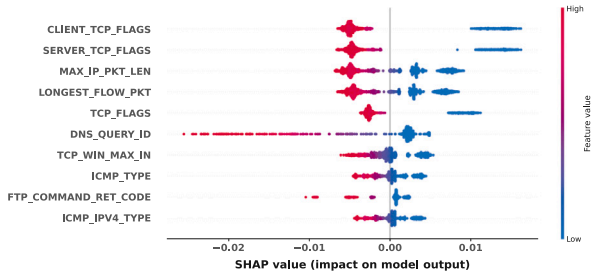
(c) SGD-oSVM

Fig. 6. Distribution of mean SHAP values of features for (a) Isolation Forest, (b) oSVM and (c) SGD-oSVM models in 16 different experiments. Each Boxplot shows the range of mean SHAP values of a feature for the same model in 16 different combinations of the source-target domains.

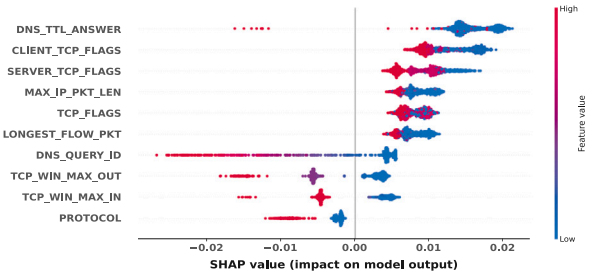
in Fig. 5-(b), the most important feature is FLOW_DURATION_IN_MILLISECONDS with a mean SHAP value of 0.014, and PROTOCOL is fifth important feature with a mean SHAP value of 0.008. This is a clear indication that the behaviour of the Feed-Forward model trained on NFv2-BoT-IoT is entirely different when tested on NFv2-BoT-IoT datasets and when tested on NFv2-UNSW-NB15. This conclusion is consistent with the results shown in Fig. 3-(C) for the Feed Forward model.

Accordingly, comparing the feature importance across all the 16 combinations of source-target domains will reveal the overall model behaviour. For this purpose, we computed the mean SHAP value of all features across all the experiments, and plotted their distributions side by side. Fig. 6 shows the distribution of these mean SHAP values for the three unsupervised learning models, the Isolation Forest, oSVM and SGD-oSVM respectively plotted in Fig. 6-(a), (b), and (c). The horizontal axis indicates the features and vertical axis indicates the normalised mean SHAP value. Since the range of mean SHAP values for different features were different, they have been normalised to make them comparable. The features are sorted in terms of their overall average in descending order, and the overall average of the mean SHAP value of each feature is also shown by a red circle.

As can be seen, the variance of the mean SHAP values of features in the case of Isolation Forest, Fig. 6-(a), is significantly larger than the other two models, oSVM and SGD-oSVM shown in Fig. 6-(b) and (c) respectively. As it was shown in Fig. 5, the variations of

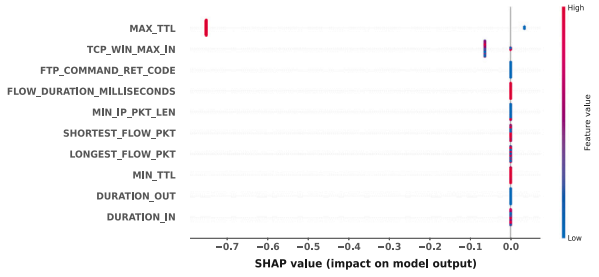


(a) Feed Forward - source:UNSW, target:UNSW

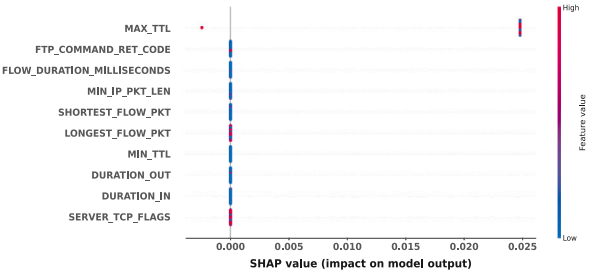


(b) Feed Forward - source:UNSW, target:ToN

Fig. 7. SHAP summary plots for the top 10 features in the experiments using the Feed Forward model where in (a) source: UNSW and target: UNSW and in (b) source: UNSW and target: ToN.



(a) Extra Tree - source:UNSW, target:BoT



(b) Extra Tree - source:BoT, target:UNSW

Fig. 8. SHAP summary plots for the top 10 features in the experiments using the Random Forest model where in (a) source: BoT and target: ToN, and in (b) source: ToN and target: BoT.

feature order and importance values are directly linked to the variations of model behaviour. Similarly, the considerable variations of the importance and order of the features across different source-target combinations for the Isolation Forest model indicates variations of its behaviour across these experiments.

The next two model, oSVM and SGD-oSVM, shown in Fig. 6-(b) and (c) respectively, have much lower variations compared to Isolation Forest. As such, it is expected that their behaviour is more similar across different combinations of source-targets, which means these two model are more generalisable compared to Isolation forest. This can be easily verified by comparing the average decay per model as shown in Table 9, which is 49.73%, 17.90% and 17.36% for the Isolation Forest, oSVM and SGD-oSVM, respectively. The oSVM and SGD-oSVM models, in addition to the lower variance of mean SHAP values, share the same order/rank for many of the features. This similarity in the distribution of mean SHAP value of features, which is an indicator of model behaviour similarity, explains their close average model decays (17.90% and 17.36%).

While the analysis of feature importance values explains the overall behaviour of the models in terms of their cross-domain performance, it cannot answer a question like why a model performs well on one dataset and not well on another dataset. To answer this kind of questions about the behaviour of the models, we need a more detailed analysis of the SHAP values.

The SHAP summary plot is an appropriate tool for this kind of analysis. A SHAP summary plot shows the SHAP value of features for individual data points. It shows how much impact each feature of a single data point has on generating the corresponding output. Fig. 7 shows two examples of SHAP summary plots. In each SHAP summary plot the horizontal axis indicates the SHAP value, the vertical axis indicates the features, and the feature values are shown using the colour range, from blue (low) to red (high). A positive SHAP value in our experiments indicates the impact of the feature towards the **Benign** class and a negative SHAP value indicates the impact towards the **Attack** class.

Each point in this plot is created by two values, the feature and the SHAP values. Accordingly, a single instance of a dataset sample corresponds with the number of points equal to the number of features. Since in these summary plots, only ten features are shown, a sample from a dataset corresponds with ten dots (data points) in a summary plot, one point per feature. Wherever multiple samples have the same SHAP value, their representative dot-points are piled up in a histogram manner. Hence, the larger height of the pile of dots indicates features of more sample points have the same SHAP value.

Since it is not possible to investigate the SHAP summary plots of all the experiment pairs separately, we chose three pairs of experiments with the maximum contrasting results that explain the main aspects of cross-domain performance observed in the results.

The first aspect is the high performance of a model in domain-specific evaluation compared to the low performance of the same model in cross-domain evaluation. Fig. 7-(a) and (b) are examples of SHAP summary plots for such a case in which the Feed-Forward model is trained on NFv2-UNSW-NB15 dataset and is evaluated against (a) the NFv2-UNSW-NB15 dataset and (b) the NFv2-ToN-IoT dataset.

(a) Extra Tree - source:BoT, target:ToN

(b) Extra Tree - source:ToN, target:BoT

Fig. 9. SHAP summary plots for the top 10 features in the experiments where in (a) source: BoT and target: ToN, and in (b) source: ToN and target: BoT, and in (c) source: UNSW and target: BoT, and in (d) source: BoT and target: UNSW using Extra Tree model.

As can be seen, in Fig. 7-(a) the feature values have a clear correspondence with the Benign and Attack classes. Most of the Blue dots (low feature values) have positive SHAP values (Benign) and most of the Red dots (high feature values) have negative SHAP values (Attack). This is a simple multi-rule classifier that can separate the Attack and Benign classes based on the simple rules in terms of the feature values.

In Fig. 7-(b), however, this correspondence of the feature values with the Attack and Benign classes hardly can be seen and most of the feature values either indicate the Benign class or mixes of two classes. This is obviously due to the difference of the feature distribution in the new target domain, NFv2-ToN-IoT with the source domain NFv2-UNSW-NB15 dataset. The result shown in Fig. 3-(c) confirms our conclusion from the SHAP summary plots and show a significant performance drop for the this case.

The other aspect of cross-domain performance observed in our results is its asymmetric behaviour, i.e., while a model has a high performance in a combination of the source-target domains, it has a low performance when the source and target domains are swapped. Fig. 8-(a) and (b) illustrate an example of SHAP summary plots for such a case. The Extra-Tree model in (a) is trained on NFv2-UNSW-NB15 dataset and evaluated on NFv2-BoT-IoT dataset, and in (b) it is trained on NFv2-BoT-IoT dataset and evaluated on NFv2-UNSW-NB15 dataset.

As can be seen, in both SHAP summary plots most of the features have a zero SHAP value, indicating no impact on the model output. Though, in Fig. 8-(a) the low (blue) and high (red) values of the first important feature, MAX_TTL, have a clear correspondence with the Benign and Attack classes (low feature values have positive and high feature values have negative SHAP values). However, in Fig. 8-(b) most of the MAX_TTL values have positive SHAP values. This means that the model has almost assigned all the MAX_TTL values to the Benign class while other features have not been used in the classification at all.

Hence, while a high performance is expected for the observed *single-rule classifier* in Fig. 8-(a) (with a huge difference between the SHAP values of the low and high MAX_TTL values), it is hard to imagine a similar performance for the model in Fig. 8-(b) that has a single feature mostly indicating to the Benign class. Both of these conclusions are consistent with the results shown in Fig. 3-(a).

Finally, we investigate another case of asymmetric cross-domain performance with a different distribution of SHAP values as shown in Fig. 9. In Fig. 9-(a) the Extra-Tree model is trained on NFv2-BoT-IoT dataset and evaluated on NFv2-ToN-IoT dataset, and in (b) it is trained on NFv2-ToN-IoT dataset and evaluated on NFv2-BoT-IoT dataset.

As can be seen in Fig. 9-(a) the SHAP values of all features except the MAX_TTL is zero, i.e. they do not affect either of classes. The dataset samples with a low MAX_TTL value are classified as Benign and samples with a high MAX_TTL value are classified as Attack. In Fig. 9-(b), however, it is not possible to identify such a simple correspondence between the feature values and classes. A mix of low and high feature values can be seen in the positive and negative ranges of the SHAP value. Accordingly, it is expected that the model in Fig. 9-(a) that is equivalent to a simple single threshold classifier perform much better than the model in Fig. 9-(b). These conclusions are consistent with the results shown in Fig. 3-(a).

7. Conclusion

Machine learning (ML) based network intrusion detection systems (NIDSs) have been around for many years to address the shortcomings of signature-based NIDSs. While a large number of methods have been proposed in the academic literature of NIDS with near perfect detection and classification performances, the ML-based NIDSs rarely have been used in the real-world scenarios. Since the evaluation of these methods is predominantly domain-specific, we assume cross-domain performance evaluation is the missing link to use these models in the real world applications.

In this paper we extensively evaluate the cross-domain performance of eight supervised and unsupervised ML-based NIDSs across four recently published publicly available NIDS datasets. In these experiments, each ML-based NIDS is trained on a dataset and evaluated against all the four datasets, including the one used for its training. This makes it possible to compare a model's performance in a domain-specific and cross-domain evaluation.

The results indicate that while some models are able to generalise over one or two datasets, none of the studied models generalise well across all datasets. The other observation is that cross-domain performance can be asymmetric, which means performance

Table 10
List of all NetFlow features of four studied datasets along with indication if they are used in this study.

No.	Feature	Description	Used in This Study
1	IPV4_SRC_ADDR	IPv4 source address	No
2	IPV4_DST_ADDR	IPv4 destination address	No
3	L4_SRC_PORT	IPv4 source port number	No
4	L4_DST_PORT	IPv4 destination port number	No
5	CLIENT_TCP_FLAGS	Cumulative of all client TCP flags	Yes
6	DNS_QUERY_ID	DNS query transaction Id	Yes
7	DNS_QUERY_TYPE	DNS query type (e.g., 1=A, 2=NS..)	Yes
8	DNS_TTL_ANSWER	TTL of the first A record (if any)	Yes
9	DST_TO_SRC_AVG_THROUGHPUT	Dst to src average thpt (bps)	Yes
10	DST_TO_SRC_SECOND_BYTES	Dst to src Bytes/sec	Yes
11	DURATION_IN	Client to Server stream duration (msec)	Yes
12	DURATION_OUT	Client to Server stream duration (msec)	Yes
13	FLOW_DURATION_MILLISECONDS	Flow duration in milliseconds	Yes
14	FTP_COMMAND_RET_CODE	FTP client command return code	Yes
15	ICMP_IPV4_TYPE	ICMP Type	Yes
16	ICMP_TYPE	ICMP Type * 256 + ICMP code	Yes
17	IN_BYTES	Incoming number of bytes	Yes
18	IN_PKTS	Incoming number of packets	Yes
19	L7_PROTO	Application protocol (numeric)	Yes
20	LONGEST_FLOW_PKT	Longest packet (bytes) of the flow	Yes
21	MAX_IP_PKT_LEN	Len of the largest flow IP packet observed	Yes
22	MAX_TTL	Max flow TTL	Yes
23	MIN_IP_PKT_LEN	Len of the smallest flow IP packet observed	Yes
24	MIN_TTL	Min flow TTL	Yes
25	NUM_PKTS_1024_TO_1514_BYTES	Packets whose IP size > 1024 and <= 1514	Yes
26	NUM_PKTS_128_TO_256_BYTES	Packets whose IP size > 128 and <= 256	Yes
27	NUM_PKTS_256_TO_512_BYTES	Packets whose IP size > 256 and <= 512	Yes
28	NUM_PKTS_512_TO_1024_BYTES	Packets whose IP size > 512 and <= 1024	Yes
29	NUM_PKTS_UP_TO_128_BYTES	Packets whose IP size <= 128	Yes
30	OUT_BYTES	Outgoing number of bytes	Yes
31	OUT_PKTS	Outgoing number of packets	Yes
32	PROTOCOL	IP protocol identifier byte	Yes
33	RETRANSMITTED_IN_BYTES	Number of retransmitted TCP flow bytes (src->dst)	Yes
34	RETRANSMITTED_IN_PKTS	Number of retransmitted TCP flow packets (src->dst)	Yes
35	RETRANSMITTED_OUT_BYTES	Number of retransmitted TCP flow bytes (dst->src)	Yes
36	RETRANSMITTED_OUT_PKTS	Number of retransmitted TCP flow packets (dst->src)	Yes
37	SERVER_TCP_FLAGS	Cumulative of all server TCP flags	Yes
38	SHORTEST_FLOW_PKT	Shortest packet (bytes) of the flow	Yes
39	SRC_TO_DST_AVG_THROUGHPUT	Src to dst average thpt (bps)	Yes
40	SRC_TO_DST_SECOND_BYTES	Src to dst Bytes/sec	Yes
41	TCP_FLAGS	Cumulative of all TCP flags	Yes
42	TCP_WIN_MAX_IN	Max TCP Window (src->dst)	Yes
43	TCP_WIN_MAX_OUT	Max TCP Window (dst->src)	Yes

of the model can significantly change when the source and target domains are swapped. The last observation indicates that the unsupervised ML-based NIDSs generalise better than the supervised ML-based NIDSs, even though their domain-specific performance is lower than the supervised ML-based NIDSs.

We have further explained our results by finding the SHAP values for the model outputs. Comparing the SHAP values of different dataset-model combinations indicates that the high classification performances in a combination of the model and source-target domains is correlated with having one or more features that have strong correspondence with the Attack/Benign classes (positive and negative SHAP values). Clearly, when such a simple rule (single or multi-threshold classifier) defines a model's performance, it can hardly be generalised to another target domain where features can have different distributions. As such, the lack of generalisation of the model performance from one domain to the other, and the asymmetric behaviour of models cross-domain performance can all be attributed to the presence of these simple rule classifiers for one combination of model-source-target datasets and its absence in other combinations due to different feature distributions.

As for our future research direction, we are investigating the ML-based approaches that can deal with/compensate for this feature distribution shifts in the alternate target domain, to enhance the cross-domain performance of ML-based NIDSs.

Declaration of competing interest

One or more of the authors of this paper have disclosed potential or pertinent conflicts of interest, which may include receipt of payment, either direct or indirect, institutional support, or association with an entity in the biomedical field which may be perceived to have potential conflict of interest with this work. For full disclosure statements refer to https://doi.org/10.1016/j.compeleceng.2023.108692.Siamak Layeghy reports financial support was provided by Queensland Government.

Data availability

The data used in this manuscript is publicly available

Acknowledgements

This research is made possible by an Advance Queensland Industry Research Fellowship, grant number RM2019002409.

References

[1] Moustafa N, Slay J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: Military communications and information systems conference (MilCIS). IEEE; 2015, p. 1–6.
[2] Sharafaldin I, Lashkari AH, Ghorbani AA. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: ICISSP 2018 - Proceedings of the 4th international conference on information systems security and privacy, 2018-January. 2018, p. 108–16.
[3] University of California Irvine. KDD Cup 1999 Data. 1999, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html [Accessed: 2020-07-30].
[4] Sarhan M, Layeghy S, Moustafa N, Portmann M. NetFlow datasets for machine learning-based network intrusion detection systems. In: Deze Z, Huang H, Hou R, Rho S, Chilamkurti N, editors. Big data technologies and applications. Springer International Publishing; 2021.
[5] Lo WW, Layeghy S, Sarhan M, Gallagher M, Portmann M. E-GraphSAGE: A graph neural network based intrusion detection system for IoT. In: NOMS 2022-2022 IEEE/IFIP network operations and management symposium. IEEE; 2022, p. 1–9.
[6] Baktashmotlagh M, Harandi MT, Lovell BC, Salzmann M. Unsupervised domain adaptation by domain invariant projection, In: Proceedings of the IEEE international conference on computer vision. 2013, p. 769–76.
[7] Fan Y, Li Y, Cui H, Yang H, Zhang Y. An intrusion detection framework for IoT using partial domain adaptation. In: International conference on science of cyber security. vol. 2, Springer International Publishing; 2021, p. 36–50.
[8] Wu P, Guo H, Buckland R. A Transfer Learning Approach for Network Intrusion Detection. In: 4th IEEE international conference on big data analytics. 2019.
[9] Sarhan M, Layeghy S, Gallagher M, Portmann M. From zero-shot machine learning to zero-day attack detection. 2021, arXiv preprint arXiv:2109.14868.
[10] Rivero J, Ribeiro B, Chen N, Leite FS. A Grassmannian approach to zero-shot learning for network intrusion detection. In: International conference on neural information processing. Cham: Springer; 2017, p. 565–75.
[11] Zhao J, Shetty S, Pan JW, Kamhoua C, Kwiat K. Transfer learning for detecting unknown network attacks. Eurasip J Inf Secur 2019;(1):1–13.
[12] Al-riyami S, Coenen F, Lisitsa A. A Re-evaluation of Intrusion Detection Accuracy : an Alternative Evaluation Strategy. In: ACM SIGSAC conference on computer and communications security. 2018, p. 2195–97.
[13] Apruzzese G, Pajola L, Conti M. The cross-evaluation of machine learning-based network intrusion detection systems. IEEE Trans Netw Serv Manag 2022;1–18.
[14] Sarhan M, Layeghy S, Portmann M. Towards a standard feature set for network intrusion detection system datasets. Mobile Netw. Appl. 2022;27(1):357–70.
[15] Shapley Ll S. A value for n-person games. In: Contributions To the Theory of Games II, Annals of Mathematical Studies. vol. 28, p. 1953.
[16] Song J, Takakura H, Okabe Y. Description of Kyoto University Benchmark Data. 2006, http://www.takakura.com/Kyoto_data/BenchmarkData-Description-v5.pdf.
[17] Perona I, Gurrutxaga I, Arbelaitz O, Martín JI, Muguerza J, Pérez JM. Service-independent Payload Analysis to Improve Intrusion Detection in Network Traffic. In: 7th Australasian data mining conference, vol. 87. 2008, p. 171–8.
[18] Tavallaee M, Bagheri E, Lu W, Ghorbani AA. A detailed Analysis of the KDD CUP 99 Data Set, In: 2009 IEEE symposium on computational intelligence for security and defense applications. 2009, p. 1–6.
[19] Moustafa N. TON _ IoT Datasets for Cybersecurity Applications based Artificial Intelligence. In: Proceedings of the eResearch Australasia Conference. 2019, p. 3–5.
[20] Koroniotis N, Moustafa N, Sitnikova E, Turnbull B. Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. Future Gener Comput Syst 2019.
[21] Hosseininoorbin S, Layeghy S, Sarhan M, Jurdak R, Portmann M. Exploring edge TPU for network intrusion detection in IoT. 2021, arXiv preprint arXiv:2103.16295.
[22] Hosseininoorbin S, Layeghy S, Sarhan M, Jurdak R, Portmann M. Exploring edge TPU for network intrusion detection in IoT. 2021, arXiv preprint arXiv:2103.16295.
[23] Cawley GC, Talbot NL. On over-fitting in model selection and subsequent selection bias in performance evaluation. J Mach Learn Res 2010;11:2079–107.
[24] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in Python. J Mach Learn Res 2011;12:2825–30.
[25] Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, et al. TensorFlow: Large-scale machine learning on heterogeneous systems. 2015, Software available from tensorflow.org.
[26] Liu FT, Ting KM, Zhou Z-H. Isolation-based anomaly detection. ACM Trans Knowl Discov Data 2012;6(1). Article 3.
[27] Amer M, Goldstein M, Slim Abdennadher D. Enhancing One-class Support Vector Machines for Unsupervised Anomaly Detection. In: ACM SIGKDD workshop on outlier detection and description. 2013, p. 8–15.
[28] Zhang T. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In: The twenty-first international conference on Machine learning. 2004, p. 116–23.
[29] Lundberg SM, Lee S-I. A unified approach to interpreting model predictions, In: Neural Information Processing Systems (NIPS 2017), (Long Beach, CA, USA), 2017.