Journal of Network and Systems Management (2022) 30:12
https://doi.org/10.1007/s10922-021-09615-7

Towards Model Generalization for Intrusion Detection: Unsupervised Machine Learning Techniques

Miel Verkerken[1] · **Laurens D'hooge**[1] · **Tim Wauters**[1] · **Bruno Volckaert**[1] · **Filip De Turck**[1]

Received: 9 January 2021 / Revised: 17 May 2021 / Accepted: 28 June 2021 /
Published online: 17 October 2021

Abstract
Through the ongoing digitization of the world, the number of connected devices is continuously growing without any foreseen decline in the near future. In particular, these devices increasingly include critical systems such as power grids and medical institutions, possibly causing tremendous consequences in the case of a successful cybersecurity attack. A network intrusion detection system (NIDS) is one of the main components to detect ongoing attacks by differentiating normal from malicious traffic. Anomaly-based NIDS, more specifically unsupervised methods previously proved promising for their ability to detect known as well as zero-day attacks without the need for a labeled dataset. Despite decades of development by researchers, anomaly-based NIDS are only rarely employed in real-world applications, most possibly due to the lack of generalization power of the proposed models. This article first evaluates four unsupervised machine learning methods on two recent datasets and then defines their generalization strength using a novel inter-dataset evaluation strategy estimating their adaptability. Results show that all models can present high classification scores on an individual dataset but fail to directly transfer those to a second unseen but related dataset. Specifically, the accuracy dropped on average 25.63% in an inter-dataset setting compared to the conventional evaluation approach. This generalization challenge can be observed and tackled in future research with the help of the proposed evaluation strategy in this paper.

Keywords Intrusion detection · Network security · Cybersecurity · Unsupervised techniques · Generalization strength · Inter-dataset evaluation

✉ Miel Verkerken
Miel.Verkerken@UGent.be

[1] Department of Information Technology, IDLab, Ghent University - imec, Technologiepark 126, 9052 Gent, Belgium

1 Introduction

The rapid increase of connected devices to the internet and the accompanying amount of data sent over these interconnected networks does not only greatly expand the attack surface for people with malicious intents, but it also enlarges the potential impact in case of a successful cybersecurity breach. Over the last years, this resulted in a continuously growing risk that will most likely persist in the future. Even the global covid-19 pandemic could not stop this trend. A recent survey conducted by Gartner regarding future investments in the internet of things (IoT) proved that the majority of companies will increase their investments with the goal of reducing costs in the future [1]. This results in a further acceleration in the number of deployed devices and consequently enlarging the attack vector surface by a multitude. One component that can be used to mitigate the risk of a successful cyberattack is an intrusion detection system (IDS). The goal of an IDS is to detect any malicious activity on a network or system. Differentiation can be made based on the input source the IDS uses to detect an attack. A host-based intrusion detection system (HIDS) will be deployed on a particular system where it monitors system logs and resource usages, while a network-based intrusion detection system (NIDS) is placed on a node in the network and monitors traffic that passes through the network. While both types can be implemented in hardware or software and have the ability to detect an attack before, during, or after it has occurred, an NIDS has the advantage to simultaneously cover an entire (sub)network rather than a single system for HIDS.

Traditionally, a misuse-based NIDS is used, which usually computes a signature over each incoming and outgoing packet, which in turn is compared against a database with signatures of known attacks. When a matching signature is found, then the corresponding packet is flagged as malicious. This approach can detect known attacks with high confidence but comes with a few drawbacks. First, this approach is limited by design to detect known attacks. Furthermore, there exist multiple evasion techniques to adapt the signature of a packet in such a way it becomes infeasible to match [2]. Similarly, the recent wide adoption of encryption in network traffic, in particular of HTTPS, makes it impossible to inspect the encrypted body of the packet [3]. Recently, the inspection of encrypted traffic became an active field within the research community with promising results supported by advances in the field of deep learning [4–6]. At last, it is a labor-intensive task to continuously maintain and update a database with known attacks [7].

In the last decade, research shifted its focus towards anomaly-based intrusion detection empowered by the developments within the field of machine learning. Machine learning models can take advantage of big amounts of data to learn certain patterns and apply these to future inputs. Two main learning techniques exist: supervised and unsupervised. Contrary to supervised, unsupervised techniques do not require labeled data for training, eliminating the need for an expensive human expert, as they mostly try to model benign behavior, and everything that differs too much from this model gets flagged as malicious. Often these techniques are not applied to the raw network packets, but instead, the traffic is aggregated into bidirectional flows of packets between the same source and destination. Over these bidirectional flows, multiple statistical features are computed in both forward as backward directions. For example, the number of packets, duration, and the average length of a packet. There are multiple advantages of transforming the raw network traffic into flows. For example, it enables a more high-level overview of the current situation as it is not limited to a single packet, besides, it also greatly reduces the amount of data to transmit in the case of a centralized fraud detector, limiting the bandwidth requirements [8].

Recent literature [9, 10] advocates the adoption of flow-based NIDS relying on unsupervised machine learning techniques in the real world because of their potential to detect zero-day attacks, as long as the attack substantially differs from normal behavior, in combination with the efficient usage of high volumes of data. Despite the extensive research efforts and promising reported results, no or limited adoption in real-world applications is found until now. This can have multiple possible causes. One of them is a generalization problem, indicating that a trained model excels in an experimental setup but does not deliver the same observed classification power on new, unseen samples. An alternative approach to evaluate the generalization power of a model is proposed in this paper that goes one step further than the commonly accepted strategy by using two different but related datasets instead of only a single dataset. Using this inter-dataset strategy, the first dataset is employed for training and validation, and the second for the final unbiased evaluation. This way, a model overfitting on a specific dataset will yield inferior results on the second dataset and urge researchers to develop well generalizing models that most probably perform better in a real-world environment.

The main contributions of this paper are three-fold. First, the evaluation and analysis of four promising unsupervised models (autoencoder, one-class SVM, isolation forest, and principal components analysis) on two recent and realistic datasets (CIC-IDS-2017 and CSE-CIC-IDS-2018). Second, a novel inter-dataset evaluation strategy is introduced to evaluate the generalization strength of newly proposed models. Third, this novel strategy is employed to estimate the generalization strength of the four unsupervised models without prior adaptation and compared with the conventional intra-dataset baseline.

The remainder of this paper is structured as follows. Section 2 starts with giving an overview of related work in the field of NIDS and available datasets to use for the evaluation of the proposed models. Next, the methodology is presented to enable reproducible results together with the inter-dataset evaluation strategy in Sect. 3. Section 4 presents the experimental results which are then extensively discussed in Sect. 5. Before drawing a conclusion in Sect. 7, possible future work is listed in Sect. 6.

2 Related Work

This section starts by describing the recent literature conducted within the field of anomaly-based NIDS, highlighting the many techniques and algorithms together with their sometimes remarkable classification scores. Next, these results reported by researchers in an experimental setup are questioned for their real-world relevancy from both the broad perspective of artificial intelligence (AI) to the specific field of network intrusion detection. This section is concluded by a brief overview of the available datasets suitable for the evaluation of NIDS.

The first IDS has already been proposed in 1980 [11], but since the rapid advances in the field of machine and deep learning of the last decade, research interest has surged with researchers applying these techniques to develop novel IDS. Multiple surveys give a good review of the developments in the IDS field [12–15]. They advocate the use of unsupervised techniques to overcome the limitations of available realistic datasets and for their ability to detect unseen, zero-day attacks. More specifically, the following studies confirmed the use of autoencoders [16, 17], isolation forest [18], one-class SVM [19] and principal components analysis [20] for anomaly detection with promising results but often request real-world validation in their future work. Otoum et al. [21] recently proposed a hybrid IDS that takes advantage of a traditional signature-based IDS for known attack detection combined with a anomaly-based detection for unknown attacks. Most of the proposed machine learning IDS use a flow-based classification approach. Because these flows are only constructed from the packet headers, they are not sensitive to encryption protocols preventing inspection or classification [22]. Khatouni et al. [23] uses four open-source traffic flow analyzers to extract and/or construct traffic flow level features from packet traces. Using these flow features they successfully identified known services from multiple encrypted service channels.

All these studies have in common that they achieve excellent classification performance and promote their models for real-world usage, yet limited adoption of these techniques is found in operational applications. Already 10 years ago, this problem was raised by the research community [24] but with little effect. Recently, a paper by a collaborative effort from Google recognized this as a global challenge and named it *underspecification*. Additionally, more specifically within the field of network intrusion detection, the remarkable high results published in the literature have been openly questioned by Leevy et al. [25]. Furthermore, the survey stresses the importance of documenting all the taken steps for reproduction purposes. The survey by Ahmad et al. [26] also questions the low performance of machine learning IDS in the real-world and requests an effective method to validate the generalization performance in their future work. A recent study by Al-Omare et al. [27] tries to limit the risk of overfitting by ranking the used features and only selecting the top-performing ones with as additional benefit of reducing the computational complexity because of the lower input dimension. Similarly, Aloqaily et al. [28] proposed a hybrid intrusion detection system, named D2H-IDS, which uses a deep belief network for dimensionality reduction and a decision tree for the attack classification. The work conducted in this study tries to close the gap in classification performance between academic prototypes and operational applications by proposing an alternative evaluation strategy to estimate the generalization strength of suggested models in the recent literature and initiate future research on this topic.

The training and evaluation of machine learning-based IDS require a lot of data. Preferably, this data should be sampled from the real-world and representative for future inputs. This often proves to be a challenge as network traffic contains sensitive data and has obvious privacy concerns. One way to overcome this barrier is by anonymizing the data to prevent any information from being traced back to an individual person. This often includes operations like data aggregation or removing and modifying certain features. History has proven that this is a tedious task. The Netflix Prize is a popular example of where it went wrong. The world's largest streaming service publicly released a dataset in 2006 containing movie ratings of 500,000 subscribers. Only after a few weeks, the anonymization algorithm was already broken and sensitive information of individual users was exposed [29]. Another way to obtain realistic network traffic is by generating a synthetic dataset. Instead of collecting data from a network and its actual users with their related challenges, an experiment can be set up using a wide range of techniques to imitate them. This approach only works if the used techniques approximate benign behavior close enough.

Limited realistic datasets exist for intrusion detection. The most widely used dataset is KDDcup99 [30], created by the Defense Advanced Research Projects Agency (DARPA) for the fifth Conference on Knowledge Discovery and Data Mining in 1999. Ten years after its release, Tavallaee et al. [31] published an updated version, NSL-KDD, together with a comprehensive analysis specifying the various issues in the original dataset. The revised dataset solved many of the demonstrated shortcomings [32], but now, more than 20 years later, the used network protocols and attacks are no longer representable for modern communication networks. In the last decade, the Canadian Institute for Cybersecurity (CIC) became the front runner regarding the generation of realistic network intrusion detection datasets. Sharafaldin et al [33] proposed 11 criteria needed to create a reliable intrusion detection dataset for benchmarking. The first dataset satisfying all 11 criteria is CIC-IDS-2017 and collects real network traffic using multiple internet protocols such as HTTP(s), FTP, SSH, IMAP, and POP3 for a duration of 5 days [34]. Unique about this dataset is that all benign traffic is generated by a B-Profile system imitating human interaction. One year later, CSE-CIC-IDS-2018 was published by the CIC in collaboration with the Communications Security Establishment (CSE). This dataset took the same principles of CIC-IDS-2017 but was executed on a larger scale network in the cloud of Amazon Web Services (AWS). Both datasets are distributed as raw network packets (PCAP) or bidirectional flows aggregated from the PCAP files by CICFlowMeter [35, 36]. The latter enables easy use in IDS employing machine learning techniques.

3 Methodology

This section is divided into subsections regarding the multiple aspects to reproduce the reported results in Sect. 4. First, Sect. 3.1 discusses the used datasets together with the necessary steps from data cleaning to feature reduction required to prepare the raw data. Next, the evaluated algorithms and their optimized hyper-parameters are described in Sect. 3.2. Sect. 3.3 describes the evaluation strategy, including the alternative inter-dataset evaluation strategy proposed in this paper to estimate the generalization strength of a model. The used metrics to report the performance of the models are documented in Sect. 3.4. Finally, the used hardware for the execution of the experiments is reported in Sect. 3.5.

3.1 Datasets

For the novel inter-dataset evaluation strategy, a minimum of two datasets are needed. It is important that these datasets contain identical features and are related, more specifically for unsupervised binary classification, the benign traffic (X, y=0) should be sampled from the same distribution (D), see equation 1. This assumption allows to evaluate a model trained on a first dataset, on a second related dataset as the learned normal behavior theoretically should still be applicable and transferable. Ideally, a well-generalized model would classify the benign samples of both datasets with a similar classification performance. Key is that all data goes through the same preprocessing pipeline, preserving the identical set of features.

$$\begin{aligned} X, y &\sim D_1 \\ X, y &\sim D_2 \\ D_1 = D_2 \quad &\forall X \text{ if } y = 0 \end{aligned} \tag{1}$$

The Canadian Institute for Cybersecurity (CIC) developed a system called B-Profile to produce benign background traffic. These profiles derived abstract behavior from a group of real users using machine learning and statistical analysis techniques. In 2017, the CIC published the first dataset, CIC-IDS-2017, using this technique. One year later, they published a second dataset in collaboration with the Communication Security Establishment (CSE), CSE-CIC-IDS-2018, which used the same B-Profiles to generate benign traffic. The second experiment took the generation process to a larger scale by bringing the whole network setup to the Amazon Web Services (AWS) cloud computing platform and using a total of 500 machines instead of only 14 in the first iteration. Because both datasets are generated using the same tools and processes while only differing the deployment environment, they satisfy all the requirements to be used in the novel intra-dataset evaluation strategy. The datasets are distributed as raw network packets (PCAP) as well as machine learning-friendly CSV files containing bidirectional flows (biflows) extracted from the PCAPs by CICFlowMeter [36]. These biflows contain next to a few flow identification features such as source and destination IP-address, also a timestamp, 80 statistical network features, and a label. This label contains "Benign" for the background traffic or the name of the attack class for malicious traffic and is only used during hyper-parameter selection and the final evaluation of the classification performance. Table 1 gives an overview of the number of occurrences for each attack class in both datasets. Important to highlight is the substantial class imbalance between the benign and multiple attack classes. Furthermore, not all classes are as present in the 2018 dataset as in the 2017 one or vice versa. Besides, even two classes, *heartbleed* and *port scan*, entirely disappeared and are not explicitly present anymore. Nevertheless, the benign class stays the most present class in both datasets with 80.32 and 84.59 percent, respectively. The class imbalance does not affect the training of the models because this paper employs a semi-supervised learning approach with only benign traffic, but can substantially influence the validation and test metrics if not carefully defined.

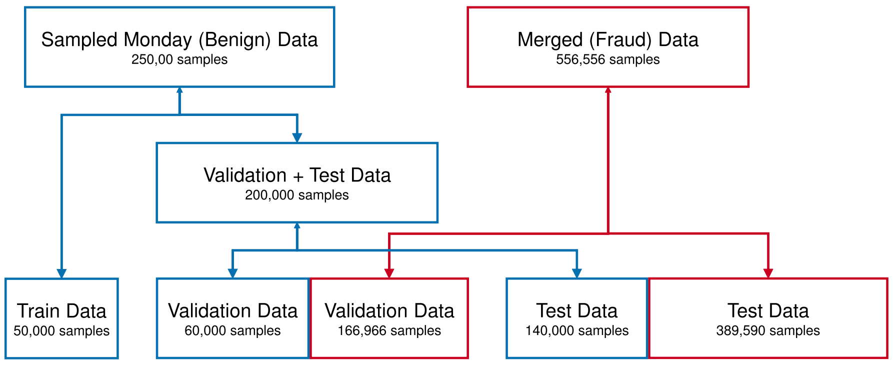
Table 1 Attack classes and their number of occurrences in CIC-IDS-2017 and CSE-CIC-IDS-2018

Attack class	Details	CIC-IDS-2017		CSE-CIC-IDS-2018	
		Count	Pct (%)	Count	Pct (%)
Benign	–	2,271,320	80.32	7,364,941	84.59
(D)DOS	Hulk	230,124	13.43	145,199	11.17
	DDOS	128,025		775,955	
	GoldenEye	10,293		41,406	
	DoS slowloris	5,796		9,908	
	Slowhttptest	5,499		43	
Port scan	–	158,804	5.62	–	–
Brute force	FTP-Patator	7,935	0.49	53	1.08
	SSH-Patator	5,897		94,041	
Web-attack	Brute Force	1,507	0.08	555	0.01
	XSS	652		227	
	SQL Injection	21		79	
Botnet	–	1,956	0.07	144,535	1.66
Infiltration	–	36	<0.01	129,786	1.49
Heartbleed	–	11	<0.01	–	–

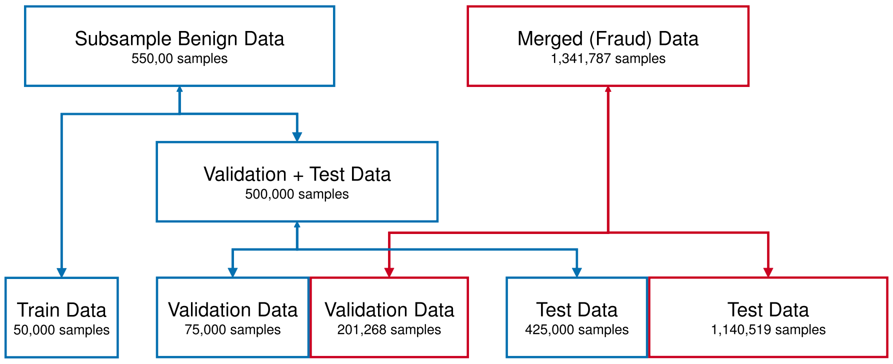
Figure 1 visualizes the applied train, validation, and test split strategy for both datasets. First, benign and malicious traffic are independently collected. Important is that these sets are clear of rows with missing values and do not contain any duplicates. Next, the benign dataset is subsampled without replacement to 250,000 flows for CIC-IDS-2017 and 550,000 flows for CSE-CIC-IDS-2018, from which 50,000 are used as the training set. The remaining benign together with all malicious flows are stratified sampled over validation and test set with a 30/70 for the 2017 and a 15/85 ratio, to retain a similarly sized validation set, for the 2018 dataset. As a result, a train, validation, and test set are obtained that are practical to use and for which the i.i.d. requirement is fulfilled.

3.1.1 Preprocessing Pipeline

The preprocessing pipeline transforms the raw data into the right format to be consumed by the machine learning model. Figure 2 visualizes the five steps of the pipeline. The first three steps clean the data by removing columns with redundant information, dropping rows with missing or infinity values, and eventually, the resulting dataset is filtered from duplicates. More specifically, step two removed a duplicate column found in the 2017 dataset, removed ten features without any variance, and six features that could lead to overfitting due to the dataset generation setup such as IP addresses and timestamps. Only very limited rows contained missing or infinity values, therefor these rows are being dropped rather than filled with techniques such as imputation. This way the information loss is minimized and guarantees original and high-quality in the next phases. After the cleaning phase, the resulting collection



(a) CIC-IDS-2017



(b) CSE-CIC-IDS-2018

Fig. 1 Train, validation and test split strategy for the datasets

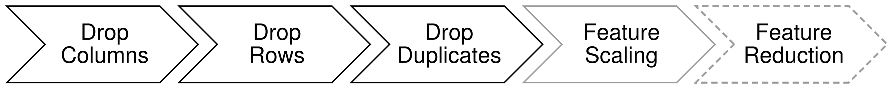


Fig. 2 The preprocessing pipeline consisting of five steps of which the last one is not employed for the algorithms relying on a reconstruction error as anomaly score

of flows contains 67 features, see appendix A for an overview. In the following feature scaling step, the input features are normalized. Because many methods exist for the normalization of the range of the data, the used method is added as a hyperparameter to the optimization problem. Following, four techniques with their implementations from the scikit learn library [37] are included: *StandardScaler, RobustScaler, QuantileTransformer, and MinMaxScaler*. At last, the final feature reduction step is only applied for the *isolation forest* and *one-class svm* since both *principal components analysis* and *autoencoder* algorithms rely on a reconstruction error as anomaly score and already apply a feature reduction technique internally. Because this paper evaluates unsupervised techniques, a PCA transformation is employed to reduce the high-dimensional to a lower-dimensional feature space without the need for any labels. The number of principal components of the resulting transformation is also added as a hyper-parameter to the optimization problem. To summarise, the first three steps cleaning the data can be applied globally while the last two with their corresponding hyper-parameters are included in the optimization phase of a particular algorithm.

3.2 Algorithms

In this section, the used anomaly detection algorithms are presented, together with the used anomaly score, and the hyper-parameters that are optimized. For all anomaly scores discussed in this section applies, the higher the score, the higher the probability for a sample to be malicious. For each algorithm, a work in the literature is referred to as a starting point for a more thorough explanation of their internal workings.

3.2.1 Principal Component Analysis

Principal components are the sequence of vectors that are linearly uncorrelated after an orthogonal transformation of an original, possibly correlated feature set. The process of calculating this transformation and executing it is called *Principal Component Analysis (PCA)* [38, 39]. This technique is often used for feature reduction by only retaining the components that explain the most variance of the original data. In this paper, PCA is also employed for anomaly detection by first performing a PCA transformation followed by the inverse transformation, to reconstruct the original data. As an anomaly score, the reconstruction error is used by computing the sum of squared errors (SSE) between the input and output vector. Without removing the components that explain the least variance, this would be a loss-less operation with an SSE equal to zero. However, when the PCA transformation is fitted on only benign data in combination with a reduction in principal components, then the malicious samples will yield a higher reconstruction error and thus anomaly score as long as the assumption that malicious traffic differentiates from normal behavior is satisfied. The number of principal components is the only hyper-parameter that needs to be optimized. For the implementation, the scikit learn library is used [37].

3.2.2 Isolation Forest

An *isolation forest* attempts to isolate anomalies in high-dimensional data instead of trying to model normal behavior and does this in linear time with low memory demands [40]. Therefore, it builds an ensemble of binary trees with each tree constructed from a random subsample of the training data. While the algorithm tries to isolate anomalies, it does not require them to be present during the training phase. For each sample, the average depth over all trees can be computed, which is equivalent to the number of splits that are needed to isolate that sample. As anomalies are assumed to be different from the benign data on which the trees are built, they will reside higher in the tree and thus have a lower average depth. As anomaly score, this average depth is negated so that a higher score results in a higher probability to be malicious. The number of trees, subsample rate, and the number of features to construct an individual tree are the hyper-parameters tuned during optimization. The scikit learn implementation is used [37].

3.2.3 Autoencoder

An *autoencoder* (AE) is an artificial neural network coming from the field of deep learning, consisting of an encoder and decoder. The encoder transforms an input vector to a lower-dimensional or latent space, after which the decoder will reconstruct the original input vector as close as possible [41]. An AE with a single hidden layer and a linear activation function results in a linear transformation which is equivalent to PCA with the number of principal components equal to the number of nodes in the single hidden layer [42]. On the other hand, when multiple hidden layers or a non-linear activation function are used, the transformation becomes nonlinear. As the anomaly score, the reconstruction error is used between the input and output vector computed as SSE. The number of hidden layers, number of neurons per layer, activation function, and regularisation terms are the optimized hyperparameters. For the implementation the easy-of-use yet powerful *Keras* [43] framework is used which itself is built on top of *Tensorflow* [44].

3.2.4 One-Class SVM

One-class SVM works similar to standard support vector machines, but instead of separating two classes with a hyper-plane, it encloses a single class with a hypersphere. As the standard SVM normally tries to find this hyper-plane with the largest possible separation margin, this translates to the smallest possible hyper-sphere for the unsupervised variant [45]. Some instances may be violating this separation to account for noise in the data and by using a kernel function a complex, non-linear boundary can be created for the sphere. The following kernel functions are examined: linear, polynomial, radial basis function (rbf), and sigmoid. The hyper-parameters to optimize are the kernel function, the kernel coëfficient and a regularization parameter ν. The parameter ν controls the number of margin errors by putting an upper bound on it. For the implementation, the scikit learn library is used [37].

3.3 Evaluation Setup

This study evaluates the generalization strength of all proposed models on basis of two different strategies. First, the commonly accepted approach to prevent overfitting and train models with high generalization power splits a single dataset in a train, validation, and test set in such a way that the independent and identically distributed (i.i.d.) assumption is satisfied. Afterwards, the internal- and hyper-parameters of the model are obtained by, respectively, using the training and validation split, often combined in a cross-validation manner. Finally, the unseen test data is used to provide an unbiased evaluation of the final model's performance. This generally results in a low risk of overfitting and a model with a good generalization strength but is still strongly dependant on the quality of the used dataset [46]. For the remainder of this paper, this strategy is called *intra-dataset* and is visualised in Fig. 3.

Secondly, this study proposes an alternative evaluation strategy employing two different but related datasets. Each dataset still gets split into three parts (train, validation, and test) according to the same rules as mentioned before. Afterwards, multiple instances of the same model with different hyperparameters are trained from the train set. The validation set is used to select the best performing model with its corresponding hyperparameters of these instances. At this point, the exact same approach is used as in the intra-dataset strategy, but instead of defining the final performance of the model on the test set of the same dataset, a test set of a second related dataset is used. With two datasets, the inter-dataset strategy can be executed twice, once train/validate on dataset A, testing on dataset B, and once the inverse. This strategy is especially valuable when using synthetically generated datasets, as often the case for problems containing sensitive information such as IDS. Models trained on a dataset containing (hidden) features specific to the generation process and highly correlated with the label will result in excellent results on that particular dataset but will fail to generalize those to a second dataset. This strategy brings the estimation of the generalization strength of a model one step closer to real-world evaluation. For the remainder of this paper, this strategy is called *inter-dataset* and is accordingly visualized in Fig. 3.

For both strategies, it is important that the hyperparameters are fully optimized to select the best model. The open-source optimization framework *Optuna* [47] is used for the implementation. The tree-structured parzen estimator (TPE) is used to search efficiently through large spaces of possible values by selecting the next combination of hyperparameters based on the highest expected improvement. All the code can be retrieved from the GitLab repository [48].

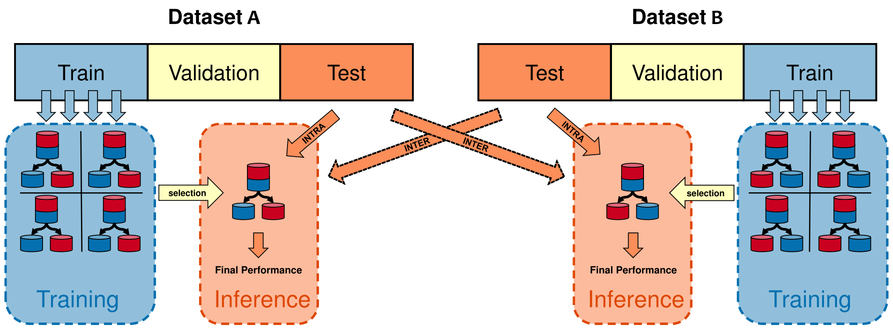


Fig. 3 Graphical representation of the common intra- and novel inter-dataset evaluation strategy proposed in this study to estimate the generalization strength of machine learning models

3.4 Metrics

The models in this study are evaluated in terms of classification performance and computational complexity. The latter is simply documented as the time needed to train the model and to evaluate the test set in seconds. To discuss the used metrics for the evaluation of the classification performance, first, four terms need to be introduced: true positive (TP), false positive (FP), true negative (TN), and false negative (FN). The positive class resembles the fraudulent flows while the negative class represents benign flows, more specifically TP are the malicious flows flagged as such, FP are benign flows flagged as fraud, TN are benign flows classified as normal and FN are malicious flows that stay undetected. A good anomaly detector maximizes the TP while having as few FN as possible. Many metrics are derived from these four situations.

3.4.1 Accuracy

The accuracy is defined as the proportion of correctly classified samples out of the total number of predicted samples. In the use-case of IDS this translates to the sum of the correctly detected malicious and benign flows, divided by the total number of classified flows, see equation 2.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

3.4.2 Recall

The recall or true positive rate (tpr) is defined as the fraction of positive samples that are successfully detected. Equation 3 defines how to compute the recall in function of TP, FP and FN. Intuitively, the recall can be interpreted as the fraction of attacks the IDS effectively detects.

$$recall = \frac{TP}{TP + FN} \tag{3}$$

3.4.3 Precision

Precision is defined as the fraction of the actual positive samples out of all the predicted positive samples. Equation 4 defines how to compute the precision in function of TP and FP. In the case of IDS, the precision can be interpreted as the fraction of actual attacks among all the raised alarms.

$$precision = \frac{TP}{TP + FP} \tag{4}$$

3.4.4 F1 Score

The F1 score combines both the recall and precision into a single measurement and is computed by taking the harmonic mean of the two, see equation 5. This single value enables easy comparison between different evaluated techniques and related work. The reported F1 score in this study is defined by the threshold on the anomaly score that yields the maximum score. This threshold is required to classify a sample either as benign or fraud if its anomaly score is, respectively, lower or higher.

$$F_1\ score = \frac{2 * recall * precision}{recall + precision} \tag{5}$$

3.4.5 Area Under Receiver Operating Characteristic Curve

The receiver operating characteristic curve plots the recall (or tpr) in function of the false positive rate (fpr) and visualizes the discrimination ability of a binary classifier with a varying threshold on the anomaly score. The area under the receiver operating characteristic curve (AUROC) summarizes this to a single metric. The maximum score of 1 is equivalent to a perfect classifier while a completely random classifier would result in a score of 0.5. The AUROC can intuitively be interpreted as the probability that a higher anomaly score will correctly be assigned to a randomly selected positive sample than to a randomly selected negative sample. This metric is suggested to be used for imbalanced problems such as fraud detection [49] because both the tpr and fpr used to construct the curve are fractions and thus independent of possible class imbalances in the dataset.

3.4.6 Area Under Precision-Recall Curve

The precision-recall curve plots the precision in function of the recall and shows the trade-off between them for different thresholds on the anomaly score. Similar to the AUROC, the area under the precision-recall curve (AUPR) summarizes the area under the curve to a single value. A high score is achieved when the model classifies most of the frauds (i.e. high recall) while making few false alarms (i.e. high precision).

3.5 Hardware Setup

Each experiment for a combination of an algorithm and dataset is submitted to a job-based distributed platform. Each job starts in an isolated container built upon a basic Python 3.7 Docker image with all the needed libraries preinstalled and receives dedicated resources for maximum performance and objective benchmarking. Each job was assigned 4 CPUs, Intel(R) Xeon(R) Silver 4108 CPU @

1.80GHz, for parallel hyperparameter optimization and 16 GB of RAM. All jobs running an AE on the Tensorflow platform were also assigned a GPU, GeForce GTX 1080 Ti, with 11 GB dedicated RAM.

4 Results

This section presents the results of both the intra- and inter-dataset evaluation strategy on CIC-IDS-2017 and CSE-CIC-IDS-2018 for all four unsupervised algorithms in order of increasing the average classification strength on the individual datasets. The AUROC score is used as the main evaluation metric during analysis. For completeness and in order to allow easy comparison with related work in the literature, the area under precision-recall (AUPR), accuracy, F1 score, and its corresponding precision and recall are also reported. How these metrics are computed is documented in Sect. 3.4. An overview of the classification scores is given in Table 2. The dataset column notes the training dataset followed by the test dataset. In the case of the inter-dataset evaluation, these datasets will differ while they will be equal for the intra-dataset evaluation strategy. Next to the classification performance also the computational complexity for both training and inference is highlighted, see Table 3 for an overview.

Table 2 Overview of the binary classification performance on both individual and inter-dataset evaluation

Algorithm	Dataset	Recall	Precision	F1	Accuracy	AUPR	AUROC std
PCA	2017–2017	0.9435	0.9346	0.9390	0.9098	0.9677	0.9373 ± 0.0004
	2018–2018	0.9481	0.8752	0.9102	0.8637	0.9041	0.8494 ± 0.0004
	2017–2018	0.7780	0.7762	0.7771	0.6748	0.8021	**0.6661 ± 0.0005**
	2018–2017	0.9951	0.7513	0.8562	0.7541	0.8670	0.6343 ± 0.0008
Isolation Forest	2017–2017	0.9314	0.9470	0.9391	0.9111	0.9831	0.9584 ± 0.0003
	2018–2018	0.9107	0.9223	0.9165	0.8790	0.9477	0.9055 ± 0.0003
	2017–2018	0.3562	0.7573	0.4845	0.4479	0.7688	0.6429 ± 0.0006
	2018–2017	0.8218	0.7204	0.7678	0.6343	0.8471	0.5883 ± 0.0008
Autoencoder	**2017–2017**	0.9778	0.9459	0.9616	0.9426	0.9911	**0.9775 ± 0.0002**
	2018–2018	0.9164	0.9144	0.9154	0.8766	0.9638	0.9200 ± 0.0002
	2017–2018	0.9025	0.7499	0.8191	0.7097	0.7580	0.6434 ± 0.0006
	2018–2017	0.8307	0.7184	0.7705	0.6360	0.8476	0.5751 ± 0.0008
One-Class SVM	2017–2017	0.9920	0.9104	0.9495	0.9223	0.9890	0.9705 ± 0.0002
	2018–2018	0.9323	0.9268	0.9296	0.8898	0.9741	**0.9420 ± 0.0004**
	2017–2018	0.9305	0.7748	0.8455	0.7523	0.8010	0.6412 ± 0.0005
	2018–2017	0.9993	0.7363	0.8479	0.7363	0.9245	**0.7739 ± 0.0007**

The bold text in the table visually emphasizes the best score achieved for each evaluation setup

Table 3 Overview of the computational complexity of training and inference for CIC-IDS-2017 and CSE-CIC-IDS-2018

Algorithm	Dataset	Training std (s)	Inference std (s)
PCA	2017	0.62 ± 0.02	7.84 ± 0.02
	2018	1.16 ± 0.18	13.83 ± 2.02
Isolation Forest	2017	2.59 ± 0.011	25.3 ± 0.058
	2018	3.62 ± 0.16	79 ± 3
Autoencoder	2017	77.0 ± 29.1	16.11 ± 0.155
	2018	103 ± 31	18.62 ± 0.07
One-Class SVM	2017	86 ± 0.086	32 ± 0.078
	2018	73.0 ± 0.1	1192 ± 104

4.1 Principal Component Analysis

The PCA model only had a single hyper-parameter to train, its number of principal components, together with selecting the most suited normalization technique to rescale the features. The model trained on the 2017 dataset employing a quantile scaler in conjunction with 32 components resulted in the highest AUROC of 0.9373. A maximum F1 score of 0.9390 with a corresponding recall and precision of 0.9435 and 0.9346, respectively, is achieved with a threshold of 0.700 on the SSE. On the 2018 dataset, the robust scaler in combination with 51 principal components proved best with an AUROC of 0.8494. A cutoff on the SSE of 0.006 resulted in the maximum F1 score of 0.9102 with a corresponding recall of 0.9481 and precision of 0.8752. These best performing models on each dataset are subsequently used to evaluate the other dataset without any prior adaptation in inter-dataset evaluation strategy. The model trained on 2017 and evaluated on 2018 was able to still achieve an AUROC of 0.6661, the highest absolute AUROC of all four analyzed algorithms. On the other hand, the model trained on 2018 and evaluated on 2017 achieved an AUROC of 0.6343. This results in an average decrease of 27.13% in AUROC while the accuracy drops 19.26% between the intra- en inter-dataset evaluation.

4.2 Isolation Forest

During optimization, the number of principal components in the last step of the preprocessing pipeline, the number of trees with their sample rate, and the fraction of the features used to construct a single tree are defined. On the 2017 dataset rescaled by the quantile scaler with 15 components, an isolation forest consisting of 58 trees with a sample rate of 0.9612% but only consisting of 6 features per tree, yielded the highest AUROC of 0.9584. With a threshold of -0.0305 on the anomaly score, a maximum F1 score of 0.9391 is obtained, compounded by a recall and precision of, respectively, 0.9314 and 0.9470. The model trained on the 2018 dataset rescaled with the min-max scaler followed by a PCA transformation with 5 principal components yielded an AUROC of 0.9055 for an isolation forest with 62 trees, a sample rate of 86.47% and only 3 features per tree. The highest F1 score of 0.9165 is obtained with a threshold of -0.2837. The corresponding recall and precision are 0.9107 and 0.9223. Without any prior adaptation, the model trained on the 2017 dataset still achieves an AUROC of 0.6429, while the model trained on 2018 yields 0.5883 as AUROC. This is an average decrease of 33.98% of AUROC and 39.34% drop in accuracy between intra- and inter-dataset evaluation.

4.3 Autoencoder

Aside from the architecture of the autoencoder also the hidden activation function, l2 regularisation term, and scaler are defined during the optimization phase. An autoencoder constructed with relu as hidden activation function, an l2 regularisation term of 4.945e-5, and an architecture with 5 hidden layers of 56, 54, 51, 54, and 56 nodes, combined with the quantile scaler yielded the maximum AUROC of 0.9775, the highest score of all four algorithms. An F1 score of 0.9616 with a corresponding recall of 0.9778 and precision of 0.9459 is obtained with a threshold of 11.58 for the reconstruction error. The min-max scaler together with an artificial neural network with a relu hidden activation function, 1.66e-4 as l2 regularisation term, 9 hidden layers, and a consecutive number of neurons per layer of 57, 51, 42, 40, 13, 40, 42, 51 and 57 yielded the maximum AUROC of 0.9200 on the 2018 dataset. The maximum F1 score is 0.9154, composed of a recall of 0.9164 and a precision of 0.9144 when using 0.0094 as the threshold for the reconstruction error. In the inter-dataset setting, the model trained on the 2017 dataset decreased with 34.18% to an AUROC of 0.6434. Similarly, the model trained on the 2018 dataset decreased with 37.49% to an AUROC of 0.5751 when evaluated on the 2017 dataset. As a result, the AUROC score decreased on average by 35.84% while the accuracy only dropped 26.08%.

4.4 One-Class SVM

A one-class SVM model has next to its hyper-parameters the used kernel function with its coëfficient and regularization parameter ν, also the used scaler to normalize the features and the number of components for feature reduction to optimize. The best parameters found on the 2017 dataset are a rbf kernel with 0.1318 as coëfficient, ν equal to 0.0358, quantile scaler, and 19 principal components. These parameters result in an AUROC of 0.9705 and 0.9495 as the maximum F1 score compounded by a recall and precision of 0.9920 and 0.9104 when a threshold on the anomaly score of -0.1807 is used. For the model trained on the 2018 dataset, an AUROC of 0.9420 is achieved when using the quantile scaler, followed by a feature reduction with 34 components and a model with an rbf as the kernel function, 0.7496 as its coëfficient and ν of 0.0035 as a regularisation term. This is the highest score achieved on the 2018 dataset. With a threshold of 8.284e-4 on the anomaly score, the maximum F1 score is reached of 0.9296 with a corresponding 0.9323 as recall and 0.9268 as precision. On the contrary, the one-class SVM has up to two orders of magnitude lower inference throughput than the other algorithms with a similar training time as the autoencoder. A decrease of 33.93% is obtained when evaluating the model trained on the 2017 dataset without prior adaptation on the 2018 dataset, resulting in an AUROC of 0.6412. On the contrary, the model trained on the 2018 dataset only decreased 17.85% in the inter-dataset evaluation to an AUROC of 0.7739, this is the best result of all four algorithms on the inter-dataset evaluation strategy. On average, the AUROC and accuracy still decreased respectively 25.89% and 17.84%.

5 Discussion

Results have been presented in Sect. 4 but are focused on documenting the most important and interesting ones. This section analyses those results in more depth and discusses multiple findings. First, the individual or baseline results are discussed before moving forward to the analysis of the inter-dataset evaluation strategy.

5.1 Intra-Dataset Evaluation

A summary of the intra-dataset classification performance is given in the first two rows for each algorithm in Table 2. The AUROC scores on CIC-IDS-2017 range from 0.94 to 0.98 with the best performing algorithm being the autoencoder, closely followed by the one-class SVM. Even a slight improvement in the AUROC will

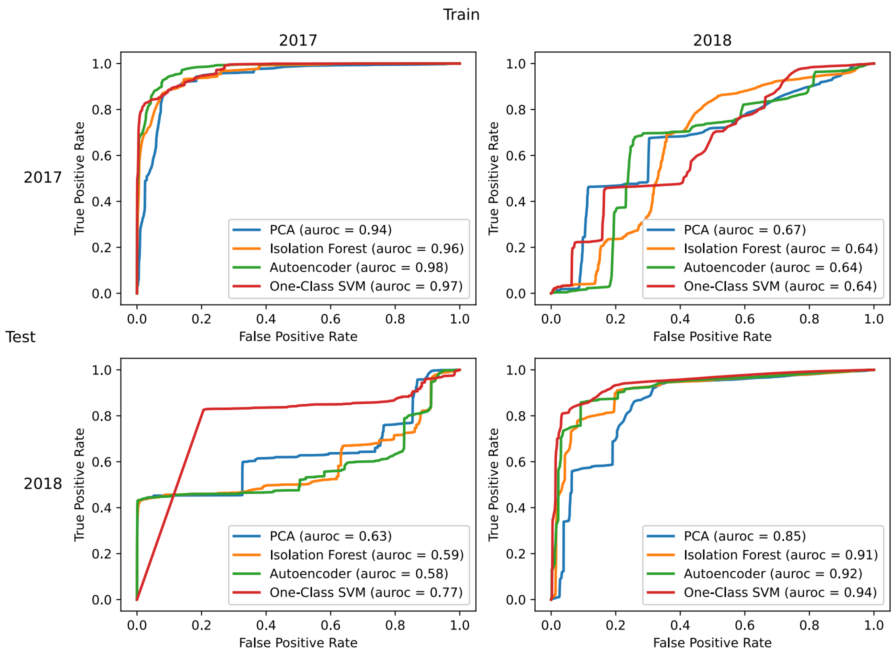
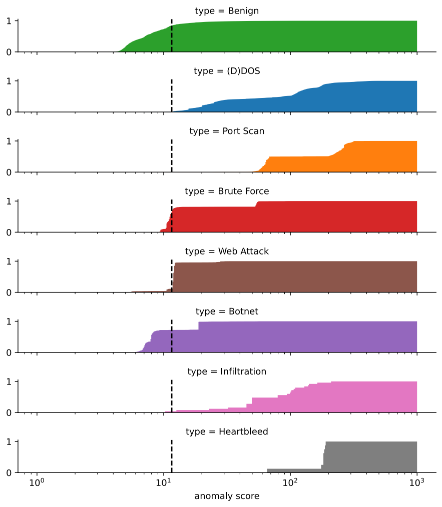


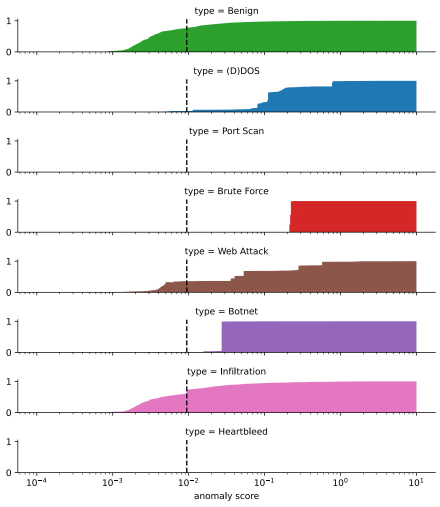
Fig. 4 The receiver operating characteristics (ROC) curves plotted for evaluated algorithms in a grid with on the x-axis and y-axis the used dataset respectively for training and testing

result in a significant decrease in false positives or an increase in true positives which in result creates a better classifier. In the left-top corner of Fig. 4 the ROC curves are plotted of the best performing model for each of the four algorithms on the 2017 dataset. There are 2 scenarios visible in which the model outperforms the rest. First, when a low fpr is key, at the cost of missing a few attacks, the one-class SVM achieves the highest score. On the contrary, when the priority is detecting all attacks at the cost of more false alarms, the autoencoder prevails. Furthermore, the few samples of heartbleed, which can be used as a proxy for zero-day attacks, are reliably classified as fraud in CIC-IDS-2017, see the bottom graph in Fig. 5a. Therefore, unsupervised NIDS proved their ability to detect unknown attacks as long as the malicious traffic differs from benign traffic.

The classification performance for all algorithms is significantly lower on the CSE-CIC-IDS-2018 dataset with AUROC scores between 0.85 and 0.94 with as best performing algorithm the one-class SVM followed by the autoencoder. Figure 5 plots the cumulative distribution of the anomaly score for each of the attack classes and the benign traffic side-by-side for both datasets for easy comparison between the attack classes as well as between the datasets. The dashed line represents the employed threshold on the anomaly score used to obtain the maximum F1 score. All flows with an anomaly score smaller than the threshold left of the dashed line are marked as normal, while flows with a higher anomaly score than the threshold are classified as fraud. The graphs are plotted using the data of the autoencoder, similar graphs for the other algorithms are accessible online at https://gitlab.ilabt.imec.be/mverkerk/cic-ids-2018. Analysis of Fig. 5 shows two root causes for the decline in classification performance between the 2017 and 2018 dataset. First, the port scan attack class, which proved easily detectable in the 2017 dataset, disappeared entirely



(a) CIC-IDS-2017



(b) CSE-CIC-IDS-2018

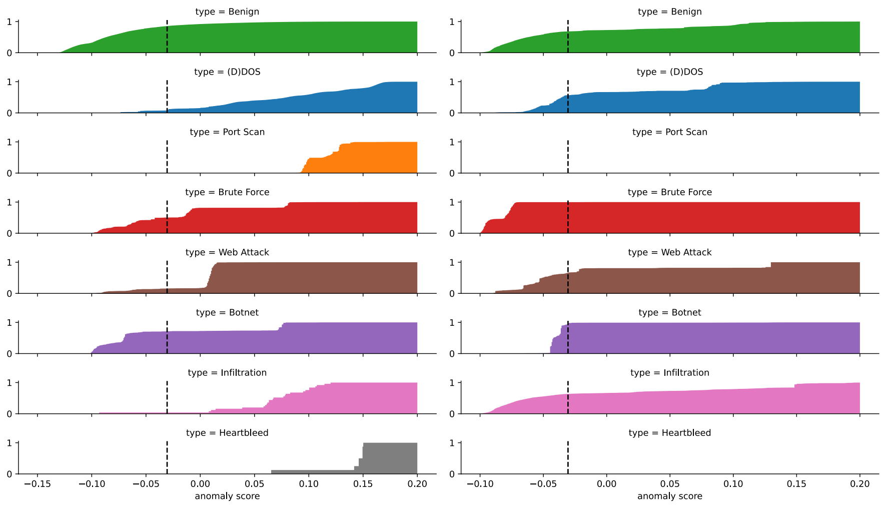
Fig. 5 Overview of the cumulative anomaly score distribution for the different attack classes for the autoencoder in intra-dataset evaluation

in the 2018 dataset. Secondly, the results demonstrate that the fairly easy detectable infiltration attack becomes more challenging in the 2018 dataset, and because of its larger share, it also weighs more through on the end result. Together they account for the decrease in AUROC. Contrarily to the 2017 results, the one-class SVM outperforms the others over the whole line, both in terms of fpr as tpr, see Fig. 4 in the right-bottom corner.

Table 3 gives an overview of the computational complexity as measured in the execution time of training and inference of the model. As expected, the time needed to train the model on the 2018 dataset is a bit longer due to the larger validation set. The time the models need to classify the test set is more diverse. PCA and autoencoder even increased their throughput because the inference time did not triple consistently with the size test set. While the isolation forest inference time changed accordingly to the test set size, the execution time of the one-class SVM exploded most likely due to different hyperparameters and the double as many principal components kept in the preprocessing pipeline. For use in operational applications, the throughput during inference is most important to serve as a real-time IDS. The used hardware allowed to classify up to 100.000s of flows per second, proving suitable for high-speed networks.

5.2 Inter-Dataset Evaluation

This study aimed to evaluate the generalization strength of promising algorithms for anomaly-based NIDS, more specifically unsupervised techniques. Therefore a novel inter-dataset evaluation strategy is used to train a model on the first dataset and evaluate it on the second dataset. Ideally, the classification performances should be similar, however, on average the AUROC decreased by 30.45%. In the best case, only a decrease of 17.85% was observed for the one-class SVM model trained on the 2018 dataset but a decrease of 37.49% in the worst case for the autoencoder also trained on the 2018 dataset. Similarly, the accuracy consistently dropped on average 25.63% between the intra- and inter-dataset evaluation strategy. While this is a vast drop in classification performance, all the final models still perform significantly better than a completely random classifier and are thus able to generalize to a certain degree. Figure 4 plots the roc curves of the four evaluated algorithms trained on the 2017 dataset and evaluated on the 2018 dataset in the bottom-left corner and for the models trained on the 2018 dataset and evaluated on the 2017 dataset in the top-right corner. All curves follow a similar trend except for the best generalizing one-class SVM model that outperforms the others. Analogous to the decline in classification performance in the intra-dataset setup, the decrease in the inter-dataset setup can be explained. Similar to Figs. 5, 6 plots the cumulative distribution of the anomaly score for different attack classes and benign traffic but using the data from the isolation forest trained on the 2017 dataset and evaluated on the 2017 and 2018 dataset. This figure clearly visualizes the disappearance of the previously easily detectable port scan attack and the more challenging detection of infiltration attacks. The heartbleed attack also disappeared, but because there are only a few samples present in the 2017 dataset, this barely influences the results. A combination of a shift in the



(a) CIC-IDS-2017 (b) CSE-CIC-IDS-2018

Fig. 6 The cumulative anomaly score distribution for the different attack classes for the same isolation forest model trained on CIC-IDS-2017 and evaluated in intra- and inter-dataset setup

distribution of attack classes between the 2017 and 2018 test set together with the model hyper-parameters selected on a validation set not representable for the distribution of attack classes in both datasets, explains the decrease in AUROC. The last observation that can be made from the plot is that the learned threshold for classification is not directly transferable to the second dataset. If the dashed line would be shifted to the left, thus taking a lower threshold, more malicious flows would be correctly flagged and result in a better F1 score than the current 0.4845.

In the inter-dataset setting, the computational complexity is not separately discussed again because the same model as trained in the intra-dataset setup without any adaptation is used, causing the same results.

6 Future Work

Current state-of-the-art algorithms for unsupervised anomaly-based NIDS are losing in the best case over 25% of their classification performance when fed with unseen but related data without any prior measures taken. This demands further research with the goal of improving the generalization performance of the proposed models. In the last decade, research has been focusing too much on achieving marginal gains on synthetic datasets and reporting them as advances while they have little effect for real-world applications. The inter-dataset evaluation strategy proposed in this paper is a strong candidate for adoption in future developments and, if necessary, further adapted to estimate the generalization strength of a model. By providing the research community with new tools to evaluate the generalization strength of proposed models, this work does not only try to close the gap in classification performance between academic prototypes and real-world applications but also raises awareness among researchers active in the field for this open challenge. Two main approaches exist to tackle this issue. First, machine learning models can only perform as well as the data they are trained on. The continuous development of new, high-quality academic datasets consisting of realistic attacks is required. Second, the used techniques can be further improved to be less prone to overfitting and actually learning a high-level representation of complex intrusions. For example, a more sophisticated feature engineering approach could be used to reduce the risk of overfitting. Another possibility is to validate that fine-tuning the hyperparameters of the pre-trained model with a very small number of flows out of the unseen dataset, improves the generalization strength. Furthermore, it would be favorable to have a similar study focusing on the generalization strength of supervised algorithms and comparing them with the results presented in this paper. We expect that unsupervised algorithms are overall more resilient to overfitting and thus achieve a higher generalization power. This needs to be validated by future work.

7 Conclusion

This study started with evaluating four unsupervised algorithms on two realistic and recent datasets. Then these results served as a baseline for estimating the generalization strength of the models with a novel proposed inter-dataset evaluation strategy. The unsupervised algorithms were able to achieve high classification scores on the individual datasets with high throughput rates up to 100.000s of flows per second. On average, the one-class SVM yielded the highest AUROC scores of 0.9705 on CIC-IDS-2017 and 0.9420 on CSE-CIC-IDS-2018, closely followed by the autoencoder. Even the most lightweight algorithm tested, PCA achieved a good classification performance with the lowest recorded AUROC of 0.8494 on CSE-CIC-IDS-2018. Moreover, unsupervised NIDS proved their capability of detecting unknown zero-day attacks as long as the malicious traffic differs from normal traffic.

The second part of this study validated if these promising results also withstand when classifying a second related dataset. This increasingly difficult inter-dataset evaluation setup decreased on average the AUROC and accuracy scores respectively by 30.45% and 25,63%, indicating that models trained on the current state-of-the-art intrusion datasets have a low generalization strength without any measures or adaptations in place. While there is a significant drop in classification performance, all four algorithms still perform better than a completely random classifier and thus are able to generalize the learned patterns to a certain degree. Again, the one-class SVM proved the best with *only* an average 25.89% decrease in AUROC. Above all, the acknowledgment of this generalization challenge can shift the current research focus of obtaining marginal gains on individual datasets in the direction of techniques improving the generalization strength. The proposed inter-dataset evaluation strategy in this study is a strong candidate for adaption in future research to estimate the generalization strength of newly developed models.

Appendix A: Overview features CIC-IDS-2017 and CSE-CIC-IDS-2018

Number	Name	Kept	Modified	Dropped	Note
1	Flow ID			∘	Remove dataset specific info
2	Source IP			∘	Remove dataset specific info
3	Source port			∘	Remove dataset specific info
4	Destination IP			∘	Remove dataset specific info
5	Destination port			∘	Remove dataset specific info
6	Protocol	∘			
7	Timestamp			∘	Remove dataset specific info
8	Flow duration	∘			
9–10	Total Fwd/Bwd packets	∘			
11–12	Total length of Fwd/Bwd packets	∘			
13–16	Fwd packet length max/min/mean/SD	∘			
17–20	Bwd packet length max/min/mean/SD	∘			
21	Flow bytes/s		∘		Remove NaN / Infinity
22	Flow packets/s		∘		Remove NaN / Infinity
23–26	Flow IAT mean/SD/max/min	∘			
27–31	Fwd IAT total/mean/SD/max/min	∘			
32–36	Bwd IAT total/mean/SD/max/min	∘			
37	Fwd PSH flags	∘			
38	Bwd PSH flags			∘	No variance
39	Fwd URG flags			∘	No variance
40	Bwd URG flags			∘	No variance
41–42	Fwd/Bwd header length	∘			
43–44	Fwd/Bwd packets/s	∘			
45–49	Packet length min/max/mean/SD/variance	∘			
50	FIN flag count	∘			
51	SYN flag count	∘			
52	RST flag count	∘			
53	PSH flag count	∘			
54	ACK flag count	∘			
55	URG flag count	∘			
56	CWE flag count			∘	No variance
57	ECE flag count	∘			
58	Down/up ratio	∘			
59	Average packet size	∘			
60–61	Avg Fwd/Bwd segment size	∘			
62	Fwd header length.1			∘	Duplicate column
63	Fwd avg bytes/bulk			∘	No variance
64	Fwd avg packets/bulk			∘	No variance

Number	Name	Kept	Modified	Dropped	Note
65	Fwd avg bulk rate			∘	No variance
66	Bwd avg bytes/bulk			∘	No variance
67	Bwd avg packets/bulk			∘	No variance
68	Bwd avg bulk rate			∘	No variance
69–70	Subflow Fwd/Bwd packets	∘			
71–72	Subflow Fwd/Bwd bytes	∘			
73	Init_Win_bytes_forward	∘			
74	Init_Win_bytes_backward	∘			
75	act_data_pkt_fwd	∘			
76	min_seg_size_forward	∘			
77–80	Active mean/SD/max/min	∘			
81–84	Idle mean/SD/max/min	∘			
85	Label			∘	Evaluation only

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Goasduff, L.: Gartner survey reveals 47% of organizations will increase investments in iot despite the impact of covid-19. https://www.gartner.com/en/newsroom/press-releases/2020-10-29-gartner-survey-reveals-47-percent-of-organizations-will-increase-investments-in-iot-despite-the-impact-of-covid-19- (2020)
2. Cheng, T., Lin, Y., Lai, Y., Lin, P.: Evasion techniques: sneaking through your intrusion detection/prevention systems. IEEE Commun. Surv. Tutor. **14**(4), 1011–1020 (2012). https://doi.org/10.1109/SURV.2011.092311.00082
3. Naylor, D., Finamore, A., Leontiadis, I., Grunenberger, Y., Mellia, M., Munafò, M., Papagiannaki, K., Steenkiste, P.: The cost of the "s" in https. In: Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies, CoNEXT '14, p. 133–140. Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2674005.2674991
4. Zeng, Y., Gu, H., Wei, W., Guo, Y.: $deep-full-range$: A deep learning based network encrypted traffic classification and intrusion detection framework. IEEE Access **7**, 45182–45190 (2019). https://doi.org/10.1109/ACCESS.2019.2908225
5. Canard, S., Diop, A., Kheir, N., Paindavoine, M., Sabt, M.: Blindids: Market-compliant and privacy-friendly intrusion detection system over encrypted traffic. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, ASIA CCS '17, pp. 561–574. Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3052973.3053013
6. Rezaei, S., Liu, X.: Deep learning for encrypted traffic classification: an overview. IEEE Commun. Mag. **57**(5), 76–81 (2019). https://doi.org/10.1109/MCOM.2019.1800819
7. Hubballi, N., Suryanarayanan, V.: False alarm minimization techniques in signature-based intrusion detection systems: a survey. Comput. Commun. **49**, 1–17 (2014) https://doi.org/10.1016/j.comcom.2014.04.012
8. Umer, M.F., Sher, M., Bi, Y.: Flow-based intrusion detection: techniques and challenges. Comput. Secur. **70**, 238–254 (2017). https://doi.org/10.1016/j.cose.2017.05.009

9. Nisioti, A., Mylonas, A., Yoo, P.D., Katos, V.: From intrusion detection to attacker attribution: a comprehensive survey of unsupervised methods. IEEE Commun. Surv. Tutor. **20**(4), 3369–3388 (2018). https://doi.org/10.1109/COMST.2018.2854724
10. Sultana, N., Chilamkurti, N., Peng, W., Alhadad, R.: Survey on sdn based network intrusion detection system using machine learning approaches. Peer-to-Peer Netw. Appl. **12**(2), 493–501 (2019). https://doi.org/10.1007/s12083-017-0630-0
11. Anderson, J.P.: Computer security threat monitoring and surveillance. James P. Anderson Company, Technical Report (1980)
12. Othman, S., Alsohybe, N., Ba-Alwi, F., Zahary, A.: Survey on intrusion detection system types **7**, 444–462 (2018)
13. Liu, H., Lang, B.: Machine learning and deep learning methods for intrusion detection systems: a survey. Appl. Sci. **9**, 4396 (2019). https://doi.org/10.3390/app9204396
14. Modi, C., Patel, D., Borisaniya, B., Patel, H., Patel, A., Rajarajan, M.: A survey of intrusion detection techniques in cloud. J. Netw. Comput. Appl. **36**(1), 42–57 (2013). https://doi.org/10.1016/j.jnca.2012.05.003
15. Otoum, S., Kantarci, B., Mouftah, H.: A Comparative Study of AI-based Intrusion Detection Techniques in Critical Infrastructures. arXiv:2008.00088 [cs] (2020)
16. Mirsky, Y., Doitshman, T., Elovici, Y., Shabtai, A.: Kitsune: An ensemble of autoencoders for online network intrusion detection. CoRR arXiv:abs/1802.09089 (2018)
17. Zavrak, S., İskefiyeli, M.: Anomaly-based intrusion detection from network flow features using variational autoencoder. IEEE Access **8**, 108346–108358 (2020). https://doi.org/10.1109/ACCESS.2020.3001350
18. Vartouni, A.M., Kashi, S.S., Teshnehlab, M.: An anomaly detection method to detect web attacks using stacked auto-encoder. In: 2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS), pp. 131–134 (2018). https://doi.org/10.1109/CFIS.2018.8336654
19. Nguyen, Q.T., Phuc Tran, K., Castagliola, P., Thu Huong, T., Nguyen, M.K., Lardjane, S.: Nested one-class support vector machines for network intrusion detection. In: 2018 IEEE Seventh International Conference on Communications and Electronics (ICCE), pp. 7–12 (2018). https://doi.org/10.1109/CCE.2018.8465718
20. Takeishi, N.: Shapley values of reconstruction errors of pca for explaining anomaly detection. In: 2019 International Conference on Data Mining Workshops (ICDMW), pp. 793–798 (2019). https://doi.org/10.1109/ICDMW.2019.00117
21. Otoum, Y., Nayak, A.: AS-IDS: Anomaly and signature based IDS for the internet of things. J. Netw. Syst. Manag. **29**(3), 23 (2021). https://doi.org/10.1007/s10922-021-09589-6
22. Dromard, J., Roudière, G., Owezarski, P.: Online and scalable unsupervised network anomaly detection method. IEEE Trans. Netw. Serv. Manag. **14**(1), 34–47 (2017). https://doi.org/10.1109/TNSM.2016.2627340
23. Safari Khatouni, A., Seddigh, N., Nandy, B., Zincir-Heywood, N.: Machine learning based classification accuracy of encrypted service channels: analysis of various factors. J. Netw. Syst. Manag. **29**(1), 8 (2020). https://doi.org/10.1007/s10922-020-09566-5
24. Sommer, R., Paxson, V.: Outside the closed world: On using machine learning for network intrusion detection. In: 2010 IEEE Symposium on Security and Privacy, pp. 305–316 (2010). https://doi.org/10.1109/SP.2010.25
25. Leevy, J.L., Khoshgoftaar, T.M.: A survey and analysis of intrusion detection models based on cse-cic-ids2018 big data. J. Big Data **7**(1), 104 (2020). https://doi.org/10.1186/s40537-020-00382-x
26. Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., Ahmad, F.: Network intrusion detection system: A systematic study of machine learning and deep learning approaches. Trans. Emerg. Telecommun. Technol. **n/a**(n/a), e4150. https://doi.org/10.1002/ett.4150
27. Al-Omari, M., Rawashdeh, M., Qutaishat, F., Alshira'H, M., Ababneh, N.: An intelligent tree-based intrusion detection model for cyber security. J. Netw. Syst. Manag. **29**(2), 20 (2021). https://doi.org/10.1007/s10922-021-09591-y
28. Aloqaily, M., Otou, S., Ridhawi, I.A., Jararweh, Y.: : An intrusion detection system for connected vehicles in smart cities. Recent advances on security and privacy. Intell. Transport. Syst. **90**, 101842 (2019). https://doi.org/10.1016/j.adhoc.2019.02.001
29. Narayanan, A., Shmatikov, V.: How to break anonymity of the netflix prize dataset. ArXiv **abs/cs/0610105** (2006)
30. of California, T.U.: Kdd cup 1999 data (28) (1999). http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

31. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the kdd cup 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1–6 (2009)
32. Protic, D.: Review of kdd cup '99, nsl-kdd and kyoto 2006+ datasets. Vojnotehnicki glasnik **66**, 580–596 (2018). https://doi.org/10.5937/vojtehg66-16670
33. Sharafaldin, I., Gharib, A., Habibi Lashkari, A., Ghorbani, A.: Towards a reliable intrusion detection benchmark datase. Software Netw. **2017**, 177–200 (2017). https://doi.org/10.13052/jsn2445-9739.2017.009
34. Sharafaldin, I., Habibi Lashkari, A., Ghorbani, A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. pp. 108–116 (2018). https://doi.org/10.5220/0006639801080116
35. of California, T.U.: Cicflowmeter https://github.com/ahlashkari/CICFlowMeter
36. Habibi Lashkari, A., Draper Gil, G., Mamun, M., Ghorbani, A.: Characterization of encrypted and vpn traffic using time-related features (2016). https://doi.org/10.5220/0005740704070414
37. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
38. Pearson, K.: Liii. on lines and planes of closest fit to systems of points in space. Philos. Mag. J. Sci. **2**(11), 559–572 (1901)
39. Shlens, J.: A tutorial on principal component analysis (2014)
40. Liu, F.T., Ting, K., Zhou, Z.H.: Isolation forest. pp. 413–422 (2009). https://doi.org/10.1109/ICDM.2008.17
41. Rumelhart, D.E., McClelland, J.L.: Learning Internal Representations by Error Propagation, pp. 318–362 (1987)
42. Plaut, E.: From principal subspaces to principal components with linear autoencoders (2018)
43. Chollet, F., et al.: Keras. https://keras.io (2015)
44. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015). https://www.tensorflow.org/. Software available from tensorflow.org
45. Schölkopf, B., Williamson, R., Smola, A., Shawe-Taylor, J., Platt, J.: Support vector method for novelty detection. pp. 582–588 (1999)
46. Xu, Y., Goodacre, R.: On splitting training and validation set: A comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization performance of supervised learning. Journal of Analysis and Testing **2**,(2018). https://doi.org/10.1007/s41664-018-0068-2
47. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2019)
48. Verkerken, M.: Gitlab repository containing code for paper "towards model generalization for intrusion detection: Unsupervised machine learning techniques". https://gitlab.ilabt.imec.be/mverkerk/cic-ids-2018 (2021)
49. Bradley, A.P.: The use of the area under the roc curve in the evaluation of machine learning algorithms. Pattern Recognition **30**(7), 1145–1159 (1997) https://doi.org/10.1016/S0031-3203(96)00142-2