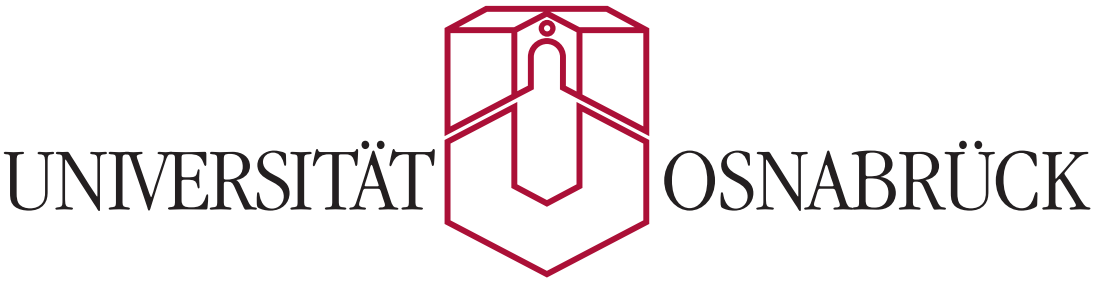


INSTITUTE OF COMPUTER SCIENCE
DISTRIBUTED SYSTEMS GROUP

Attacks against Deep Learning based Network Intrusion Detection Systems

Bachelor Thesis

Dominik Brockmann

Date of Submission: February 2, 2024

First Supervisor
Prof. Dr. rer. nat. N. Aschenbruck

Second Supervisor
E. Lanfer, M. Sc.

Abstract

Cyber-attacks are an increasingly widespread threat that pose serious risks to all types of information systems. Intrusion detection systems (IDS) can be used to detect such attacks. However, traditional IDS are based on signatures and are unable to detect unknown threats. Therefore, the use of deep learning models to detect intruders is becoming increasingly popular. While deep learning can help by providing better generalization capabilities, it lacks explainability and has a common vulnerability that allows attackers to make minimal changes to the input that alter the output. So-called adversarial attacks are particularly popular in the field of computer vision. Our research goal is to evaluate the vulnerability of deep learning-based IDS against such attacks and to investigate the impact of different datasets, model architectures and attack methods on attack success. Therefore, we train three common deep learning architectures with three publicly available datasets. We use state-of-the-art white-box, substitute and black-box attacks and investigate the resulting attack success considering domain-specific constraints. We find that deep learning based IDSs are highly vulnerable to all types of adversarial attacks. Furthermore, our results indicate that the vulnerability lies in the data itself and not in the model architectures.

Zusammenfassung

Cyberangriffe sind eine zunehmend weit verbreitete Bedrohung, die ernsthafte Risiken für alle Arten von Informationssystemen darstellt. Intrusion Detection Systeme (IDS) können zur Erkennung solcher Angriffe eingesetzt werden. Herkömmliche IDS beruhen jedoch auf Signaturen und sind nicht in der Lage, unbekannte Bedrohungen zu erkennen. Daher wird der Einsatz von Deep Learning Modellen zur Erkennung von Angriffen zunehmend beliebter. Deep Learning kann helfen, indem es bessere Generalisierungsfähigkeiten bietet, aber es mangelt an Erklärbarkeit und weist eine allgemeine Schwachstelle auf, die es Angreifern ermöglicht, minimale Änderungen an der Eingabe vorzunehmen, die die Ausgabe verändern. Sogenannte Adversarial Attacks sind besonders aus dem Bereich der Computer Vision bekannt. Unser Forschungsziel ist es, die Vulnerabilität von Deep Learning basierten IDS gegen solche Angriffe zu bewerten und die Auswirkungen verschiedener Datensätze, Modellarchitekturen und Angriffsmethoden auf den Angriffserfolg zu untersuchen. Daher trainieren wir drei gängige Deep Learning Architekturen mit drei öffentlich verfügbaren Datensätzen. Wir verwenden modernste White-Box-, Substitute- und Black-Box-Angriffe und untersuchen den daraus resultierenden Angriffserfolg unter Berücksichtigung der domänenspezifischen Beschränkungen. Wir stellen fest, dass Deep Learning basierte IDSs sehr anfällig für alle Arten von Adversarial Attacks sind. Darüber hinaus deuten unsere Ergebnisse darauf hin, dass die Anfälligkeit in den Daten selbst und nicht in den Modellarchitekturen liegt.

Contents

1 Introduction

Cyber-attacks are an increasingly popular threat and pose risks to the confidentiality, integrity, and availability of information systems [16]. Therefore, Intrusion Detection Systems (IDS), which can discover and identify unauthorized access to systems, are a crucial component of every network infrastructure. However, attacks are becoming more complex and attackers more creative in creating unique attacks that are difficult to detect. Traditional IDS based on signatures are loosing their effectiveness and are unable to detect zero-day attacks [42]. Thus, anomaly-based IDS are the main focus of research, which detect intrusions by anomalies instead of pre-extracted signatures. Regarding such systems, the application of deep learning techniques is becoming more popular, since they are known to have better generalization abilities. Deep learning techniques also require less feature engineering and domain expert knowledge to be successful, which reduces the necessary effort to create a working system [34]. But deep learning models are black-box models, which means that their decision-making is unclear due to their complex structure [116]. The reliable success of an IDS is crucial to ensure the security of computers, servers, and data, and IDS that cannot be interpreted could have unforeseen flaws and pose a security risk. Szegedy et al. [97] demonstrate by using computer vision problems, that deep learning models are vulnerable to adversarial examples, which are samples with subtle perturbations that change the output of the model. Such perturbations can be added at the training stage of a model to change its structure or at model inference to fool a model that is performing well according to all evaluation metrics [17]. For IDS in particular, a targeted adversarial attack can disguise malicious traffic as benign traffic and perform an intrusion while remaining undetected. It is important to note that the success of such attacks is not dependent on poor model performance, i.e., models with high accuracy can be exploited as well [35]. From this vulnerability rises the question about the reliability of deep learning based IDS.

Figure 1 illustrates a possible real-world adversarial attack against an IDS. In this scenario, the IDS is treated as a black-box model and the attacker has no information about the internal structure. First, the attacker sends malicious traffic to the victim model. The IDS decides using the underlying deep learning model whether the traffic is blocked or accepted. Based on the response of the victim network, the attacker can then add perturbations to the traffic that change a subset of the traffic features. These perturbations are targeted changes to traffic features, e.g., increasing the size of packets. By repeating this process, the attacker can bypass the IDS and perform an undetected intrusion.

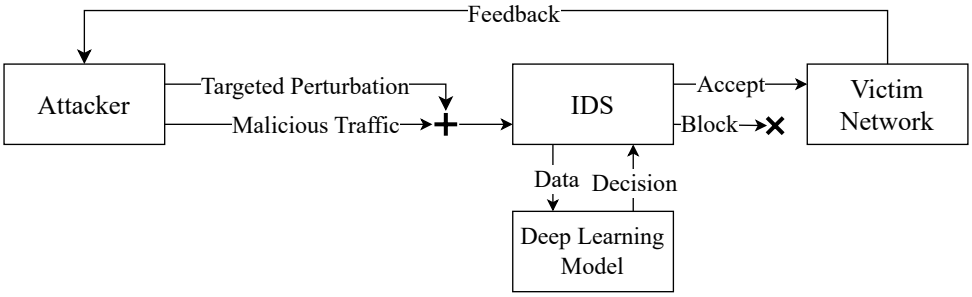
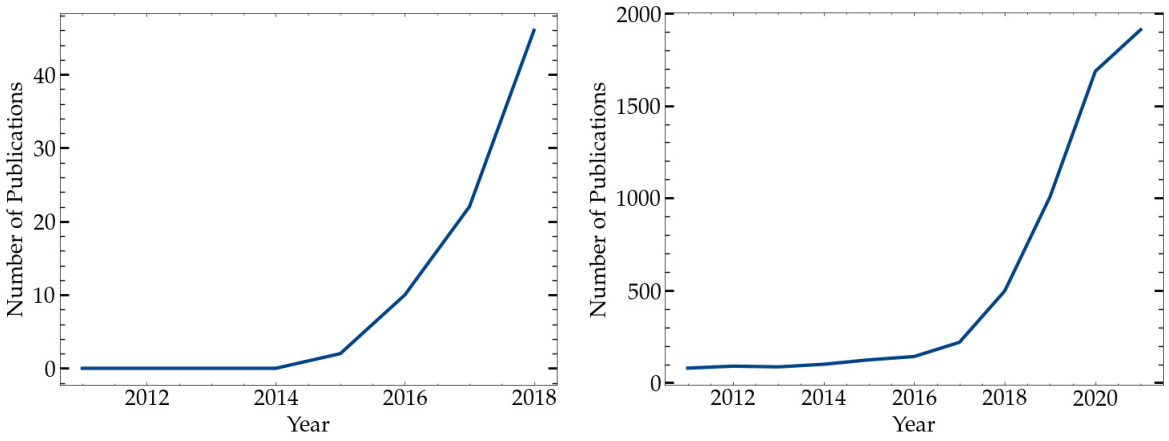


Figure 1: Illustration of the Attack Process of a Real-World Black-Box Adversarial Attack, inspired by [112]

In the last decade, an increasing interest in deep learning techniques for intrusion detection can be observed, shown by the increasing number of publications studying such techniques. Osken et al. [76] analyze the number of publications, which include the keywords *Deep Learning*

and *Intrusion Detection*. Figure 2a summarizes their findings and shows the number of publications for each year between 2011 and 2018 according to their analysis. In the same time frame, the awareness of adversarial attacks has increased. Long et al. [64] analyze articles in the field of adversarial attacks against deep learning models. The authors demonstrate that since the first discovery of adversarial examples by Szegedy et al. [97] in 2013, the number of publications has steadily increased until 2017, when a rapid increase in publications can be observed. Their findings are shown in Figure 2b. This shows the relevance of the topic and the motivation behind research regarding adversarial attacks against deep learning based IDS.



(a) Number of Publications per Year in the Field of Deep Learning based IDS according to [76]

(b) Number of Publications per Year in the Field of Adversarial Attacks according to [64]

Figure 2: Number of Publications showing an Increase in Interest of Deep Learning based IDS and Adversarial Attacks

Due to the previous motivation, we aim to evaluate the vulnerability of deep learning based IDS to adversarial attacks. We focus on deep learning techniques and do not consider other machine learning-based approaches, since we want to investigate adversarial attacks, which are a phenomenon observed specifically for deep learning techniques. Our research goal is to analyze the influence of datasets, model architectures and different attack methods on the success of such attacks. We investigate deep learning classifiers in two adversary settings: white-box, where an adversary has access to the model and black-box, where only model outputs are available. In the first part, we train three commonly applied model architectures (i.e., DNN, CNN, and Autoencoders) on three publicly available IDS datasets (i.e., UNSW-NB15, CIC-IDS2017, and CSE-CIC-IDS2018) and evaluate their performances for classification. In the second part, we apply various state-of-the-art white-box (i.e., BIM, PGD, Deepfool, and C&W) and black-box (i.e., ZOO, Boundary, and HopSkipJump) attacks on them. Additionally, we test the transferability of white-box adversarial examples using substitute models. We evaluate the success in compliance with domain constraints using subsets of modifiable features and test the vulnerability when modifying only a single feature. In our work, we focus on multiclass classifiers because of their higher interpretability, which is necessary for IDS applications, but we also consider binary classifiers and evaluate their vulnerability to adversarial attacks.

The thesis is structured as follows: First, we explain background information regarding our topic in Chapter 2 and summarize work that is related to our research in Chapter 3. Then, in Chapter 4, we present our approach to the research goal in detail. In Chapter 5, we show the results following our approach and discuss them in Chapter 6. Finally, we conclude our research in Chapter 7 and give suggestions for future work.

2 Background

2.1 Intrusion Detection Systems

Intrusion Detection Systems (IDS) are systems with the goal of identifying possible intrusions in network traffic. An intrusion can be described as an unauthorized activity that poses a threat to the confidentiality, integrity, or availability of an information system [52]. Different platforms can have IDS set up to detect attacks: on a single host (host-based), on the network (network-based) or on both using a hybrid approach. Host-based IDS can monitor system-level activities of a host, including running applications and system logs. In contrast, network-based IDS analyze network traffic and are the research subject of this study [77].

2.1.1 Types of Intrusion Detection Systems

IDS can be divided into two groups describing the detection method: signature-based and anomaly-based IDS [52]. Signature-based IDS identify intrusions by pattern matching. The system requires a database with signatures that are extracted from previous intrusions. Such signatures can consist of simple conditions (e.g., source IP equals destination IP address) or more complicated, chained conditions that can be represented by state machines or formal language [50]. The main disadvantage of signature-based detection methods is their inability to detect unknown threats in the form of zero-day attacks. Holm [42] analyzes the ability of a signature-based IDS to detect zero-day attacks, and estimates a detection rate of 8.2%. This indicates that those systems are not completely unable to detect unknown intrusions but lack the ability of reliable detection.

Anomaly-based IDS, however, do not rely on previous extracted signatures. Their detection method is based on the creation of normal behavior profiles. If an activity shows significant deviations to the normal profile, it is interpreted as an intrusion [52]. Anomaly-based IDS can be further categorized into groups, describing the profile extraction method. The main methods are statistical-based, knowledge-based or based on machine learning. A statistical-based approach captures network traffic activity and derives statistical metrics from them (e.g., the traffic rate), which can be compared to current activity. Knowledge-based methods, on the other hand, define rules specifying normal user activity based on human expert knowledge, a so-called expert system. The third group based on machine learning describes training a model to analyze and categorize patterns in traffic. The main difference to the statistical-based approach is the ability to improve the model while acquiring new information [32]. The advantage of anomaly-based IDS in comparison to signature-based IDS is the ability to detect zero-day attacks. But since unusual user behavior is difficult to distinguish from an anomaly caused by an intrusion, such systems often have a high false positive rate [52]. Additionally, anomaly-based IDS often lack explainability, especially if they are based on deep learning [116].

2.1.2 Evasion of Intrusion Detection Systems

An evasion describes different ways to escape the detection of an IDS when performing an intrusion. Examples of techniques to evade an IDS include packet splitting, duplicate insertion, payload mutation and shellcode mutation [21]. We shortly present the named evasion techniques.

Packet splitting is a method of splitting packets into IP fragments or TCP segments. The underlying intrusion is not changed, since the victim host will reassemble the packets. Ideally, the IDS keeps track of all packets, but especially regarding a high number of concurrent connections the available resources may not be sufficient. Therefore, packet splitting can evade detec-

tion since the signature or other features of the traffic used by the detection method are altered. Another evasion technique is the insertion of duplicate or overlapping fragments/segments. A possible attack regarding this technique is the use of small Time-To-Live (TTL) values that are big enough to reach the IDS but are dropped before reaching the victim system. Additionally, overlapping fragments/segments can be used to create an ambiguity for the IDS, since the interpretation can depend on the operating system of the victim. Payload mutation describes the transformation of the packet payload of an intrusion. The goal is to apply the transformation in a way that does not change the semantics of the payload, meaning the intrusion is still effective. The last evasion technique that we present here is the mutation of shellcode. Shellcode is a piece of code used in the exploitation of a software vulnerability. To evade the detection by an IDS, the shellcode can be encrypted, compressed or transformed into a semantically equal form. The original code is then decrypted or decompressed on the victim system. The detection of this technique is especially difficult and may require the emulation of code execution [21].

All the presented evasion techniques target more traditional IDS, especially IDS based on signature matching. However, research in the last few years shows increasing interest in using deep learning methods for intrusion detection [31]. Such methods allow for a new evasion technique based on adversarial attacks, which will be further studied in this research.

2.2 Deep Learning Architectures

In the following sections, we describe three commonly used deep learning architectures, including deep neural networks, convolutional neural networks and autoencoders. The description *deep* is often used for any artificial intelligence approach, but it is defined as the combination of multiple concepts, or more specific layers, stacked on top of each other. Building a system out of many simple concepts can allow a computer to learn more complicated concepts [34].

2.2.1 Deep Neural Networks

Deep Neural Networks (DNNs), also called Deep Feedforward Networks (DFNs) or Multi-Layer Perceptrons (MLPs), present the classic deep learning architecture. A DNN learns to approximate a function. In the case of a classification task, the model learns the mapping function of an input to a category. The model is a network consisting of multiple layers. The first layer is called the input layer and the final layer, the output layer. All intermediate layers are called hidden layers. The number of all layers determines the depth of the model and differentiates deep learning architectures from other machine learning models. This type of network is called feedforward, which describes the information flow inside the network. In feedforward networks, the information is propagated through the layers in the forward direction and there exist no feedback connections. For each computed value from the previous layer, an activation function is applied to the value. This function is nonlinear and allows the linear structured network to learn nonlinear mappings, making the model as a whole nonlinear [34].

DNNs can be trained using supervised learning, where for each data sample the corresponding true output is available. The commonly used training algorithm for DNNs is back-propagation. This algorithm uses gradient descent to minimize the loss function, which in most cases is the difference between the input and true output. Because of the nonlinearity of DNNs, gradient descent is not guaranteed to find the global but only a local minimum [34]. The currently most often used optimization method is the Adam optimizer [55], which is based on stochastic gradient descent (SGD). SGD applies gradient descent iteratively using a fixed learning rate. However, a fixed learning rate introduces noise, which does not vanish when being

near a minimum. Adam extends this algorithm with an adaptive learning rate to solve this problem.

2.2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a deep learning architecture for data with spatial properties. In image data, for example, the position of a pixel in relation to other pixels is an important property. The typical data of CNNs are images or sequence data, but other data types can be applied to make use of their pattern recognition capabilities. CNNs can learn such representations using discrete convolution, which is a mathematical operation based on an input and a kernel, which can be learned [34].

CNNs typically consist of multiple layers, each performing three operations, with the first being the application of convolutions. Similar to DNNs, a nonlinear activation function is used on the outputs of this operation. The third operation is a pooling function, which summarizes values within a rectangular neighborhood and enforces the model to be invariant to translation, which can improve efficiency. A common pooling function is max pooling, which reduces a neighborhood to the maximum value in it [36]. In addition to such layers, the network can be extended using typical feedforward layers. CNNs can be trained using backpropagation, similar to DNNs [34].

2.2.3 Autoencoders

An autoencoder (AE) is a feedforward network with special properties that is trained to replicate the input in the output, essentially learning the identity function. But the topology of AEs is designed to make it impossible to perfectly learn this function for all possible inputs. The network consists of two parts, an encoder and a decoder. The output of the encoder (code layer) is a lower dimensional representation of the input, which the decoder uses to reconstruct the original input. Since the dimension of the code layer is smaller than the input dimension, the model learns an undercomplete representation which depends on the training data [34]. AEs can be trained in the same way as DNNs and CNNs. Since the data is used for both input and output and no additional target output is necessary, AEs can be considered unsupervised.

AEs are utilized for various tasks, including dimension reduction, feature extraction, noise removal, anomaly detection and generative modeling. Regarding feature extraction, AEs learn a more powerful nonlinear generalization of PCA. But the efficiency of extracting useful features depends on the size of the code layer. If the AE has too much capacity, it can learn to copy data without extracting any useful representations. Therefore, a sparse AE extends the original model by adding an activation penalty to the code layer [111]. This enforcement of sparse representations can lead to the extraction of useful features even in the overcomplete case, where the code layer is larger than the input [34]. For a classification task, the learned representations of an AE can be used to train a supervised DNN.

2.3 Attacks against Deep Learning Models

Szegedy et al. [97] report unexpected outputs from deep neural networks which an adversary can actively cause through imperceptible perturbations in the data. Deep learning based models seem generally vulnerable to such attacks, but as Goodfellow et al. [35] report, the vulnerability is not limited to such models and is a general problem which affects linear classifiers as well. Such attacks can be divided into adversarial attacks that are applied on an already trained model and backdoor attacks which modify parts of the training data before training.

2.3.1 Adversarial Attacks

An adversarial attack describes the perturbation of an original sample with the goal of misclassification by a classification system. Such a perturbed sample is called an adversarial example [35]. There exist various methods to create adversarial examples, which can be grouped by their adversary goal and their knowledge and access to the victim classifier model.

Untargeted and Targeted Attacks An untargeted adversarial attack has the goal of changing the class prediction of a correctly classified sample to any other class. In the case of many classes, this goal can lead to the prediction of a similar class (e.g., mistaking a dog by another dog race instead of an entirely different animal), which might be less interesting from an adversary perspective. In such a scenario, a targeted attack can lead to misclassifications which are more dissimilar [57]. Regarding targeted attacks, the adversarial attack is only successful if the prediction is changed to a specified class. The target can be the same for every class, or depend on the class of the input. Another adversary goal can be confidence reduction. For this goal, the predicted class label can stay the same, but the confidence of the prediction is minimized [17].

White- and Black-Box Attacks The white- or black-box property of an attack specifies the information an adversary has about the victim model. White-box attacks require internal knowledge about the model, especially the trained weights. This knowledge can be exploited to compute adversarial examples. But the weights of a deployed model are typically not accessible. Therefore, black-box attacks specify no or only partial information about the model [17].

Black-box attacks have no access to the model weights, but can be further divided by the necessary information of an output. Score-based attacks require the class probabilities of the classification model given a certain input. Decision-based attacks, on the other hand, rely their crafting algorithm only on the final class prediction [12]. Another commonly used attack is the substitute attack, which describes training a substitute model and applying white-box attacks on it. The resulting adversarial examples are then transferred to the victim model. While this attack is not based on information about the weights, it requires partial training data. It is considered another type of black-box attack, but it should be noted that acquiring training data can be difficult [17]. Table 1 summarizes the access to the victim model of each type of attack.

Table 1: Access of Different Attack Types to the Victim Model

Method	Weights	Training Samples	Class Predictions	Class Output
white-box	X		X	X
substitute		X		
black-box score-based			X	X
black-box decision-based				X

2.3.2 Backdoor Attacks

A backdoor attack, also known as a poisoning attack, involves the attacker manipulating or poisoning the training data before the actual training [17]. Therefore, an adversary can introduce a backdoor to the learning system that consists of a hidden trigger. Only when certain conditions are fulfilled is the output of the system changed. This results in a system that performs well on test data but has unforeseen flaws which are actively programmed into the system. Manipulation of training data can be difficult, but especially when data is extracted from the internet, the backdoor attack can pose a serious threat [61].

3 Related Work

Our research is related to work concerning the general use of deep learning techniques for network intrusion detection and the application of adversarial attacks against IDS. First, we present work concerning the first task, deep learning based IDS. The second section summarizes work regarding the application of adversarial attacks on such systems.

3.1 Deep Learning based Intrusion Detection Systems

Deep Learning is an increasingly popular research field, and more and more other disciplines are starting to utilize its power for their tasks. We present related work using different deep learning architectures for intrusion detection in the following paragraphs.

Artificial Neural Networks Regarding IDS, the first work in the direction of deep learning is the application of artificial neural networks (ANNs) to detect intrusions. Cannady [14] employs an ANN for binary classification of self collected data using nine extracted features. Li et al. [60] follow a similar approach and use ANNs to distinguish between benign and attack traffic. Other work utilizes ANNs on public datasets, including the DARPA'98 and KDDCUP'99 datasets [72, 68, 91]. Extending the binary classification approach, Moradi and Zulkernine [68] and Sammany et al. [91] split the attack class into two classes of different attacks. Furthermore, Naoum et al. [73] make use of ANNs for a multiclass classification task on the NSL-KDD dataset. ANNs are not considered deep learning, but since most of the presented work applies ANNs with two or more hidden layers, they can already be classified as deep learning.

Deep Neural Networks More sophisticated work uses DNNs which have more layers and often also more nodes per layer. Kim et al. [53] present a DNN based approach for binary classification of the KDDCUP'99 dataset. Regarding multiclass classification, Tang et al. [98] use the same type of architecture on the NSL-KDD dataset. To reduce the time needed for training, Potluri and Diedrich utilize a layer-wise unsupervised pre-training of a DNN. They can enhance the training efficiency in both multiclass and binary tasks [83].

Deep Belief Networks Another deep learning architecture is Deep Belief Networks (DBNs), which consist of Restricted Boltzmann Machines (RBMs) and can be trained unsupervised [43]. To obtain class predictions based on labels, a classifier can be attached to the DBN. Salama et al. [90] and Dong and Wang [24] employ DBNs with Support Vector Machines (SVMs) for multiclass classification on the NSL-KDD and KDDCUP'99 datasets, respectively. Alternatively, Alrawashdeh and Purdy [8] make use of a softmax layer for supervised classification.

Long Short-Term Memory Since network data is time-based, researchers study the ability of Recurrent Neural Networks (RNNs) to identify network intrusions. The particular architecture used by most of this work is the Long Short-Term Memory (LSTM) [41]. Saharkhizan et al. [89] apply LSTMs for binary traffic classification on IoT traffic data [30]. Lin et al. [62] employ an LSTM for multiclass classification of traffic from the CSE-CIC-IDS2018 dataset. Additionally, He et al. [40] introduce a new approach by combining LSTMs with multimodal deep AEs. They analyze their technique using the NSL-KDD, UNSW-NB15 and CIC-IDS2017 datasets.

Convolutional Neural Networks The combination of LSTMs with other deep learning techniques is a common approach. Zhang et al. [115] utilize CNNs in combination with LSTMs

for the multiclass and binary classification task on the CIC-IDS2017 dataset. Further studies use CNNs, e.g., Kim et al. [54] use a pure CNN architecture to classify DoS attacks in two datasets (NSL-KDD and CSE-CIC-IDS2018). Another approach by Xiao et al. [105] consists of two-dimensional CNNs for multiclass classification.

Autoencoders Traffic data often includes many features that can be reduced by feature extraction using AEs. Javaid et al. [47] extract features of the NSL-KDD dataset using a sparse AE. Niyaz et al. [75] utilize a similar approach on self collected data to detect DDos attacks in both the multiclass and binary setting. Extending the idea of standard AEs, Shone et al. [94] employ non-symmetric deep AEs for feature extraction. Another capability of AEs is the detection of anomalies while requiring only benign data and no attack traffic. Cordero et al. [22] follow this approach for binary anomaly detection using replicator neural networks on the MAWI dataset [29]. Replicator neural networks are similar to AEs but use a step-wise activation function on the encoder output layer to obtain representations with discrete values [38].

Table 2 shows a summary of the presented papers. The papers are grouped by the used deep learning architecture, with specifications on the dataset and binary/multiclass setting.

Table 2: Related Work using Deep Learning for IDS

Paper	Method	Dataset	Classes
Cannady (1998) [14]	ANN (2 hidden layers)	self collected	binary
Mukkamala et al. (2002) [72]	ANN (2 hidden layers)	KDDCUP'99	binary
Moradi and Zulkernine (2004) [68]	ANN (2 hidden layers)	DARPA'98	multiclass (2 attacks)
Li et al. (2004) [60]	ANN	self collected	binary
Sammany et al. (2007) [91]	ANN (2 hidden layers)	DARPA'98	multiclass (2 attacks)
Naoum et al. (2012) [73]	ANN	NSL-KDD	multiclass
Potluri and Diedrich (2016) [83]	DNN with pre-training	NSL-KDD	multiclass & binary
Tang et al. (2016) [98]	DNN	NSL-KDD	multiclass
Kim et al. (2017) [53]	DNN	KDDCUP'99	binary
Salama et al. (2011) [90]	DBN with SVM	NSL-KDD	multiclass
Dong and Wang (2016) [24]	DBN with SVM	KDDCUP'99	multiclass
Alrawashdeh and Purdy (2016) [8]	DBN with softmax	KDDCUP'99	multiclass
Lin et al. (2019) [62]	LSTM	CSE-CIC-IDS2018	multiclass
He et al. (2019) [40]	LSTM with MDAE	NSL-KDD, UNSW-NB15 & CIC-IDS2017	multiclass & binary
Saharkhizan et al. (2020) [89]	LSTM	IoT Traffic Data	binary
Zhang et al. (2019) [115]	CNN with LSTM	CIC-IDS2017	multiclass & binary
Xiao et al. (2019) [105]	CNN (2-dimensional)	KDDCUP'99	multiclass
Kim et al. (2020) [54]	CNN	NSL-KDD & CSE-CIC-IDS2018 (DoS attacks)	multiclass & binary
Javaid et al. (2016) [47]	sparse AE with softmax	NSL-KDD	multiclass & binary
Cordero et al. (2016) [22]	replicator NN	MAWI dataset	binary
Niyaz et al. (2016) [75]	AE	self collected	multiclass & binary
Shone et al. (2018) [94]	asymmetric AE with RF	NSL-KDD	multiclass

We apply our tests using DNN, CNN and AE architectures, since they are the most common and cover up three different types of classification techniques. In particular, the usage of DBNs is similar to AEs which is the reason we only use one of these architectures. We do not include LSTMs in our research because they are difficult to compare to the other approaches, which are not based on sequential data.

3.2 Adversarial Attacks against Intrusion Detection Systems

The awareness of adversarial attacks as an attack vector against deep learning based systems is increasing, shown by the number of publications regarding this topic in the last few years. Work regarding adversarial attacks against deep learning based IDS can be categorized in white-box and black-box attacks, specifying the necessary access of an adversary to the victim.

White-Box Attacks First, we present publications about white-box attacks. Most of these attacks consist of the application of the fast gradient sign method (FGSM) or its iterative extensions (BIM and PGD), which create fixed-size perturbed adversarial examples [35]. Warzyński and Kołaczek [104] and Debicha et al. [23] research the vulnerability of DNNs against the named methods. Other work uses the same classifier architecture and attack methods but for the BoT-IoT and UNSW-NB15 datasets [44, 4]. Peng et al. [81] applies momentum-based FGSM [25], L-BFGS, as proposed by Szegedy [97] and Simultaneous Perturbation Stochastic Approximation (SPSA) [101], which is a gradient-free optimization method. Rigaki [87] and Wang [103] test the vulnerability of IDS using the Jacobian-based Saliency Map Attack (JSMA) [78], which only perturbs a fraction of the features, determined by their importance. More sophisticated attack methods aim to produce adversarial examples with minimal perturbations, such as Deepfool or the Carlini and Wagner attack (C&W) [15]. Abou Khamis and Matrawy [3] investigate Deepfool and the C&W attack against DNN, CNN and LSTM based binary classifiers on the NSL-KDD and UNSW-NB15 datasets. They achieve success rates of around 60% using DeepFool and 26% using the C&W attack against DNNs. Wang [103] examines the success of these attacks for a multiclass DNN on the NSL-KDD dataset. Adversarial attacks against IDS need to follow certain domain constraints that are not needed for image data. Most of the above publications do mention that such constraints exist, but do not present applicable feature subsets or other techniques to ensure that domain constraints are fulfilled. However, Teuffenbach et al. [100] categorize features that can be manipulated in a real-world scenario and extend the C&W attack to be compatible with the restrictions. They test their methodology using DNNs, DBNs and AEs on the NSL-KDD and CIC-IDS2017 datasets and achieve success rates of about 55%. Hashemi et al. [37] propose an attack method which consists of three techniques (split, delay and inject) to manipulate traffic features in a realistic setting.

Black-Box Attacks Black-box attacks, in contrast to white-box attacks, do not require internal information about the victim model and are therefore more feasible for a real-world attack. Qiu et al. [84] and Yang et al. [108] use substitute attacks with FGSM and the C&W attack, respectively. Substitute attacks describe the training of a substitute model to mimic the victim model and application of white-box attacks in a black-box setting. Another type of attack is the generation of adversarial examples using Generative Adversarial Networks (GANs). Lin et al. [63] employ GANs to fool a multiclass DNN classifier, distinguishing two attack types and benign traffic. Other publications regarding adversarial attacks utilizing GANs focus on binary classifiers and use the KDDCUP'99 and NSL-KDD datasets [108, 102, 107]. To improve adversarial attacks against IDS, a few papers developed their own attack methodology. Kuppa et al. [56] describe a method to reduce the number of queries to the victim model using a Manifold Approximation Algorithm (MAA). Zhang et al. [114] develop a brute-force method to find successful adversarial examples. Peng et al. [80] present a boundary-based method, which optimizes the Mahalanobis distance. Another black-box attack type is the estimation of gradients to apply white-box attacks on them. Yang et al. [108] use Zeroth-Order Optimization (ZOO) to

estimate the gradients of a DNN classifier. Zhang et al. [113, 112] use NES to estimate gradients of DNNs, CNNs and C-LSTMs, which are a combination of CNNs and LSTMs. Additionally, they test four other black-box attacks including the Boundary, Pointwise, HopSkipJump and Opt-Attack. Their attacks can fool the victim model, with success rates between 20% and 30%. Black-box attacks are by definition more realistic, and most of the work studying such attacks also includes the separation of features that can be changed and features that cannot. These papers propose or contribute to defining plausible domain constraints [102, 107, 56, 113, 63].

Table 3 shows an overview of related work covering adversarial attacks against deep learning based IDS. The number of publications regarding this topic has increased in the last few years, but since the underlying techniques are relatively new, more research is necessary to understand the threat of adversarial attacks in the IDS domain. Especially work covering targeted black-box attacks on multiclass classifiers and general overviews to compare different attack methods, model architectures and datasets are rare.

Table 3: Related Work using Adversarial Attacks against Deep Learning based IDS

Paper	Classifier	Dataset	Classes	Methods	Attacks
Rigaki (2017) [87]	DNN	KDDCUP'99 & NSL-KDD	multiclass (un- & targeted)	white-box	FGSM & JSMA
Warzyński and Kołaczek (2018) [104]	DNN	NSL-KDD	binary	white-box	FGSM
Wang (2018) [103]	DNN	NSL-KDD	multiclass (un- & targeted)	white-box	FGSM, JSMA, Deepfool & C&W
Ibitoye et al. (2019) [44]	DNN & SNN	BoT-IoT	multiclass (untargeted)	white-box	FGSM, BIM & PGD
Peng et al. (2019) [81]	DNN	NSL-KDD	multiclass (untargeted)	white-box	PGD, MI-FGSM, L-BFGS & SPSA
Hashemi et al. (2019) [37]	DNN	CIC-IDS2017	multiclass (untargeted)	white-box	own method (Split-Delay-Inject)
Piplai et al. (2020) [82]	GAN	BigData 2019	binary	white-box	FGSM
Abou Khamis et al. (2020) [4]	DNN	UNSW-NB15	binary	white-box	FGSM & PGD
Abou Khamis and Matrawy (2020) [3]	DNN, CNN & LSTM	NSL-KDD & UNSW-NB15	binary	white-box	FGSM, BIM, PGD, Deepfool & C&W
Teuffenbach et al. (2020) [100]	DNN, DBN & AE	NSL-KDD & CIC-IDS2017	multiclass (targeted) & binary	white-box	C&W extension
Debicha et al. (2021) [23]	DNN	NSL-KDD	binary	white-box	FGSM, BIM & PGD
Yang et al. (2018) [108]	DNN	NSL-KDD	binary	substitute & black-box	C&W, ZOO & GAN
Usama et al. (2019) [102]	DNN	KDDCUP'99	binary	black-box	GAN
Yan et al. (2019) [107]	CNN	KDDCUP'99	binary (DoS)	black-box	GAN
Peng et al. (2019) [80]	DNN	KDDCUP'99 & CIC-IDS2017	multiclass (untargeted)	black-box	own method (Boundary-based)
Kuppa et al. (2019) [56]	AE, GAN & others	CSE-CICIDS2018	binary	black-box	own method (using MAA)
Qiu et al. (2020) [84]	AE	pre-trained IoT model	binary	substitute	FGSM
Zhang et al. (2020) [114]	DNN	NSL-KDD	multiclass (un- & targeted)	black-box	own method (brute-force)
Zhang et al. (2022) [113, 112]	DNN, CNN & C-LSTM	CSE-CICIDS2018	multiclass (targeted) & binary	black-box	NES, Boundary, Pointwise, HopSkipJump & Opt-Attack
Lin et al. (2022) [63]	DNN	NSL-KDD	multiclass (2 attacks)	black-box	GAN

4 Methods

In the following sections, we will present our approach to the research goal. The methodology can be divided into two parts: the first part consists of the data processing and creation of classifier models, the second part includes the application of attacks on the models. Figure 3 summarizes this pipeline and shows the different processing steps in a simplified way.

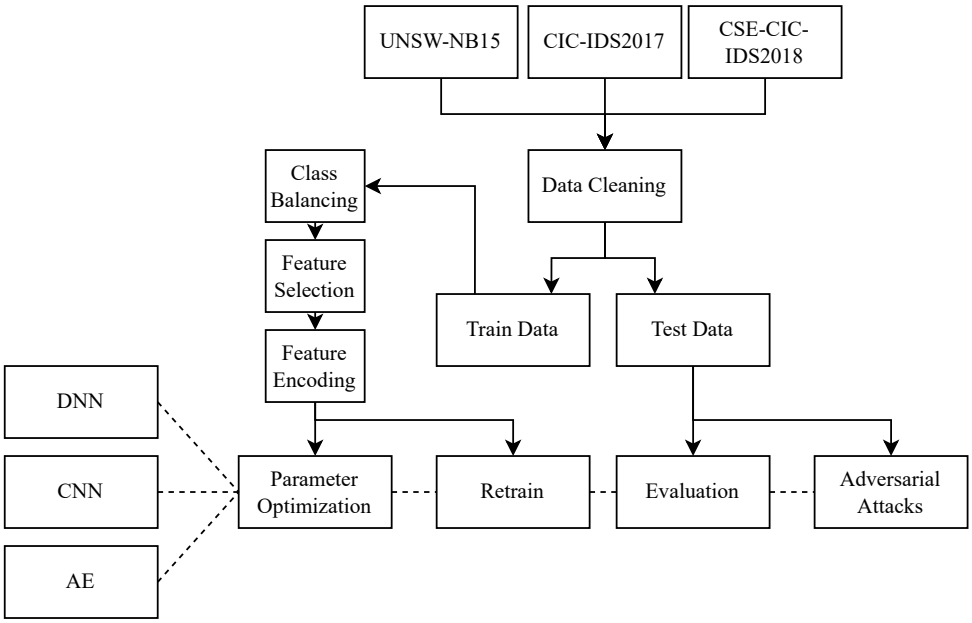


Figure 3: Methods Pipeline

First, the data is cleaned and then split into training and test sets. The training data is used to fit the preprocessing steps and to find the best hyperparameters for each model architecture. The final models are then retrained, evaluated using test data. Finally, adversarial attacks using the test data are applied on the models.

4.1 Datasets

We apply our tests on three commonly used IDS datasets. These datasets contain recent attacks and are rather new compared to other common datasets (e.g., NSL-KDD [99], which includes attack data from the DARPA'98 dataset from 1998). The next three sections describe the datasets in more detail.

4.1.1 UNSW-NB15

The UNSW-NB15 IDS dataset [70] from 2015 consists of 47 features and 10 classes (nine attack categories and one benign class) with over 2.5 million records. From those records, around 2.2 million are benign traffic and 300 thousand attacks. It aims to improve the disadvantages of the existing and widely used datasets KDDCUP'99 and its improved version NSL-KDD [99]. The KDDCUP'99 dataset includes specific TTL values for benign or attack traffic, which leads to classifiers focusing mainly on this feature without learning the characteristics of the different attack types. Additionally, the possible TTL values vary between the test and training sets. Another issue is the distribution of the test set, which is different from the training set

since additional attack samples were added to it [99]. However, the main problem of both the KDDCUP'99 and NSL-KDD dataset is their lack of modern and up-to-date attacks.

The IXIA traffic generator is used to generate both benign and attack traffic. It is employed on three servers, two generating benign and one generating malicious traffic. The traffic is captured on a router directly connected to one of the benign servers and the malicious server, while the benign traffic from the last server is routed through another router and a firewall before reaching the capturing router [70]. From the captured traffic, features are extracted using Argus, Bro-IDS and additional extraction algorithms. The resulting features can be grouped into five categories: flow features, basic features, content features, time features and additional features, which in turn can be broken down into general purpose and connection features. The connection features are added to identify attacks in connection scenarios by capturing similar characteristics for each 100 connections in sequence [70].

The records are binary labelled, discriminating between benign and attack traffic. Attack records are further classified into nine categories. These attack categories contain many attacks that follow a common strategy by which they are grouped (*Fuzzers*, *Analysis*, *Backdoors*, *DoS*, *Exploits*, *Generic*, *Reconnaissance*, *Shellcode*, *Worms*) [70]. Unfortunately, this grouping of attacks leads to overlaps between classes and large differences between samples inside a class, and thereby increases the difficulty of classification.

4.1.2 CIC-IDS2017

The CIC-IDS2017 dataset [93] from 2017 includes 12 different attacks with 80 extracted features and a total of over 3 million records, including approximately 2.3 million benign and 500 thousand attack records (the rest are without label). It claims to fulfill all 11 evaluation criteria for IDS datasets from Gharib et al. [33]. See Table 4 for more details on the criteria.

Table 4: Criteria for IDS Datasets according to Gharib et al. [33]

Criterion		Short Description
1.	complete network configuration	realistic network with all components including PCs, servers, router, and firewall
2.	complete traffic	proper traffic generation technique
3.	labeled dataset	correctly labeled data with information about the specific attack
4.	complete interaction	data includes all network interactions
5.	complete capture	complete captures without removal of seemingly non-functional data
6.	available protocols	protocols of normal user interactions in addition to attack related protocols
7.	attack diversity	diverse attack types
8.	anonymity	privacy conformity without removal of important information
9.	heterogeneity	complete information covering all aspects of the detection process
10.	feature set	proper feature analysis and extraction
11.	metadata	proper documentation regarding network configuration, attack scenarios and extracted features

The testbed includes two networks, one victim network and one attacking network. The victim network includes all common equipment (e.g., router, firewall, switch) and different versions of different operating systems. The attacking network consists of one router, switch, and multiple attacking PCs. The main switch in the victim network is configured to capture all traffic passing it from either direction.

From the captured traffic, the final 80 features are extracted using the CICFlowMeter [58]. This application generates bidirectional flows and includes information about both forward and backward directed traffic. A flow is terminated by either connection teardown in the case

of TCP or a predefined flow timeout for UDP traffic. The features include different metrics of the flow with extracted statistical properties (mean, standard deviation, maximum, minimum, total).

The traffic generation is based on profiles for benign (*B-Profile*) and attack scenarios (*M-Profile*) [92]. The *B-Profile* aims to extract the behavior of human users by using machine learning techniques. Profiles are created by recording real user activity daily for individual profiling of the five most popular protocols (HTTP, HTTPS, FTP, SSH, email protocols). The second step is clustering, which aggregates similar behaviors of users by using XMeans clustering. XMeans clustering is an extension to KMeans with a variable number of clusters [79]. Additionally, browsing behavior is emulated using a modified approach using web-crawling techniques for more realistic requests. The derived profiles are then used to generate benign traffic. Regarding *M-Profiles* for attack generation, six attack scenarios are used. The scenarios are the infiltration from inside (exploitation of a document viewer through email and scan of internal vulnerabilities), unsuccessful infiltration from inside (unsuccessful exploit attempts), denial of service (HTTP DoS against web servers), web application attacks (e.g., SQL injection, unrestricted file upload), brute force attacks (dictionary-based brute force attack against different services) and recent attacks (famous zero-day vulnerabilities) [92]. The final set of attacks includes 14 classes (*Bot*, *DDoS*, *DoS GoldenEye*, *DoS Hulk*, *DoS Slowhttptest*, *DoS slowloris*, *FTPPatator*, *Heartbleed*, *Infiltration*, *PortScan*, *SSHPatator*, *Web Attack Brute Force*, *Web Attack Sql Injection*, *Web Attack XSS*).

4.1.3 CSE-CIC-IDS2018

The CSE-CIC-IDS2018 dataset [26] from 2018 is a successor to the CIC-IDS2017 dataset and uses the same feature extraction method (CICFlowMeter) and traffic generation techniques based on profiles. In comparison to its predecessor, the used network infrastructure is much larger. The attacking network consists of 50 machines, while the victim organization includes five departments with in total 420 machines and 30 servers. With over 16 million records, including around 13 million benign and 2.7 million attack records, the dataset is more than five times as big as the CIC-IDS2017 dataset. Regarding attack classes, the dataset includes 14 different classes with partial overlaps to the attacks of the CIC-IDS2017 dataset (*Bot*, *Brute Force Web*, *Brute Force XSS*, *DDOS attackHOIC*, *DDOS attackLOICUDP*, *DDoS attacksLOICHTTP*, *DoS attacksGoldenEye*, *DoS attacksHulk*, *DoS attacksSlowHTTPTest*, *DoS attacksSlowloris*, *FTPBruteForce*, *Infilteration*, *SQL Injection*, *SSHBruteforce*).

4.2 Preprocessing

The data of the three described datasets is available through CSV files, including the extracted features with the corresponding labels for each flow. But the data still needs to be further processed to use it for classification because the datasets contain erroneous and non-numeric data. Additionally, the datasets show large variations in the number of samples per class, which can affect the classification process. Therefore, we apply four preprocessing steps, including data cleaning, class balancing, feature selection and feature encoding.

4.2.1 Data Cleaning

Cleaning of the available data consists of eight steps, including the removal of unneeded features and rows, the replacement of certain values and the filtering of classes. We a) remove all time and address related features (e.g., timestamps, IP addresses, port numbers) and flow IDs. Additionally, we b) remove all features with a standard deviation of 0 and c) with a correlation

coefficient of 1 or -1. Such features contain no additional information and are therefore not helpful for classification. The datasets contain missing and infinite values, which can cause problems regarding normalization. We d) replace missing values of numeric features with -1 and introduce a new category *missing* for categorical features. Regarding infinite values, we e) change all values which are ∞ or $-\infty$ to 0. To not lose any information, a new feature is introduced which is either 0 or 1, indicating whether the corresponding value was infinite. Some of the datasets contain rows without an attack label, which are not usable for classification. We f) remove rows without label and further filter the rows by g) removing duplicate rows while keeping the first. Rows that are complete duplicates except the label are completely removed since they cannot be distinguished by any classifier. Lastly, h) classes with less than 50 samples are removed. Since we apply oversampling based on existing samples (see Section 4.2.2), a minimum number is required to derive new samples. In particular, the oversampling strategy requires at least six samples, but since we apply oversampling for the training data only, the total number of samples needs to be higher. The minimum number is then calculated by the training split of 0.6 and the cross validation split of 0.2, which is $\frac{1}{0.6} \cdot \frac{1}{0.2} \cdot 6 = 50$.

4.2.2 Class Balancing

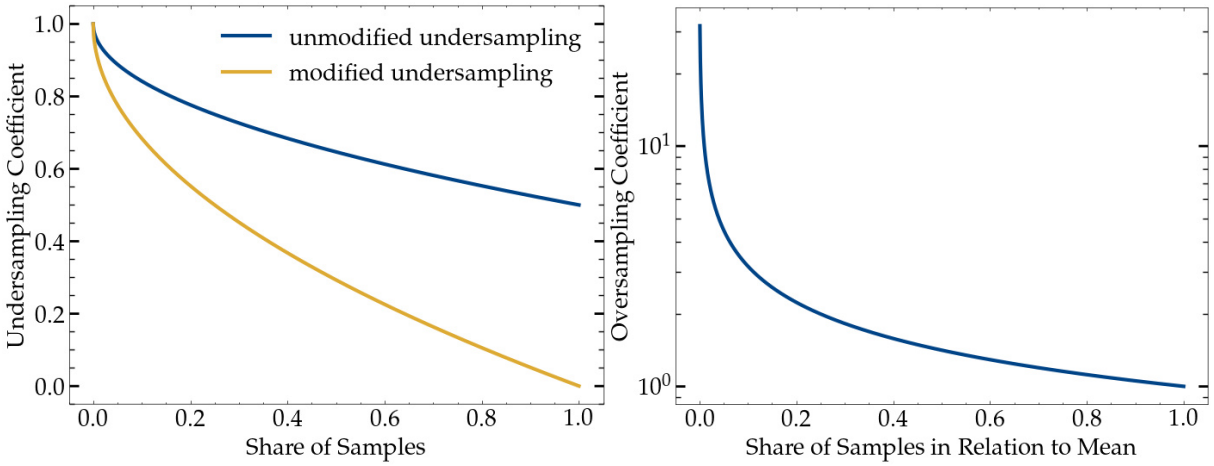
The class distribution in all three datasets is imbalanced, with classes taking up to 95% of an entire dataset. Classification on balanced data can improve overall classification performance [39], which is the reason we apply two sampling methods to change the class distribution. But a sampling strategy has to be chosen carefully. While the random removal of samples can help in balancing the data, removing too many samples may cause the classifier model to lose important concepts. Oversampling by duplicating samples to increase the number of samples in under-represented classes, on the other hand, may lead to overfitting because the classifier learns the specific attributes of the over-representation of specific samples [39].

We employ a modified version of the reduced skewed fixed-ratio undersampling method introduced in [13] for IDS datasets. Fixed-ratio undersampling randomly removes samples using a fixed ratio. We do not use an informed undersampling approach, which incorporates the features of the samples because of the high computing power that is required. The undersampled number of samples for a class with N samples, where s is the share regarding all classes, is $N\,(1 - \sqrt{s})$. This function reduces the skew of the class distribution non-linearly, penalizing over-represented classes stronger. The introduced modification of the function from the original version $N\left(1 - \frac{\sqrt{s}}{2}\right)$ increases the penalization of over-represented classes. Therefore, the class distribution is more balanced but as a trade-off more samples are removed which may lead to missing important information. Figure 4a shows both the unmodified and modified undersampling strategies. We decide to use the described undersampling strategy because of its balance between solving the imbalance problem, decreasing the necessary time for training and avoiding loss of information.

After undersampling, the number of samples of over-represented classes are reduced, but minority classes are still under-represented. Therefore, we employ oversampling using SMOTE [18]. In comparison to oversampling by duplication, SMOTE avoids the overfitting problem by adding synthetic samples of minority classes. Lee and Kim [59] analyze different oversampling sizes and discover that it is not necessary to attain complete class balance, and adding less samples can even lead to better results. Additionally, SMOTE uses existing samples to generate the synthetic samples. Therefore, with a low number of original samples and a high number of added synthetic samples, SMOTE can still lead to problems with overfitting. We follow this observation and increase the number of samples of classes with fewer samples than the mean, non-linearly, as an inverse approach to the undersampling strategy. With the mean

M of the number of samples in a class, the oversampled number of samples for a class with N samples, where $N < M$, is $N\sqrt{\frac{M}{N}}$. Figure 4b shows the oversampling strategy in relation to M.

Regarding the binary classification case, all samples labeled with an attack are combined in one class *Attack*. Therefore, the number of samples in this class is sufficient for classification and no oversampling needs to be done. Following the approach to binary classification in [53], we do not oversample individual classes inside *Attack*. We undersample the larger class to get an equal number of samples for both classes.



(a) Unmodified and Modified Undersampling Strategy

(b) Oversampling Strategy

Figure 4: Under- and Oversampling Strategies

4.2.3 Feature Selection

Feature selection is a technique to reduce the number of features given to a classification algorithm. Depending on the algorithm, this technique can increase classification accuracy, avoid overfitting of the model and decrease model complexity, thereby making the model easier to interpret [45, 1].

There are various ways to select features, which are not all tested in this work. Since the datasets are publicly accessible and used, we test feature subsets from different publications. Moustafa and Slay [69] analyze the features of the UNSW-NB15 dataset. They extract a subset of features for each attack class of the dataset by applying a method utilizing an association rule mining algorithm (ARM). Janarthanan and Zargari [46] reduce the most frequent features of the class specific feature sets to a subset of eight features (*ARM-based*), which is one of the subsets we include in the test for the UNSW-NB15 dataset. Additionally, they propose a set of five features selected by machine learning techniques from various attribute selection methods (*ML-based*). Another approach to ranking the feature importance of the UNSW-NB15 dataset uses a discrete variant of the cuttlefish algorithm (D-CFA) by first calculating the dependency between features and classes and testing the classification performance using a Neural Network [7]. We evaluate two subsets using the 11 and 20 most important features according to this method (*D-CFA-based*).

Regarding the CIC-IDS2017 dataset, Kurniabudi et al. [96] study the relevance of the features and their influence on the classification accuracy of different classifier algorithms. They use information gain to rank the features. From that ranking, we remove *Destination Port*, since it is address-related, and redundant features that have a correlation coefficient of 1, see Sec-

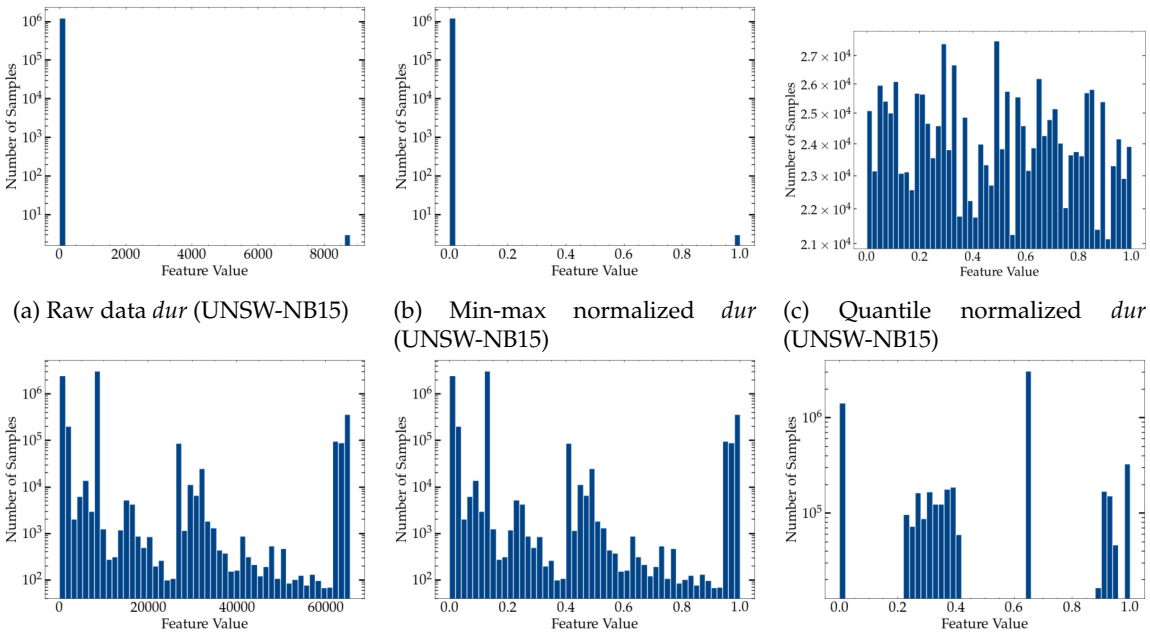
tion 4.2.1 for more details. We test three different subsets taking features with an information gain of above 0.5, 0.4 and 0.3 with 13, 20 and 33 features, respectively (*Information Gain-based*).

Since the features of the CSE-CIC-IDS2018 dataset are almost identical to the features of the CIC-IDS2017 dataset, we test the classification performance of the above subsets by translating the feature names to the corresponding features of the CSE-CIC-IDS2018 dataset. The resulting subsets differ slightly in the number of features, which is due to differences in the data cleaning process (e.g., different redundant features). The subsets consist of 14, 21 and 36 features (*Information Gain-based*). The transferability of the subsets between different datasets is questionable. Therefore, we test another feature subset based on recursive feature elimination with decision trees (DT-RFE) from [86]. This method tests the accuracy of a decision tree using different feature subsets to find the optimal subset and rank the features by importance. The resulting subset consists of 12 features (*DT-RFE-based*).

We compare the classification results using these subsets with a correlation-based approach. In this approach, we remove features that have an absolute Pearson correlation coefficient of above a predefined threshold (*Correlation-based*). The removal of features based on correlation coefficients in the context of intrusion detection datasets can increase the performance of classifiers [45, 28]. Additionally, high correlation among features reduces the model interpretability using SHAP, which works on an independence assumption to compute its scores [1]. All feature selection methods, including different correlation thresholds (0.9, 0.7, 0.5, 0.3, 0.1) are tested by applying cross validation, see Section 4.3.2 for further details.

4.2.4 Feature Encoding

The resulting data from the previous preprocessing steps consists of either categorical or numeric features. To be able to use those features using Deep Learning models, the categorical features need to be transformed to a numeric representation and already numeric features need to be normalized to values in $[0, 1]$, to change the values of different features to a common scale. We transform categorical features to vectors using one-hot encoding, with each value representing one category. The resulting number of features is for each feature increased by the number of unique categories. Regarding numeric features, we test two normalization techniques, min-max normalization and quantile normalization. Min-max normalization transforms data linearly, therefore preserving the scale of the original data. For a value $x \in X$, the normalized value is defined as $\frac{x - \min X}{\max X - \min X}$. Quantile normalization on the other hand transforms data by mapping original values to a uniform distribution. Therefore, it is a non-linear transformation and changes the distribution of the data. The advantage of quantile normalization in comparison to min-max normalization is its robustness to outliers [2]. Figure 5 shows the distribution of the original data and after applying each normalization technique of the features *dur* (total flow duration) of UNSW-NB15 (Figures 5a, 5b, 5c) and *Init Fwd Win Byts* (number of bytes of the initial window sent in forward direction) of CSE-CIC-IDS2018 (Figures 5d, 5e, 5f). Both techniques have individual advantages that depend on the given data. The histogram of the raw *dur* feature shows that the values are mostly the minimum or maximum value of the feature. With min-max normalization, this distribution is maintained, leading to difficulties in recognizing small differences in the data. Quantile normalization, on the other hand, can transform the data to a uniform distribution and makes the data samples distinguishable. Feature *Init Fwd Win Byts* on the other hand, shows already a mostly uniform distribution in the raw data, which makes min-max normalization the better fit. The best normalization technique for the given data is tested by applying cross validation, see Section 4.3.2 for further details.



(a) Raw data *dur* (UNSW-NB15)

(b) Min-max normalized *dur* (UNSW-NB15)

(c) Quantile normalized *dur* (UNSW-NB15)

(d) Raw data *Init Fwd Win Byts* (CSE-CIC-IDS2018)

(e) Min-max normalized *Init Fwd Win Byts* (CSE-CIC-IDS2018)

(f) Quantile normalized *Init Fwd Win Byts* (CSE-CIC-IDS2018)

Figure 5: Comparison of Min-Max and Quantile Normalization using Histograms of Selected Features

4.3 Classification

Classification combines the described preprocessing pipeline with the application of a classifier model that uses the preprocessed data to predict the class of a given data point. To find the best model for this task, we first search for the best hyperparameters for each model architecture and the most effective feature selection technique. We use three different deep learning architectures, namely deep forward networks (DNNs), convolutional neural networks (CNNc) and autoencoders (AEs).

4.3.1 Classifier Models

The DNN is based on multiple fully connected layers with the ReLU activation function. The output layer uses the softmax function as activation to obtain class predictions. The number of layers and nodes per layer is optimized in the hyperparameter search process.

The CNN architecture consists of a variable number of blocks, each with a 1D max pooling layer with a pool size of 2 following a 1D convolution layer. The convolution layers use zero-padding equally applied to the left and right end of the data. The output from these chained blocks is flattened and used as the input for fully connected layers with dropout. Dropout randomly drops inputs at a certain rate, which reduces the complexity of the model and helps to avoid overfitting [95]. We use ReLU as the activation function for both the convolution and fully connected layers, since it is the state-of-the-art activation function for deep learning [5]. Similarly to the DNN, we apply softmax as activation of the last layer. Optimization of the hyperparameters is used for the number of convolution blocks, the number of filters in each convolution layer, the kernel size, the number of fully connected layers and the number of nodes per layer as well as the dropout rate. The loss function of both DNN and CNN is the sparse categorical cross entropy.

The AE architecture is based on a stacked sparse autoencoder. First, the autoencoder consisting of an encoder and a decoder is trained unsupervised to reconstruct the input. The encoder is built with multiple fully connected layers using the ReLU activation function. The number of nodes in the last layer of the encoder in relation to the original input size determines the compression rate. Additionally, an activation penalty using the L1 norm is applied to the outputs of this layer to obtain sparse representations. The decoder mirrors the structure of the encoder, with sigmoid as the activation for the last layer to be able to reconstruct the input. From the trained autoencoder, we extract the encoder and use it as the first layers of the final classifier model. Additionally, a variable number of fully connected layers with ReLU in the first layers and softmax in the last layer is appended to the model. Before training of the classifier, we freeze the weights of the encoder so that they are not changed during training. The loss function of the autoencoder is optimized to either be the mean-squared error or the cosine similarity. The cosine similarity can be advantageous for sparse data, which is the case when having a high number of one-hot encoded inputs. Other optimized parameters are the compression rate, the number of layers of both the autoencoder and fully connected layers, the number of nodes per layer and the activation penalty factor. The parameters of the autoencoder are optimized for the classification results instead of the reconstruction success.

All models are trained using the Adam optimizer [55] with a batch size of 32 and an individually optimized number of epochs and learning rate. Adam extends stochastic gradient descent using a variable learning rate and therefore converges more quickly than other optimizers [55]. The full search space for each architecture can be seen in Tables A.1, A.2 and A.3.

4.3.2 Model Selection

We first apply the data cleaning steps to the entire dataset, see Section 4.2.1 for more details. The resulting data is split into a training and test set with a ratio of 0.6. The training data is used in a two-step process to find the optimal model for each architecture. First, we test the model architecture-specific hyperparameters using random search with 5-fold cross validation. Random search randomly generates values for each hyperparameter following predefined distributions. In comparison to grid search, random search has several benefits, including better efficiency and easier application [11]. We apply 16 iterations of random search for both the DNN and CNN architectures and use 32 iterations for the AE architecture, since the parameter search space is significantly bigger. The hyperparameters that achieve the highest mean macro-averaged F1 score across the five cross validation splits are considered the best parameters. We follow the suggestion of Bergstra and Bengio [11] regarding random search in a controlled setting and use the resulting best hyperparameters to test the different feature selection techniques using grid search with 5-fold cross validation. The different feature selection techniques and subsets for each dataset are described in detail in Section 4.2.3. We fit and apply the preprocessing pipeline consisting of first undersampling, then feature selection, feature encoding and finally oversampling to the training data of each cross-validation split. The fitted feature selection and encoding methods are applied to the test data of each split. Regarding feature selection and encoding, fitting using only the training data in each split prevents data leakage between the training and test data, which otherwise could lead to higher test results than using completely unseen data. Resampling methods for class balancing are only applied to the training data to not modify the test results, especially since oversampling generates synthetic samples.

Finally, the best model regarding hyperparameters with the selected feature selection technique is retrained using the entire training data. The model is then evaluated using the separate

and completely unknown test set to validate the hyperparameter selection process and prevent overfitting.

4.4 Adversarial Attacks

To test the vulnerability of the trained models, we apply various targeted adversarial attacks on them by perturbing samples from the unseen test data with the goal of changing the class prediction from any attack class to *Benign*. The success of an attack is measured by the attack success rate (ASR), which is the ratio of samples whose prediction is successfully changed to the target class. We apply white-box attacks that have information about the weights of the model, substitute attacks that use partial training data and black-box attacks that only require partial information or access to the model. We describe the attacks we use in the next three sections. Additionally, Table 5 gives a short overview of the information each attack requires about the victim model and a brief description.

Table 5: Overview of Adversarial Attack Methods

Method	Required Information	Short Description
BIM	model weights	step-wise fixed-size perturbations using gradients
PGD	model weights	BIM with random start
DeepFool	model weights	BIM with adaptive step size
C&W	model weights	perturbation minimization using gradient descent
Substitute	partial training data & model structure	transfer of white-box attacks using substitute model
ZOO	class probabilities	C&W with estimated gradients
Boundary	class prediction	step-wise reduction of random perturbation
HopSkipJump	class prediction	query-efficient version of Boundary using gradient estimations

4.4.1 White-Box Attacks

We test four different white-box attacks, namely the Basic Iteration Method, Projected Gradient Descent, DeepFool and the Carlini-Wagner attack. These attacks are the most common in the field of adversarial attacks and also cover two categories of white-box attacks based on fixed-size and minimal perturbations. All the attacks require access to the weights of the victim model and use this information to compute adversarial examples.

The Basic Iteration Method (BIM) [57] is an iterative extension to the fast gradient sign method (FGSM) [35]. Goodfellow et al. describe the computation of an optimal perturbation as a linear optimization problem, which results in the FGSM attack based on the gradients of a neural network [35]. This method has one hyperparameter ϵ which controls the size of the perturbation. BIM applies FGSM multiple times with a small step size, clipping the intermediate values at each step according to the parameter ϵ. Both FGSM and the original BIM are untargeted and create perturbations with the goal of misclassification. The authors of BIM propose a variation to create targeted adversarial examples by following the gradient of a specific class instead of using the negated gradient of the original class [57]. We use the targeted version of BIM to create adversarial examples. Apart from the perturbation size (ϵ), the attack has two hyperparameters, the number of steps and the step size.

Projected Gradient Descent (PGD) is an optimization method that can be used to create adversarial examples [66]. PGD generally follows the same algorithmic as BIM and applies FGSM in multiple steps. The only difference between PGD and BIM is the initialization. PGD is initialized randomly in the ball of interest, determined by the parameter ϵ. BIM, on the other

hand, uses the unperturbed original sample as the starting point. PGD has equally to BIM only the number of steps and the step size as hyperparameters.

Both BIM and PGD compute perturbations limited by a fixed parameter ϵ. Therefore, the generated perturbations are not minimal. Moosavi-Dezfooli et al. [67] on the other hand propose the DeepFool algorithm to compute minimal adversarial perturbations. The attack procedure first finds the direction of perturbation by calculating the orthogonal projection distance. Using the direction, a minimal perturbation is found through iteration [110]. The algorithm operates greedily and is not guaranteed to find minimal perturbations. It is similar to BIM but uses an adaptive step size in each step of gradient descent to reduce the applied perturbation [67]. Hyperparameters of DeepFool which we optimize are the number of considered most likely classes and a boundary overshoot parameter.

Carlini and Wagner [15] introduce an attack algorithm to find minimal perturbations for the L_0, L_2 and L_∞ distance metrics (C&W Attack). The algorithm tries to minimize the applied perturbation by using gradient descent with the Adam optimizer [55]. The constant c controls the step size of gradient descent and can, if too small or too large, lead to unsuccessful or suboptimal adversarial examples. The optimal parameter c is found using binary search. Regarding white-box adversarial attacks, the C&W Attack is currently considered the state-of-the-art attack to find adversarial examples with minimal perturbations but requires a higher computational effort than BIM or DeepFool [6]. We test two implementations of this algorithm from different packages. We optimize the step size, learning rate and the initial value of c.

4.4.2 Substitute Attacks

Substitute attacks utilize the power of white-box attacks without requiring access to the model weights. We train a substitute model for each model architecture with the same hyperparameters using 10%, 1% and 0.1% of the original training data. After training, we apply the same white-box attacks as described in Section 4.4.1 to the substitute model to create adversarial examples which later can be evaluated against the victim model. The success of this attack type depends on the transferability of the adversarial examples to the victim model.

4.4.3 Black-Box Attacks

In the black-box setting, the information and access to the model is limited. We test three black-box attacks, including Zeroth Order Optimization, the Boundary Attack and the HopSkipJump Attack. Similar to the white-box attacks, these attacks are very popular and cover different types of black-box attacks using gradient estimation, binary search and a hybrid approach.

Zeroth-order optimization describes the optimization of a function using only function values and without information about gradient values. Chen et al. propose a method to generate adversarial examples based on Zeroth-Order Optimization (ZOO) [20]. ZOO estimates the gradients of a model using the class probabilities. The gradient estimations can be used to apply the C&W Attack or any other white-box attack and obtain adversarial examples similar to the white-box scenario. Since gradient estimation is computationally expensive, the attack method uses importance sampling with predefined regions of an image (e.g., higher importance to central regions than corners). For IDS data, this computational optimization is not applicable.

The Boundary Attack [12] is a decision-based black-box attack. It only requires the predicted class of the victim model and no information about class probabilities. The attack starts from an initial random perturbation, which achieves the adversarial goal. This perturbation is reduced step-wise while staying adversarial, resulting in an adversarial example with a small perturbation size.

While the Boundary Attack can find adversarial examples only using the decisions of a classifier, the required number of queries is very high. A high number of queries makes an attack harder to apply and easier to detect. The HopSkipJump Attack [19] tries to minimize the required number of queries to the victim model while still requiring only the resulting decisions. The algorithm is similar to the Boundary Attack and starts with a perturbed sample. It is based on multiple iterations, each including a binary search towards the decision boundary and an estimation of the gradient direction, similar to ZOO but based purely on decision information. HopSkipJump can produce adversarial examples more query-efficient than the Boundary Attack, but is computational expensive.

4.4.4 Hyperparameter Optimization

We test different hyperparameters for white-box and substitute attacks using random search, as full model access can be assumed in the white-box and substitute attack scenarios. In the case of substitute attacks, the random search is applied to the trained substitute model instead of the victim model. We do not use train/test splits or cross-validation because no fitting is necessary and full data access can be assumed, which is the reason the attacks do not need to be tested using unknown data. To achieve the best results, we apply the hyperparameter optimization of the different attack methods separately for each dataset, model and class. Since the number of required tests is considerable large, we use only up to 1000 randomly selected samples of each class for parameter optimization. Black-box attacks do not have unrestricted access to the model. Hyperparameters could still be optimized using random search, but this approach would drastically increase the number of queries to the model, rendering the approach less realistic. Therefore, we assume the best hyperparameters using default values and minor testing.

4.4.5 Domain Constraints

All the used attacks are designed for image classification. The main difference is that in image classification, each pixel can be perturbed and only the size of the perturbation is relevant. The perturbation size can be measured and, furthermore, human agents can evaluate the resulting adversarial examples further if they still match their original class. The data for intrusion detection, on the other hand, has a different structure. Foremost, it is one dimensional and the order does not follow spatial properties, which already renders some attacks unusable. Secondly, not all features can be perturbed, and the possible perturbation size varies across features. When applying perturbations on all features of a sample, the underlying intrusion is most likely not possible anymore. Additionally, the data includes categorical values which are one-hot encoded. Such features can only hold binary values with the additional restriction that one and only one of them is 1, which makes them imperturbable. Determining which feature can be changed and by how much is difficult, especially given the sparse documentation of the datasets.

We therefore test all attacks (white-, black-box and substitute attacks) using feature subsets with features that most likely are generally possible to perturb. Zhang et al. [113] describe features which are amendable for the CSE-CIC-IDS2018 dataset. Their feature subset is based on previous work of Kuppa et al. [56], which test the possibility of perturbations by modifying original *pcap* files to make sure that possible adversarial examples are not discarded before reaching the victim IDS system. It is important to note that their work only tests the general validity of packets and not a successful intrusion. The proposed subset contains time-based features in both directions (source to destination and destination to source) of the inter arrival time (time between two packets), active-idle time (time a flow was idle before becoming active

and vice versa) and the average number of bytes in the initial window and sub-flows. We call the resulting feature subset *Time-based*.

Another approach to define a perturbable feature subset from Teuffenbach et al. [100] uses a grouping of flow-based features of the CIC-IDS2017 dataset, as proposed by Hashemi et al. [37]. The groups are sorted by increasing difficulty of applying changes in a realistic setting. The authors exclude all features which are metrics of packets in backward direction or based on general flow data, with the assumption that an attacker cannot change features based on traffic from other hosts. The remaining subset consists only of metrics purely based on traffic in the forward direction, which we call *Source-based*.

Since both the CIC-IDS2017 and CSE-CIC-IDS2018 datasets use the same feature extraction method (CICFlowMeter), we test the two subsets (*Time-based* and *Source-based*) for both datasets with translated feature names. The UNSW-NB15 dataset includes different types of features, and we cannot find previous work on subsets of amendable features for this dataset. Therefore, we follow the *Source-based* approach [100] and filter the features to only include metrics in the forward direction. We remove categorical and additional generated features, which contain information about previous flows, making them more difficult to change.

4.4.6 Single Feature Attacks

We further test the vulnerability to adversarial attacks and analyze the effect of each feature. In this approach, we test the ASR for each feature if only this one feature can be altered. We use the optimized hyperparameters using the full feature set for the single feature attacks. From this process, we get an ASR ranking for each feature and class. These features can be further evaluated for the possibility of applying perturbations in a real-world setting. In the white-box setting, the single feature tests are directly applied on the victim model. For substitute attacks, the same procedure is used on the trained substitute model and only the generated adversarial examples are tested against the victim model in the final evaluation. We therefore not only test the general transferability of adversarial examples but also the transferability of single feature perturbations. We do not apply single feature attacks to black-box attacks, since testing each feature requires many iterations, which is not applicable in a black-box setting. A possible extension to the perturbation of single features are pair-wise features, but we decided to only test single feature perturbations, since these require less manipulation and already showed great performances in early tests.

4.4.7 Attack Implementation

We implement white-box and black-box attacks using the libraries *Foolbox* (*FB*) [85] and *Adversarial Robustness Toolbox* (*ART*) [74]. Both libraries claim to be usable with any data type, but have clear restrictions for non-image data. We modify the implemented attacks to be usable with one dimensional data and allow feature restrictions using perturbation masks. Since the attack algorithms are not initially designed to comply with such restrictions, the ASR can be lower than expected. This is especially true for attacks which try to find minimal perturbations, since the enforcement of the perturbation masks may break successful adversarial examples.

Since the attacks are directly applied to the data given to the model instead of the raw data, we implement an attack pipeline which is used for all attacks. First, we filter the data to only include samples with labels of the currently tested class. Furthermore, we remove all samples that are not correctly classified by the model. Therefore, the resulting ASR includes only samples that were successfully perturbed by the attack and not due to shortcomings of the classifier model. Next, we apply the selected feature selection technique. We save the removed

values to be able to restore them on the inverse pass. The samples are encoded using the fitted one-hot encoder and normalization method. The attack is performed on the resulting data samples, creating adversarial examples from the original samples. We create a perturbation mask, defining which feature can be perturbed by either a 1 (can be perturbed) or 0 (cannot be perturbed). All one-hot encoded features are assigned a 0. If specific features are defined (e.g., domain constraints or single feature attacks) only those features are assigned a 1. Despite that we modify the attacks to actively use feature restrictions, the adversarial examples are not guaranteed to only include perturbations in allowed features. Therefore, we enforce the perturbation mask and replace all other values with the original values. Finally, we inversely encode the adversarial examples and restore removed features from the feature selection technique. The final adversarial examples are tested against the model, whether their prediction is successfully changed to the target *Benign*.

5 Results

In the next sections, we will present the results from our methodology. First, we examine the preprocessing steps for each dataset in more detail. Then we present the classification results, followed by the evaluation of the attack results.

5.1 Preprocessing Results

The preprocessing pipeline as described in Section 4.2 is generally the same for all three datasets, but there are different results from the applied methods, which will be described in the following sections.

5.1.1 Preprocessing of UNSW-NB15

Data Cleaning The UNSW-NB15 dataset is the smallest of the datasets with 2540047 records in the raw data. The dataset contains the wrong label *Backdoor*, which we replace with the existing label *Backdoors*. First, we clean the data following the steps as described in 4.2.1. We remove the address related features *srcip*, *sport*, *dstip* and *dsport*, as well as the time related features *Stime* and *Ltime*. All features have a standard deviation of greater than 0, which results in no features being removed. The dataset does not contain duplicate features with a correlation coefficient of 1 or -1, but still contains features with high correlation. The correlation matrix of all features can be seen in Figure 6.

All records are properly labeled, and no values are infinite. The features *ct_flw_http_mthd*, *is_ftp_login* and *ct_ftp_cmd* contain in total a number of 4207903 missing values, which we replace with -1. The dataset contains 525764 duplicate records, not counting the first occurrence of each unique duplicate, which is 20.7% of all records. The classes contain a different percentage of duplicate records, with *Shellcode* containing only 4.17% and *Generic* containing up to 92.01% duplicates. The absolute and relative number of duplicate records for each class can be seen in Figure 7. Since all classes contain a sufficient number of classes for oversampling, all classes are used for classification. The final cleaned UNSW-NB15 dataset contains 2014283 records with 41 features and 10 classes.

Class Balancing The data in the multiclass task is highly imbalanced regarding the number of samples in each class. The class *Benign* forms about 96.23% of the entire dataset, while only 1.26% of the samples are of the next largest class *Exploits*. The class with the lowest number of samples is *Worms* with 95 samples, which is 0.0079% of all samples. After application of the

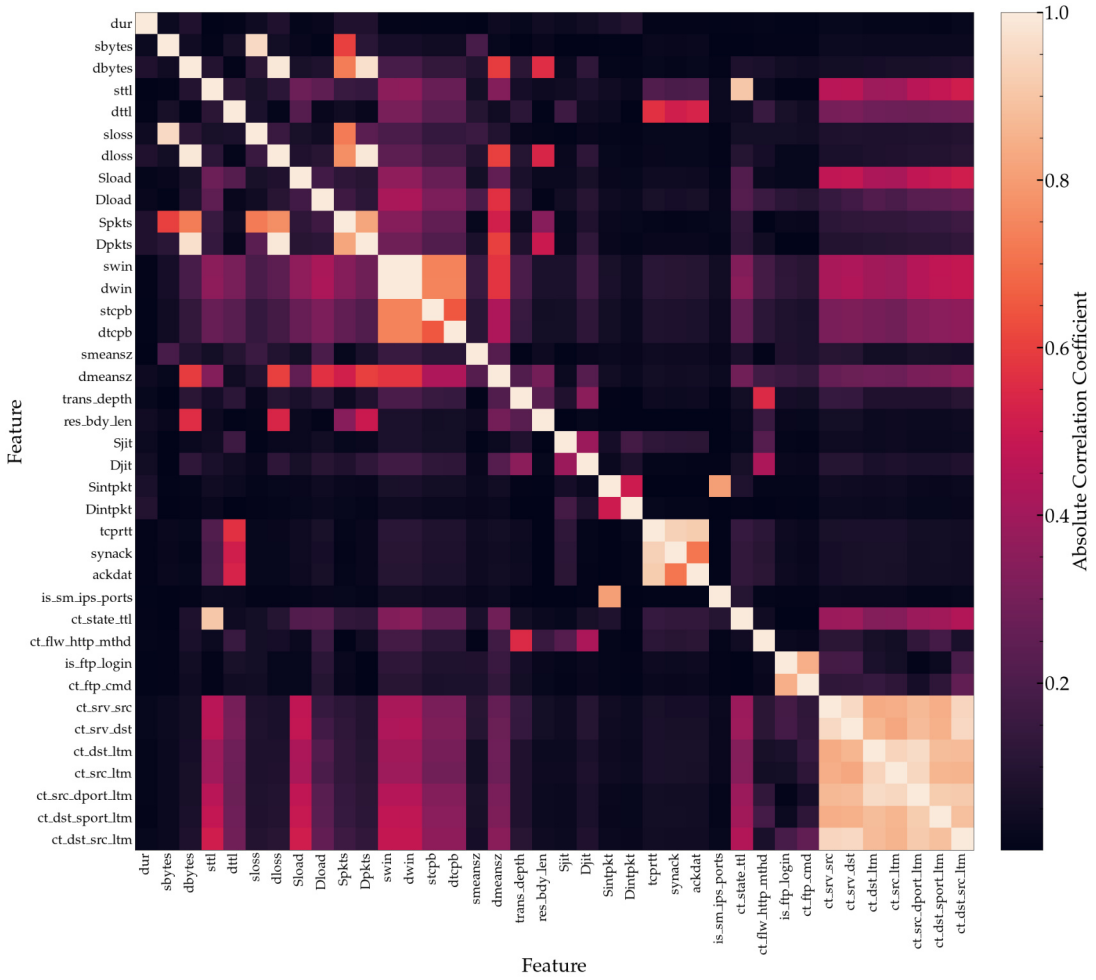
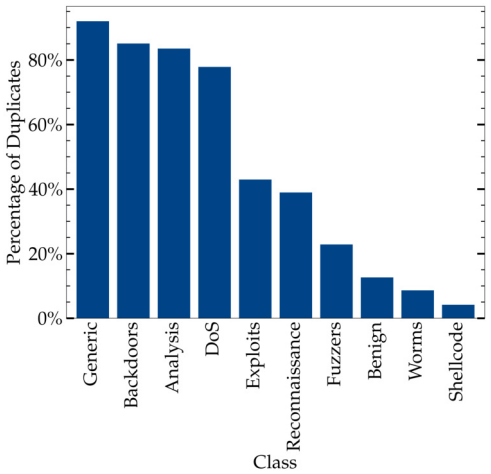


Figure 6: Correlation Matrix of the UNSW-NB15 Dataset

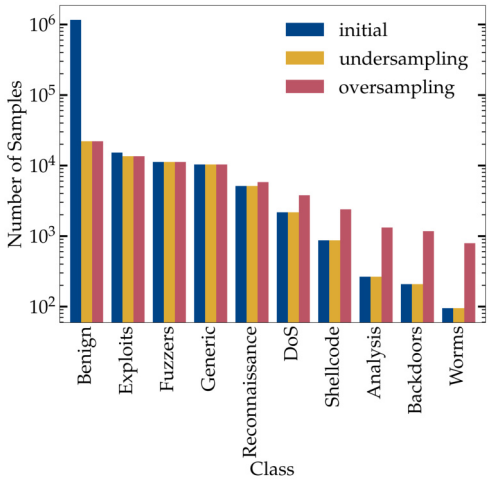


Class	Duplicates	
Generic	198264	92.01%
Backdoors	1982	85.10%
Analysis	2236	83.52%
DoS	12733	77.86%
Exploits	19129	42.96%
Reconnaissance	5449	38.96%
Fuzzers	5542	22.86%
Benign	280351	12.64%
Worms	15	8.62%
Shellcode	63	4.17%
Total	525764	20.70%

Figure 7: Duplicates in Classes of the UNSW-NB15 Dataset

undersampling strategy as described in Section 4.2.2, the number of samples in the training data belonging to *Benign* is reduced by over 98% from 1163047 to 22114 samples. This results in only 33.54% of the data belonging to *Benign* and 20.51% to the next largest class, *Exploits*. This

solves the issue of massively over-represented classes, but the smallest classes are still under-represented, e.g., *Worms* making up 0.14% after undersampling. Therefore, an oversampling strategy is applied using SMOTE. The number of samples in *Worms* is increased by over 733%, from initially 95 samples to 792 with oversampling. The initial class distribution, after applying undersampling and after under- and oversampling, can be seen in Figure 8. Regarding the binary classification task, all attacks are combined in one class (*Attack*). Therefore, no class is under-represented but *Benign* is still over-represented, making up 96.13% of the dataset. After undersampling of the larger class (*Benign*), both classes have 46798 samples. The test data is not resampled, see Section 4.3.2 for more details on the data split and prevention of data leakage.



Class	Initial		Undersampling		Oversampling	
Benign	1163047	96.23%	22114	33.54%	22114	30.52%
Exploits	15238	1.26%	13527	20.52%	13527	18.67%
Fuzzers	11222	0.93%	11222	17.02%	11222	15.49%
Generic	10330	0.85%	10330	15.67%	10330	14.25%
Reconnaissance	5123	0.42%	5123	7.77%	5812	8.02%
DoS	2172	0.18%	2172	3.29%	3785	5.22%
Shellcode	869	0.07%	869	1.32%	2394	3.30%
Analysis	265	0.02%	265	0.40%	1322	1.82%
Backdoors	208	0.02%	208	0.32%	1171	1.62%
Worms	95	0.01%	95	0.14%	792	1.09%

Figure 8: Multiclass Class Distribution of the UNSW-NB15 Training Data after Under- and Oversampling

Feature Selection We test different feature selection techniques based on previous work and a correlation-based approach (see Section 4.2.3) using grid search. Table 6 shows the mean macro-averaged F1 score across the five cross validation splits for each feature selection technique and the baseline model with the full feature set. The techniques are sorted by the number of selected features. The *ARM-based* and *ML-based* approaches do not perform well, which can be explained by the low number of features remaining, but they still achieve better results than *Correlation-based* with a threshold of 0.1, which extracts about the same number of features. The *D-CFA-based* approaches perform better and show only a small reduction in performance while lowering the number of features. However, *D-CFA-based* with 11 features reduces the F1 for the multiclass AE by over 0.1, indicating that the AE architecture is especially sensitive to feature reduction. Finally, the *Correlation-based* approach with a threshold of 0.5 performs the best overall, it almost completely preserves the F1 score while decreasing the number of features needed by over 50% and is used as the feature selection method for the final models.

Feature Encoding The UNSW-NB15 dataset contains three categorical features (*proto*, *state*, *service*), which are one-hot encoded to obtain numeric representations. Feature *proto* contains six unique values (*arp*, *icmp*, *igmp*, *ospf*, *tcp*, *udp*), *state* contains eight (*ACC*, *CLO*, *CON*, *ECO*, *FIN*, *INT*, *REQ*, *RST*) and *service* distinguishes between 13 values, including a category for not specified and 12 services (*dhcp*, *dns*, *ftp*, *ftpdata*, *http*, *irc*, *pop3*, *radius*, *smtp*, *snmp*, *ssh*, *ssl*). Therefore, the number of inputs is increased by 24, resulting in 65 input values in total. We test the normalization techniques (see Section 4.2.4) for numeric features as a parameter of the hyper-

Table 6: Mean Macro F1 of Cross Validation Splits using Feature Selection Methods on UNSW-NB15 Training Data

		Multiclass			Binary		
Method	**#Features**	**DNN**	**CNN**	**AE**	**DNN**	**CNN**	**AE**
Baseline (Full Feature Set)	41	0.5889	0.5917	0.5328	0.9212	0.922	0.9198
Correlation-based (0.9)	30	0.5823	0.5764	0.5186	0.9212	0.9208	0.9192
Correlation-based (0.7)	25	0.5837	0.5657	0.5173	0.9196	0.9206	0.9172
Correlation-based (0.5)	20	0.5843	0.5706	0.5211	0.9194	0.9204	0.9174
D-CFA-based (first 20)	20	0.5737	0.5603	0.5255	0.9213	0.9209	0.9192
Correlation-based (0.3)	13	0.55	0.5296	0.5168	0.9176	0.9188	0.9164
D-CFA-based (first 11)	11	0.5561	0.5345	0.4247	0.92	0.9204	0.9117
ARM-based	8	0.4493	0.4441	0.2973	0.9097	0.9176	0.7915
Correlation-based (0.1)	7	0.3544	0.3334	0.3391	0.9171	0.918	0.8988
ML-based	5	0.4992	0.4941	0.4752	0.8967	0.9147	0.8966

parameter optimization process. For all three model architectures, quantile normalization is selected over min-max normalization.

5.1.2 Preprocessing of CIC-IDS2017

Data Cleaning The raw CIC-IDS2017 dataset, as imported from the provided CSV files, contains 3119345 records. It is therefore slightly larger than the UNSW-NB15 dataset regarding the number of records. The dataset contains eight features with a standard deviation of 0, which means they are constant and contain no useful information. We remove such features (*Bwd PSH Flags*, *Bwd URG Flags*, *Fwd Avg Bytes/Bulk*, *Fwd Avg Packets/Bulk*, *Fwd Avg Bulk Rate*, *Bwd Avg Bytes/Bulk*, *Bwd Avg Packets/Bulk* and *Bwd Avg Bulk Rate*). Figure 9 shows the correlation matrix of the CIC-IDS2017 dataset. There are several features with high correlation coefficients, especially statistical features of the same metric, e.g., *Flow IAT Mean*, *Flow IAT Std*, *Flow IAT Max* which are all based on *Flow IAT*. Features with a correlation coefficient of 1 or -1, showing complete duplicate values, are removed while keeping one of the correlated features. Removed features are notated using the form *removed feature(s)* $\rightarrow$ *kept feature*: *SYN Flag Count* $\rightarrow$ *Fwd PSH Flags*, *CWE Flag Count* $\rightarrow$ *Fwd URG Flags*, *Avg Fwd Segment Size* $\rightarrow$ *Fwd Packet Length Mean*, *Avg Bwd Segment Size* $\rightarrow$ *Bwd Packet Length Mean*, *Fwd Header Length 2* $\rightarrow$ *Fwd Header Length*, *Subflow Fwd Packets* $\rightarrow$ *Total Fwd Packets*, *Subflow Bwd Packets* $\rightarrow$ *Total Backward Packets*.

Around 9.25% of the records are unlabeled and therefore removed. The feature *Flow Bytes/s* contains 1358 missing values, which we replace with -1. Infinite values from the features *Flow Bytes/s* and *Flow Packets/s* are replaced with 0 and we add the new features *Flow Bytes/s_infinite* and *Flow Packets/s_infinite*, showing whether the corresponding value was infinite. Out of the 15 classes, 11 contain duplicates, ranging from 0.01% duplicate records in class *DDos* to 98.84% in class *PortScan*. In total, we remove 596918 or 21.09% duplicate records (see Figure 10). Finally, the classes *Infiltration*, *Web Attack Sql Injection* and *Heartbleed* are removed since they each contain less than 50 samples. The final cleaned CIC-IDS2017 dataset contains 2233757 records with 65 features and 12 classes.

Class Balancing The classes of the CIC-IDS2017 dataset are similarly unbalanced as the classes of the UNSW-NB15 dataset. The *Benign* class forms about 84.95% of the entire dataset, while *Web Attack XSS*, *Bot*, *Web Attack Brute Force* and *PortScan* each represent under 0.1%. In the next steps, the training data is under- and oversampled to reduce the skew of the class distribution. The results from both resampling strategies can be seen in Figure 11. After undersampling, the number of samples in the *Benign* class is reduced by 92.17%. The classes *DoS Hulk* and *DDos*

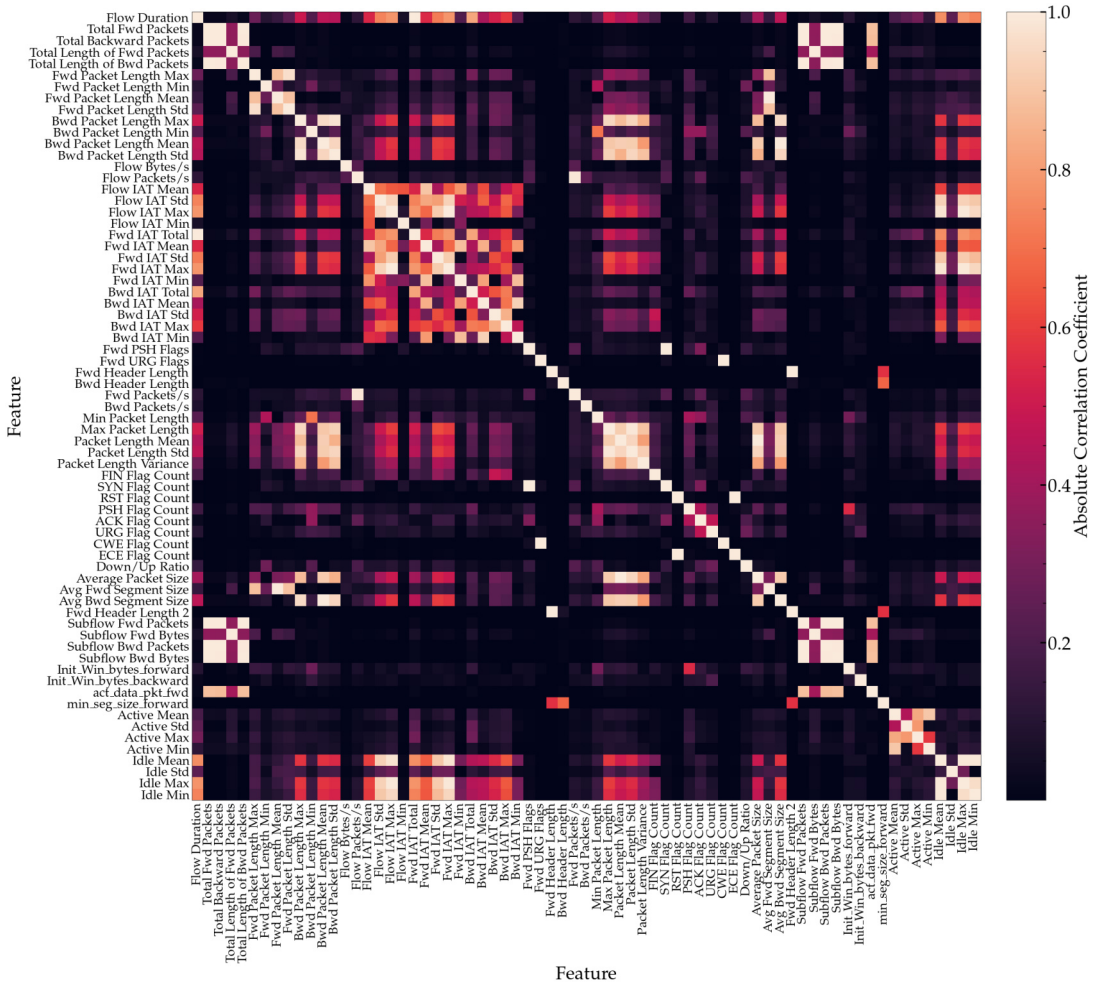
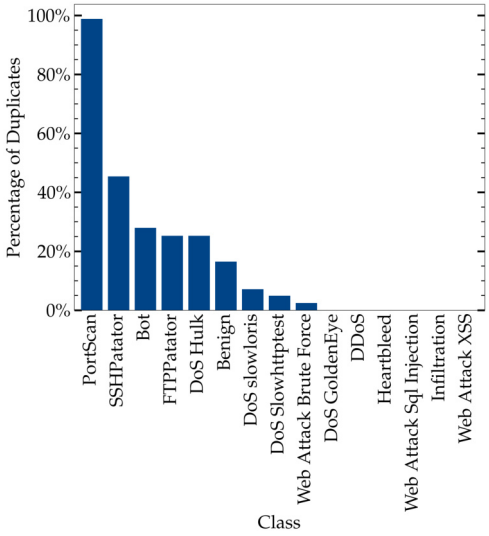


Figure 9: Correlation Matrix of the CIC-IDS2017 dataset

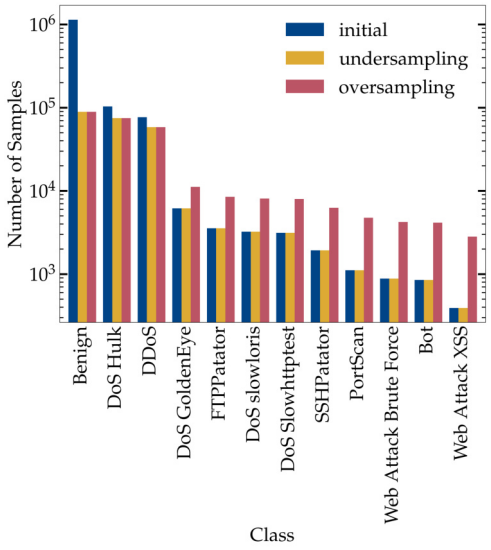


Class	Duplicates	
PortScan	157079	98.84%
SSHPatator	2678	45.41%
Bot	550	27.98%
FTPPatator	2005	25.26%
DoS Hulk	58358	25.26%
Benign	375506	16.52%
DoS slowloris	413	7.13%
DoS Slowhttptest	271	4.93%
Web Attack Brute Force	37	2.46%
DoS GoldenEye	7	0.07%
DDoS	14	0.01%
Infiltration	0	0.00%
Web Attack Sql Injection	0	0.00%
Heartbleed	0	0.00%
Web Attack XSS	0	0.00%
Total	596918	21.09%

Figure 10: Duplicates in Classes of the CIC-IDS2017 Dataset

are reduced as well, but only by 27.81% and 23.94%, respectively. All other classes are oversampled, following the oversampling strategy as introduced in Section 4.2.2. With the combination

of under- and oversampling, the class distribution is fixed to the point that no class has fewer samples than 1% of the data. Considering the binary case, the *Attack* class is undersampled to obtain an equal number of samples (201740) for both classes.



Class	Initial		Undersampling		Oversampling	
Benign	1138554	84.95%	89165	36.59%	89165	31.80%
DoS Hulk	103629	7.73%	74814	30.70%	74814	26.68%
DDoS	76808	5.73%	58421	23.98%	58421	20.84%
DoS GoldenEye	6171	0.46%	6171	2.53%	11194	3.99%
FTPPatator	3560	0.27%	3560	1.46%	8503	3.03%
DoS slowloris	3230	0.24%	3230	1.33%	8099	2.89%
DoS Slowhttptest	3137	0.23%	3137	1.29%	7982	2.85%
SSHPatator	1931	0.14%	1931	0.79%	6262	2.23%
PortScan	1111	0.08%	1111	0.46%	4750	1.69%
Web Attack Brute Force	882	0.07%	882	0.36%	4232	1.51%
Bot	850	0.06%	850	0.35%	4155	1.48%
Web Attack XSS	391	0.03%	391	0.16%	2818	1.01%

Figure 11: Multiclass Class Distribution of the CIC-IDS2017 Training Data after Under- and Oversampling

Feature Selection The results of the different feature selection techniques can be seen in table 7, where the mean macro-averaged F1 score across the five cross validation splits is shown, as well as the number of features that were selected. Generally, the achieved F1 scores change according to the number of features, with the highest score for the baseline model and lower scores for higher reduction rates. The *Information Gain-based* and *Correlation-based* approaches differ not much in this regard, but for a low number of features (13), *Correlation-based* achieves significantly better results. *Correlation-based* with a threshold of 0.5 performs the best when including both the F1 score and the number of features, but shows a higher performance reduction for the AE architecture than for the other architectures. We decided to use *Correlation-based (0.5)* as the feature selection method of the final models because of the high reduction rate of around 64% while maintaining high F1 scores.

Table 7: Mean Macro F1 of Cross Validation Splits using Feature Selection Methods on CIC-IDS2017 Training Data

		Multiclass			Binary		
Method	#Features	DNN	CNN	AE	DNN	CNN	AE
Baseline (Full Feature Set)	65	0.8395	0.8461	0.8143	0.9965	0.9969	0.9916
Correlation-based (0.9)	40	0.8383	0.8407	0.7768	0.996	0.9967	0.9897
Information Gain-based (0.3)	33	0.7766	0.781	0.7321	0.9923	0.9877	0.9792
Correlation-based (0.7)	30	0.7798	0.7799	0.7382	0.9951	0.9952	0.9879
Correlation-based (0.5)	25	0.8268	0.8333	0.7266	0.9934	0.9939	0.9862
Correlation-based (0.3)	22	0.7752	0.7924	0.679	0.9906	0.9916	0.9771
Information Gain-based (0.4)	20	0.7633	0.7719	0.6481	0.9895	0.9901	0.9782
Correlation-based (0.1)	13	0.7189	0.7619	0.5619	0.9898	0.9892	0.9261
Information Gain-based (0.5)	13	0.5851	0.6411	0.5096	0.9815	0.9818	0.952

Feature Encoding Only one feature of the CIC-IDS2017 dataset contains categorical data (*Protocol*) and consequently is one-hot encoded. The feature contains three unique values (*0*, *6*, *17*). The authors of the dataset give no further information about this feature, but the values seem to represent the IANA protocol numbers of TCP (*6*) and UDP (*17*), other packets are labeled as *0*. But the labeling regarding the *Protocol* feature appears to be inconsistent, as Rosay et al. report [88]. Following their analysis, the CICFlowMeter that is used for feature extraction has a false protocol detection and partially mistakes packets by the wrong protocol. Regarding the normalization of numeric features, for all model architectures, quantile normalization achieves the better results in comparison to min-max normalization.

5.1.3 Preprocessing of CSE-CIC-IDS2018

Data Cleaning The CSE-CIC-IDS2018 dataset contains 16232943 records and is the largest dataset of the three. Due to errors in the provided CSV files, the header line is duplicated in random locations of the files. We filter the corrupted lines while importing the data. Next, the features are cleaned. We remove eight features with a standard deviation of 0 (*Bwd PSH Flags*, *Bwd URG Flags*, *Fwd Byts/b Avg*, *Fwd Pkts/b Avg*, *Fwd Blk Rate Avg*, *Bwd Byts/b Avg*, *Bwd Pkts/b Avg*, *Bwd Blk Rate Avg*). The correlation matrix of the CSE-CIC-IDS2018 dataset (Figure 12) shows high correlation coefficients, especially of statistical features that are extracted from the same metric, similarly to the CIC-IDS2017 dataset (e.g., *Flow IAT Mean*, *Flow IAT Std*, *Flow IAT Max*). We combine features with a correlation coefficient of 1 or -1 to one feature. Therefore, we apply the following changes to seven features (*removed feature(s)* $\rightarrow$ *kept feature*): *SYN Flag Cnt* $\rightarrow$ *Fwd PSH Flags*, *CWE Flag Count* $\rightarrow$ *Fwd URG Flags*, *Fwd Seg Size Avg* $\rightarrow$ *Fwd Pkt Len Mean*, *Bwd Seg Size Avg* $\rightarrow$ *Bwd Pkt Len Mean*, *Subflow Fwd Pkts* $\rightarrow$ *Tot Fwd Pkts*, *Subflow Fwd Byts* $\rightarrow$ *TotLen Fwd Pkts*, *Subflow Bwd Pkts* $\rightarrow$ *Tot Bwd Pkts*.

The data contains no records without proper label. We replace 59721 missing values in *Flow Byts/s* with -1 and replace 131799 infinite values in *Flow Byts/s* and *Flow Pkts/s* with 0. Additionally, we introduce two new features *Flow Byts/s_infinite* and *Flow Pkts/s_infinite*, indicating whether the corresponding feature was infinite. Overall, the dataset contains 5454384 (33.6%) duplicate records, which we remove. The classes *FTPBruteForce* and *DoS attacksSlowHTTPTest* contain almost no unique records, with 99.98% and 99.97% duplicates, respectively. Other classes contain almost no duplicates or no duplicate records at all, e.g., *DDos attacksLOICHTTP* (0.14%) and *DDos attackLOICUDP* (0%). We remove such duplicates while keeping the first, see Section 4.2.1 for more details. The share of duplicates for each class can be seen in Figure 13. Finally, we remove the classes *DoS attacksSlowHTTPTest* and *FTPBruteForce* since they contain less than 50 samples. The cleaned CSE-CIC-IDS2018 dataset contains 10778476 total records with 64 features and 13 classes.

Class Balancing The class distribution of the CSE-CIC-IDS2018 dataset is unbalanced: *Benign* makes up 87.9% of all data while only 5.33% of all samples belong to the second most represented class *DDos attacksLOICHTTP*. *SQL Injection* has the lowest number of samples, with a share of only 0.0007%. Figure 14 shows the class distribution of the initial class distribution and the modification through resampling methods. After the application of undersampling, the size of *Benign* is reduced by 93.76%, leaving it at 34.76% regarding the entire dataset. Oversampling affects all classes except *Benign*, *DDoS attacksLOICHTTP* and *DDOS attackHOIC*. The number of samples in the training data belonging to *SQL Injection* is synthetically increased from 46 to 1902. For binary classification of the CSE-CIC-IDS2018 dataset, the *Benign* class is undersampled to achieve equal number of samples for both classes. In the training data, 201740 samples each belong to *Benign* and *Attack*.

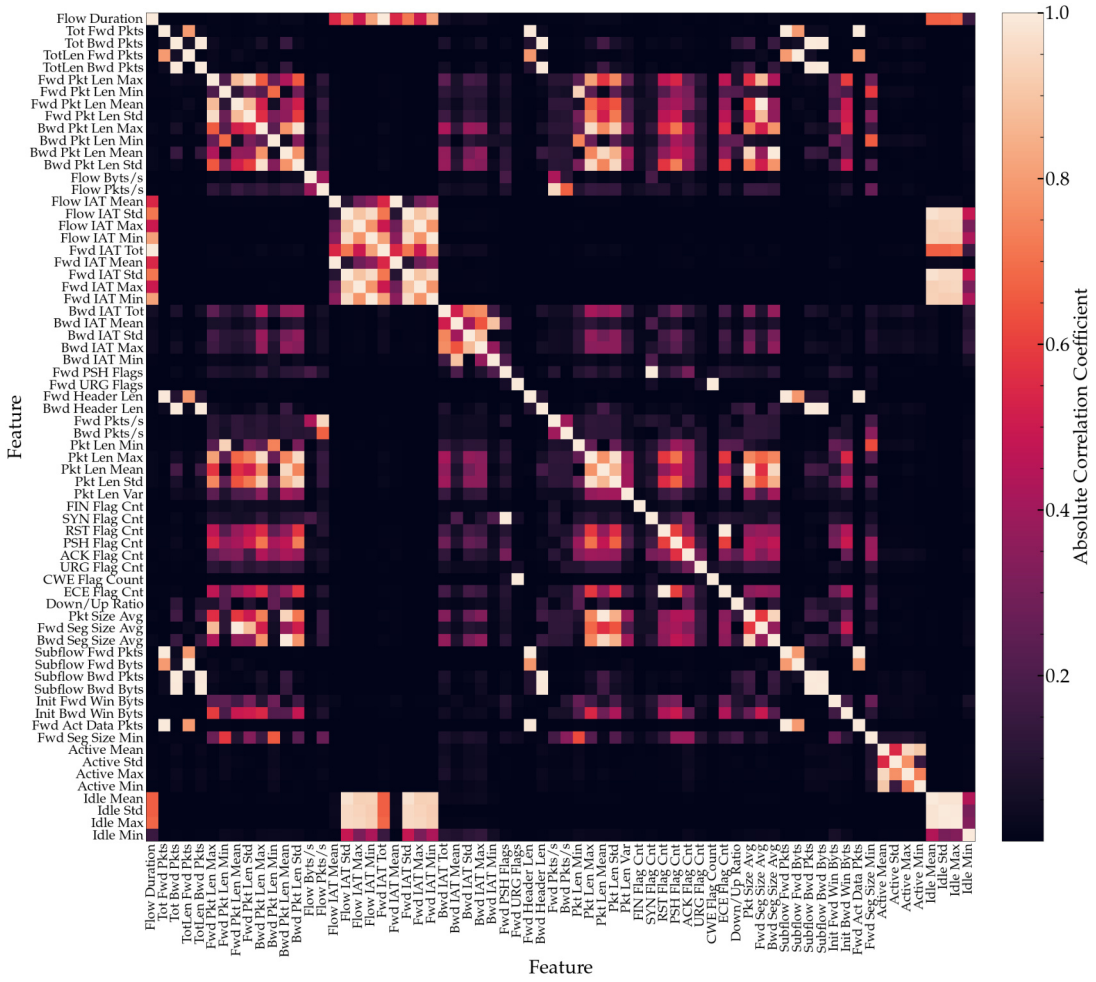
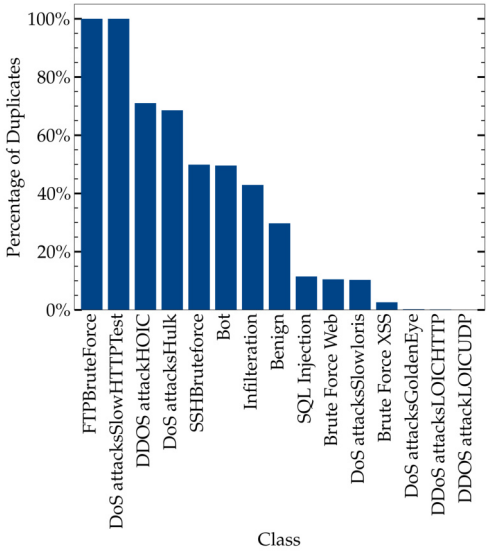


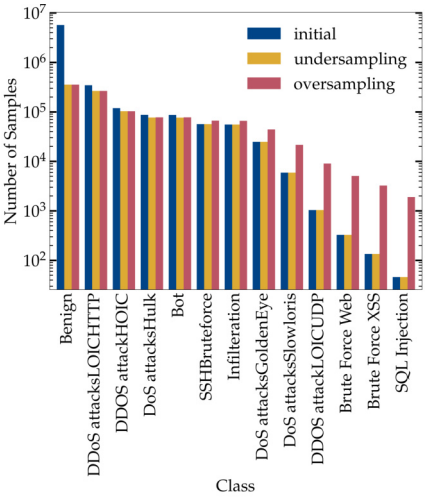
Figure 12: Correlation Matrix of the CSE-CIC-IDS2018 dataset



Class	Duplicates	
FTPBruteForce	193320	99.98%
DoS attacksSlowHTTPTest	139847	99.97%
DDOS attackHOIC	487152	71.01%
DoS attacksHulk	316713	68.57%
SSHBruteforce	93548	49.87%
Bot	141874	49.57%
Infilteration	69462	42.90%
Benign	4010325	29.74%
SQL Injection	10	11.49%
Brute Force Web	64	10.47%
DoS attacksSlowloris	1134	10.32%
Brute Force XSS	6	2.61%
DoS attacksGoldenEye	102	0.25%
DDoS attacksLOICHTTP	827	0.14%
DDOS attackLOICUDP	0	0.00%
Total	5454384	33.60%

Figure 13: Duplicates in Classes of the CSE-CIC-IDS2018 Dataset

Feature Selection Table 8 shows the mean macro-averaged F1 score across the five cross validation splits for each tested feature selection technique. The differences in the achieved F1



Class	Initial		Undersampling		Oversampling	
Benign	5684629	87.90%	354977	34.76%	354977	32.37%
DDoS attack-sLOICHTTP	345218	5.34%	265458	25.99%	265458	24.21%
DDOS attack-HOIC	119316	1.85%	103110	10.10%	103110	9.40%
DoS attack-sHulk	87119	1.35%	77008	7.54%	77783	7.09%
Bot	86590	1.34%	76571	7.50%	77562	7.07%
SSHBruteforce	56425	0.87%	56425	5.52%	66581	6.07%
Infilteration	55483	0.86%	55483	5.43%	66023	6.02%
DoS attacks-GoldenEye	24844	0.38%	24844	2.43%	44180	4.03%
DoS attacksS-lowloris	5914	0.09%	5914	0.58%	21556	1.97%
DDOS attack-LOICUDP	1038	0.02%	1038	0.10%	9031	0.82%
Brute Force Web	328	0.01%	328	0.03%	5077	0.46%
Brute Force XSS	135	0.00%	135	0.01%	3257	0.3%
SQL Injection	46	0.00%	46	0.00%	1902	0.17%

Figure 14: Multiclass Class Distribution of the CSE-CIC-IDS2018 Training Data after Under- and Oversampling

scores for each selection method are similar to the CIC-IDS2017 dataset, but especially *Information Gain-based (0.5)* and *DT-RFE-based* show great performances even with a low number of features (13 and 12, respectively), which is similar to *Correlation-based (0.5)* with 20 features. Regardless, we apply *Correlation-based (0.5)* feature selection for the final models, since it maintains better classification results for the AE architecture. Furthermore, using a single feature selection technique for all datasets allows for a better comparison.

Table 8: Mean Macro F1 of Cross Validation Splits using Feature Selection Methods on CSE-CIC-IDS2018 Training Data

		Multiclass			Binary		
Method	**#Features**	**DNN**	**CNN**	**AE**	**DNN**	**CNN**	**AE**
Baseline (Full Feature Set)	64	0.7531	0.7682	0.7624	0.9656	0.9653	0.9637
Correlation-based (0.9)	34	0.7476	0.7613	0.7474	0.9652	0.9647	0.9623
Information Gain-based (0.3)	33	0.7372	0.7606	0.7395	0.9644	0.9637	0.9616
Correlation-based (0.7)	29	0.7387	0.7643	0.7445	0.9651	0.9646	0.9634
Correlation-based (0.5)	23	0.7374	0.7571	0.7318	0.9648	0.9641	0.9608
Information Gain-based (0.4)	20	0.7397	0.7538	0.7191	0.9646	0.9641	0.9547
Correlation-based (0.3)	17	0.6606	0.68	0.5907	0.9484	0.9385	0.9507
Information Gain-based (0.5)	13	0.7379	0.7599	0.7057	0.9621	0.9608	0.9474
DT-RFE-based	12	0.7034	0.7469	0.7068	0.9598	0.9564	0.945
Correlation-based (0.1)	11	0.6388	0.6764	0.5416	0.9407	0.9324	0.9347

Feature Encoding The CSE-CIC-IDS2018 dataset contains the same features as the CIC-IDS2017 dataset. Therefore, only one feature contains categorical data (*Protocol*). The possible values *6*, *17* and *0* represent TCP, UDP and any other packet. This feature most likely has the same issues as *Protocol* of CIC-IDS2017, see Paragraph *Feature Encoding* of Section 5.1.2 for more details. For the normalization of numeric features, all model architectures achieve better results with min-max normalization.

5.2 Classification Results

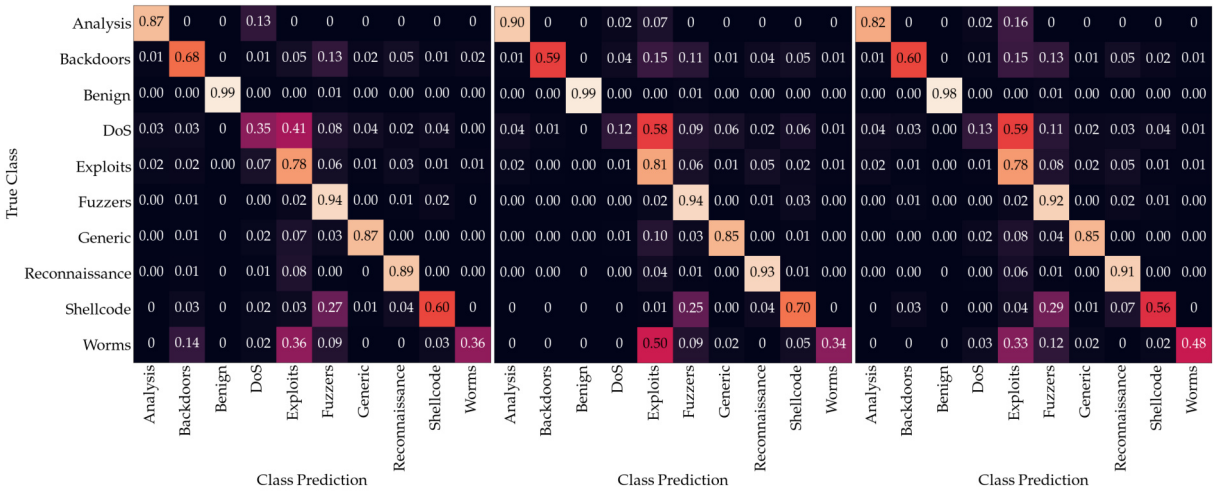
After finding the best model regarding hyperparameters and feature selection for each model architecture and dataset, we retrain the models using the full training data. The final hyperparameters that achieved the best results in the cross-validation process can be seen in Tables A.1, A.2 and A.3. We use *Correlation-based* feature selection with a threshold of 0.5 for all models because it achieves high classification performance while significantly decreasing the number of features. For more details on the results of different feature subsets, see Section 5.1. Until this stage, the test data is not utilized in any manner to ensure that no data is leaked, and the outcomes demonstrate the performance of each model for unknown data. We test the ability of each classifier to discriminate between the classes using the precision, recall and F1 scores of predictions on the test data. Scores that summarize all classes are given as the macro average, which is the mean of the scores of all classes without weighting the number of samples in each class. Since the classes are imbalanced and the majority of samples belong to the benign class, weighted metrics would give an unrealistic assessment of the abilities of the classifier (i.e., the weighted F1 for all our classifiers is above 0.97). Additionally, we investigate the confusion matrices for further insides on which classes are difficult to distinguish.

Since these results include no information about the decision-making process of the classifier models, we employ explainable artificial intelligence (XAI) methods. In particular, we use SHAP [65] to compute importance values for each feature. SHAP utilizes equations from cooperative game theory to compute values which explain the decision-making of a model. A higher SHAP value indicates a higher relevance of that feature for the classification of a particular class.

The next three sections show the achieved classification results using the described evaluation methods for the UNSW-NB15, CIC-IDS2017 and CSE-CIC-IDS2018 datasets.

5.2.1 Classification Results of UNSW-NB15

Figure 15 and Table 9 show the classification results of the DNN, CNN and AE architectures on the UNSW-NB15 dataset. All architectures can successfully distinguish benign from attack traffic, with F1 scores of above 0.99 for the *Benign* class. Regarding different attack classes, all models achieve F1 scores of above 0.7 for the classes *Exploits*, *Generic*, and *Reconnaissance*. The class *Analysis* has a high recall of between 0.82 and 0.9, but it lacks precision, since samples from other classes are mistakenly assigned to it. The same is true for Backdoors, which has a recall of around 0.65 but a precision of only 0.17 and 0.15 for DNN and AE, respectively. All architectures fail to confidently recognize samples of the class *Worms*, which is heavily under-represented in the original dataset, making up only 0.01%. Even after under- and over-sampling, the samples do only make up 1.09% of the training data, which makes it difficult to learn the specific attack characteristics. In more detail, samples of *Worms* are often mistaken to be of class *Exploits*. Since this particular misclassification can be observed for all architectures, it is likely that the classes share common characteristics, which make them challenging to distinguish. The same effect can be seen for *DoS*, which is often misclassified as *Exploits* and *Shellcode*, which is mistaken as *Fuzzers*. This observation is consistent with the critique of a large class overlap in the UNSW-NB15 dataset. The combination of entirely different attacks in one broad category increases the difficulty of proper classification. Generally, all three architectures achieve similar results, especially regarding the described misclassifications, but they show minor differences in their classification performances for different classes. Finally, this leads to macro-averaged F1 scores for the DNN and CNN architectures of 0.57 and 0.56, respectively. The AE architecture achieves a slightly worse F1 score of 0.53.



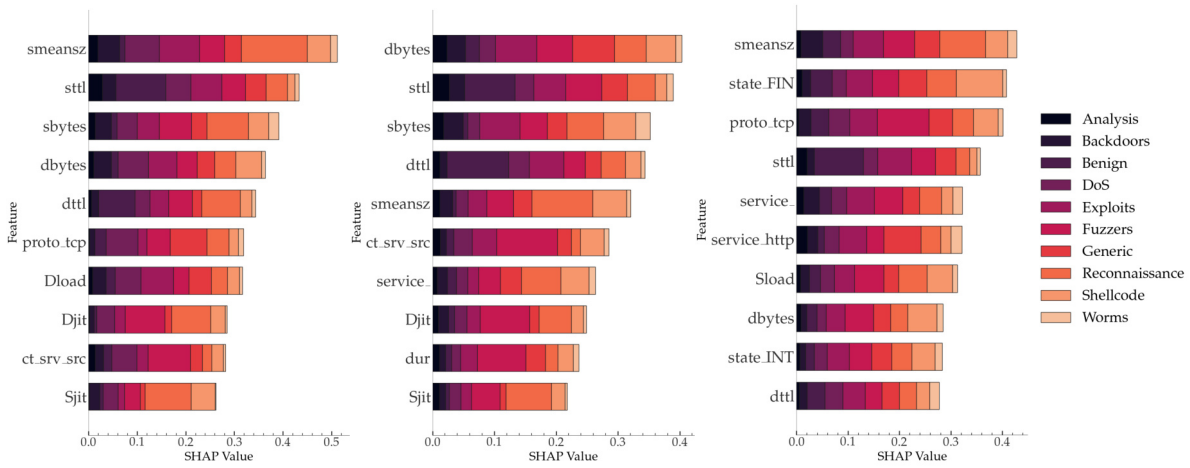
(a) Confusion Matrix DNN (b) Confusion Matrix CNN (c) Confusion Matrix AE

Figure 15: Confusion Matrix of DNN, CNN and AE Models on UNSW-NB15 Test Data

Table 9: Results of Multiclass Classifiers on UNSW-NB15 Test Data

	DNN			CNN			AE		
Class	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Analysis	0.10	0.87	0.17	0.10	0.90	0.17	0.09	0.82	0.16
Backdoors	0.17	0.68	0.28	0.47	0.59	0.52	0.15	0.60	0.24
Benign	1.00	0.99	0.99	1.00	0.99	0.99	1.00	0.98	0.99
DoS	0.32	0.35	0.33	0.49	0.12	0.19	0.24	0.13	0.17
Exploits	0.79	0.78	0.78	0.77	0.81	0.79	0.72	0.78	0.75
Fuzzers	0.41	0.94	0.57	0.41	0.94	0.57	0.39	0.92	0.55
Generic	0.97	0.87	0.91	0.96	0.85	0.90	0.93	0.85	0.89
Reconnaissance	0.84	0.89	0.87	0.80	0.93	0.86	0.72	0.91	0.81
Shellcode	0.41	0.60	0.49	0.31	0.70	0.43	0.42	0.56	0.48
Worms	0.20	0.36	0.26	0.14	0.34	0.20	0.19	0.48	0.28
macro-averaged	0.52	0.73	0.57	0.54	0.72	0.56	0.48	0.70	0.53

We further investigate our trained DNN, CNN and AE classifiers by applying SHAP. Figure 16 shows the importance of each feature for classification. The most important feature of the DNN according to SHAP is *smeansz* followed by *sttl*. The first, *smeansz*, describes the mean packet size transmitted from the source to the destination. High *smeansz* values are mostly related to the classes *Exploits* and *Shellcode*. Low values, on the other hand, are an indication for *DoS*, *Fuzzers* and *Reconnaissance*. Samples of the class *Analysis* have almost entirely a *smeansz* in the midrange. The second most important value is the Time-To-Live (TTL) value of packets transmitted by the source (*sttl*). The TTL value of a packet indicates its hop limit and is decreased while traversing a router. Therefore, a specific TTL value is not a typical characteristic of an attack but rather dependent on the operating system of the host machine and the network infrastructure [106]. Regarding the SHAP analysis of the UNSW-NB15 dataset, this feature has the highest importance for identifying the class *Benign*. More particular, a low *sttl* value is related to *Benign* and *Analysis*, while high *sttl* values are a strong indication for *Fuzzers*, *Generic*, *Reconnaissance* and *Worms*. The authors of the UNSW-NB15 dataset themselves list the usage of TTL values that can be directly related to benign or attack traffic as a critique to the KDDCUP'99 dataset [70], but their dataset seems to have the same flaw. For more details on the SHAP results regarding different classes, see Figure A.1.



(a) SHAP Results DNN (b) SHAP Results CNN (c) SHAP Results AE

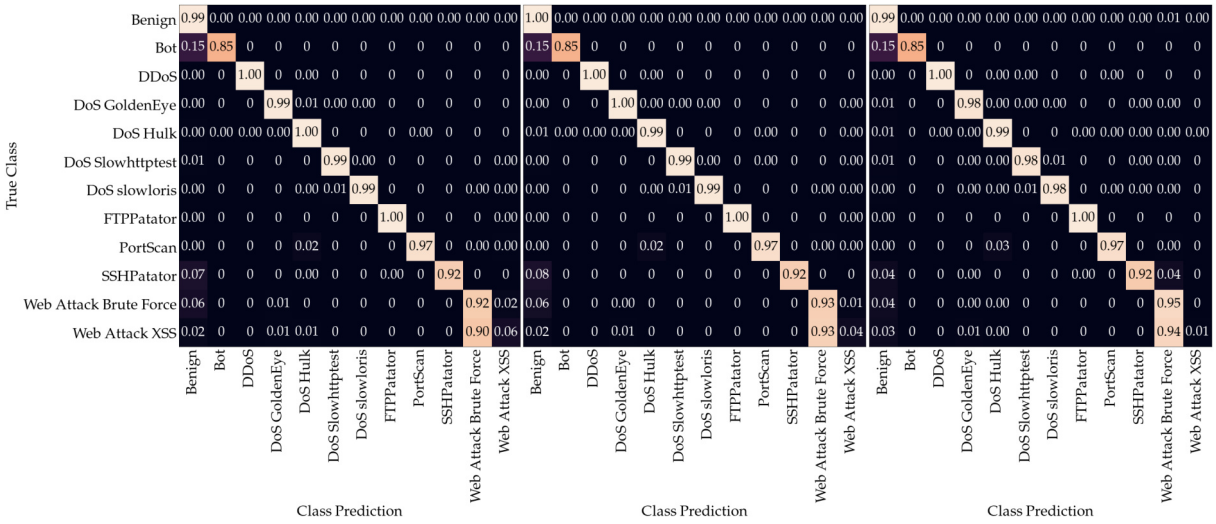
Figure 16: SHAP Feature Importance of DNN, CNN and AE Models on UNSW-NB15

5.2.2 Classification Results of CIC-IDS2017

The classifiers of the CIC-IDS2017 dataset achieve almost perfect classification results, which can be seen by the diagonal in the confusion matrices (see Figure 17). The benign class, similar to the UNSW-NB15 dataset, is successfully distinguished from attack samples by all three architectures with F1 scores of nearly 1.0 for class *Benign* (see Table 10). The same classification performance can be observed for all *DoS* classes and *FTPPatator*, which have an F1 score of above 0.9. The classifiers show a small deficit in discriminating *Bot* from *Benign*. It is noticeable, that exactly 15% of the samples of *Bot* are misclassified by all architectures, which indicates that these samples cannot be distinguished from *Benign* samples by any possible classifier. The same observation, but with almost all samples of that class, can be made for *Web Attack XSS*, which is mistaken as *Web Attack Brute Force*. They seem to be no characteristics that differ between these two classes. Therefore, all samples are classified as the class with more samples (*Web Attack Brute Force*), to minimize the loss. The classification results of the model architectures do not differ much, only the AE architecture achieves a considerably lower macro-averaged F1 score of 0.76 in comparison to the DNN and CNN architectures with 0.83 and 0.84, respectively.

Table 10: Results of Multiclass Classifiers on CIC-IDS2017 Test Data

	DNN			**CNN**			**AE**		
Class	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Benign	1.00	0.99	1.00	1.00	1.00	1.00	1.00	0.99	0.99
Bot	0.41	0.85	0.55	0.40	0.85	0.55	0.38	0.85	0.53
DDoS	0.99	1.00	1.00	0.99	1.00	1.00	0.99	1.00	0.99
DoS GoldenEye	0.90	0.99	0.94	0.90	1.00	0.95	0.87	0.98	0.92
DoS Hulk	0.96	1.00	0.98	0.99	0.99	0.99	0.97	0.99	0.98
DoS Slowhttptest	0.82	0.99	0.90	0.81	0.99	0.89	0.82	0.98	0.90
DoS slowloris	0.93	0.99	0.96	0.96	0.99	0.98	0.94	0.98	0.96
FTPPatator	0.99	1.00	1.00	0.99	1.00	0.99	0.96	1.00	0.98
PortScan	0.88	0.97	0.92	0.91	0.97	0.94	0.70	0.97	0.81
SSHPatator	0.86	0.92	0.89	0.93	0.92	0.92	0.72	0.92	0.80
Web Attack Brute Force	0.63	0.92	0.75	0.66	0.93	0.77	0.12	0.95	0.21
Web Attack XSS	0.06	0.06	0.06	0.09	0.04	0.06	0.05	0.01	0.01
macro-averaged	0.79	0.89	0.83	0.80	0.89	0.84	0.71	0.88	0.76



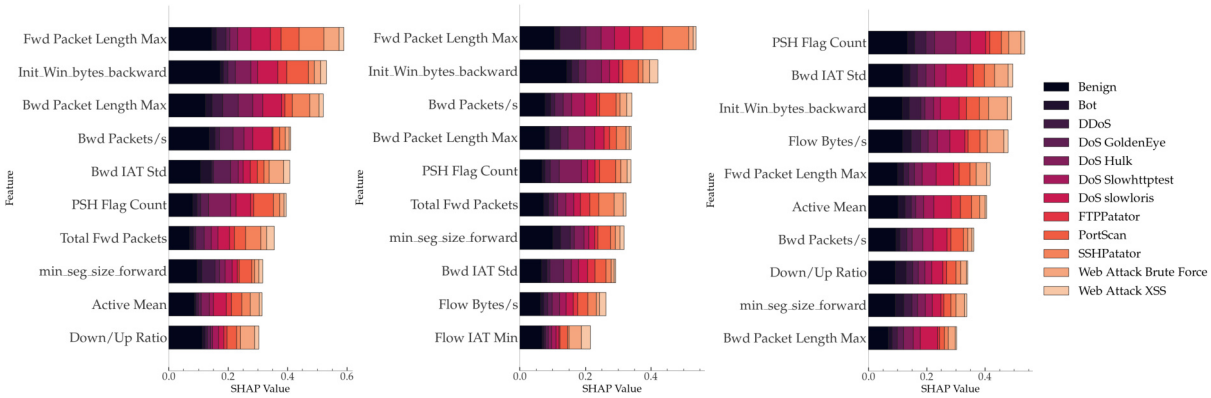
(a) Confusion Matrix DNN (b) Confusion Matrix CNN (c) Confusion Matrix AE

Figure 17: Confusion Matrix of DNN, CNN and AE Models on CIC-IDS2017 Test Data

We further evaluate our trained classifiers by applying SHAP, which computes importance values for each feature. Figure 18 shows the resulting SHAP values for each architecture. The features are ordered by their importance, summed up over all classes, beginning with the most important feature according to the SHAP analysis. The importance of features for both the DNN and CNN architecture is similar, and the order of features exhibits only minor differences. The feature with the highest SHAP value is *Fwd Packet Length Max*, which represents the maximum length of a packet in a flow, sent from the source to the destination. A high value in this feature often leads to predictions of *DoS Slowhttptest*, *Web Attack Brute Force*, or *SSHPatator*, while a low packet length is related to the classes *DDoS*, *DoS slowloris*, *FTPPatator*, and *PortScan*. For more details on the feature importance for each class, and especially if it is related to a high or low value, see Figure A.2. The second most important feature is *Init_Win_bytes_backward*, which measures the number of bytes in the initial window, sent in the backward direction. The AE model, however, has slightly different feature importances, with *PSH Flag Count* as the most important feature, which is the number of packets in a flow with the PSH flag. In the CIC-IDS2017 dataset, this feature is either 1 or 0 for all samples. A *PSH Flag Count* of 1 is mostly related to either *DoS GoldenEye*, *PortScan*, *SSHPatator*, or one of the two web attacks. If the flag count is 0, it is, according to the SHAP analysis, a strong indicator for *DoS Hulk* (see Figure A.3, which shows the SHAP values of the AE model for each class).

5.2.3 Classification Results of CSE-CIC-IDS2018

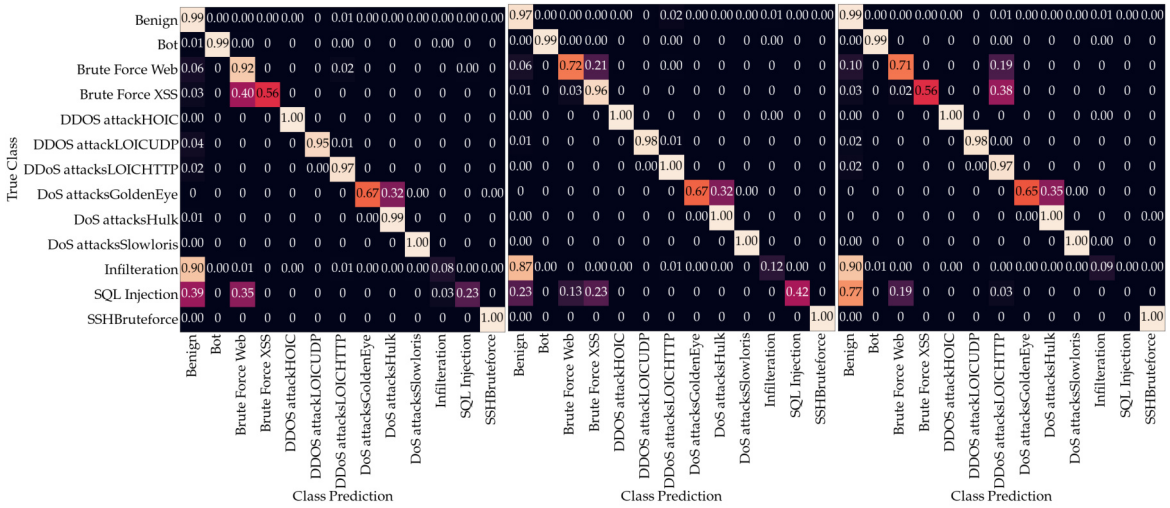
The classification results of the DNN, CNN, and AE architectures on the CSE-CIC-IDS2018 dataset can be seen in Table 11 with the achieved precision, recall and F1 scores, and Figure 19, which shows the confusion matrices of each architecture. All models can discriminate benign samples from attack samples with a high success rate, and have F1 scores for the *Benign* class of above 0.98. Regarding the classes *Brute Force Web* and *Brute Force XSS*, the classifiers show difficulties distinguishing samples between these classes, similar to *Web Attack Brute Force* and *Web Attack XSS* of the CIC-IDS2017 dataset. Both the DNN and CNN models misclassify a large proportion of the samples from one of these classes as the other class. Interestingly, the AE-based model does not misclassify them with each other, but with *DDoS attacksLOICHTTP*. Other classes, which seem to be difficult to classify, are *SQL Injection*, *Infiltration*, and *DoS at-*



(a) SHAP Results DNN (b) SHAP Results CNN (c) SHAP Results AE

Figure 18: SHAP Feature Importance of DNN, CNN and AE Models on CIC-IDS2017

tacksGoldenEye. The class *SQL Injection* suffers from a very low number of samples: only 46 samples from the original training data belong to this class, which appears to be insufficient for learning the characteristics of this attack. Regarding *Infiltration*, the number of samples should be high enough, but almost all (about 90%) samples are predicted to be *Benign*. Additionally, around 35% of *DoS attacksGoldenEye* samples are misclassified as *DoS attacksHulk*. All other classes achieve great results, with F1 scores around 0.9. In total, the DNN, CNN, and AE architectures reach macro-averaged F1 scores of 0.7, 0.68, and 0.68, respectively.



(a) Confusion Matrix DNN (b) Confusion Matrix CNN (c) Confusion Matrix AE

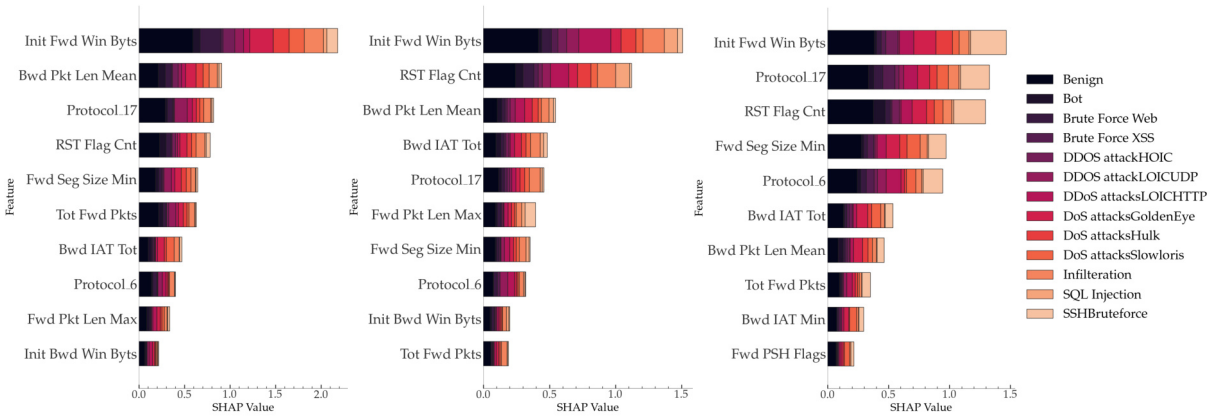
Figure 19: Confusion Matrix of DNN, CNN and AE Models on CSE-CIC-IDS2018 Test Data

For further analysis of the decision-making of the classifiers, we employ XAI techniques in the form of SHAP. Figure 20 shows the resulting feature importances for the three model architectures. The most important feature for all architectures is *Init Fwd Win Byts*, which represents the number of bytes in the initial window, sent from the source to the destination. This feature cannot be directly related to classes by discriminating between low and high values, i.e., while *DDOS attackHOIC* is mostly related to high values of this feature, *DoS attacksSlowloris* and *SSH-Bruteforce* tend to be predicted for neither high nor low but midrange values. Other features with a high importance are *Bwd Pkt Len Mean* (mean size of packets in backward direction) and *RST Flag Cnt* (number of packets with RST flag). A high *Bwd Pkt Len Mean* is mostly related to

Table 11: Results of Multiclass Classifiers on CSE-CIC-IDS2018 Test Data

	DNN			CNN			AE		
Class	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Benign	0.99	0.99	0.99	0.99	0.97	0.98	0.99	0.99	0.99
Bot	0.99	0.99	0.99	1.00	0.99	0.99	0.96	0.99	0.98
Brute Force Web	0.04	0.92	0.08	0.07	0.72	0.13	0.04	0.71	0.08
Brute Force XSS	0.82	0.56	0.67	0.09	0.96	0.17	0.38	0.56	0.45
DDOS attackHOIC	0.99	1.00	0.99	0.95	1.00	0.98	0.99	1.00	0.99
DDOS attackLOICUDP	0.73	0.95	0.83	0.71	0.98	0.83	0.72	0.98	0.83
DDoS attacksLOICHTTP	0.91	0.97	0.94	0.73	1.00	0.85	0.92	0.97	0.94
DoS attacksGoldenEye	0.96	0.67	0.79	0.97	0.67	0.80	0.88	0.65	0.75
DoS attacksHulk	0.90	0.99	0.94	0.90	1.00	0.95	0.89	1.00	0.94
DoS attacksSlowloris	0.67	1.00	0.80	0.72	1.00	0.84	0.73	1.00	0.84
Infilteration	0.14	0.08	0.10	0.14	0.12	0.13	0.13	0.09	0.11
SQL Injection	0.00	0.23	0.01	0.10	0.42	0.16	0.00	0.00	0.00
SSHBruteforce	0.98	1.00	0.99	1.00	1.00	1.00	0.98	1.00	0.99
macro-averaged	0.70	0.80	0.70	0.65	0.83	0.68	0.66	0.76	0.68

Brute Force Web, *Brute Force XSS* and both DDoS LOIC attacks, while a low value in this feature is an indication for either *Bot*, *DDOS attackHOIC* or *SSHBruteforce*. The *RST Flag Cnt* feature has a particularly high importance for both CNN and AE. It is only binary and shows the highest relevance of the CNN model for the prediction of *Bot*, *Brute Force Web* and *SQL Injection*. For more details on the SHAP values for each class, see Figure A.4.



(a) SHAP Results DNN (b) SHAP Results CNN (c) SHAP Results AE

Figure 20: SHAP Feature Importance of DNN, CNN and AE Models on CSE-CIC-IDS2018

5.2.4 Classification Results of Binary Classifiers

In this section, we present the results of our binary classifiers using the test data. Table 12 shows the F1 scores of the three model architectures for each dataset, when only discriminating between *Attack* and *Benign*.

We observe that all classifiers perform well and can discriminate between attack and benign traffic. The models achieve macro-averaged F1 scores of around 0.92, 0.99 and 0.97 for the UNSW-NB15, CIC-IDS2017 and CSE-CIC-IDS2018 dataset, respectively. There are no major differences between the model architectures. For all datasets, the F1 score of the class *Attack* is slightly lower than for *Benign*. In the case of the UNSW-NB15 and CIC-IDS2017 datasets, this is due to benign samples being falsely classified as *Attack*, which is shown by a lower precision

Table 12: F1 Scores of Binary Classifiers

	UNSW-NB15			CIC-IDS2017			CSE-CIC-IDS2018		
Class	**DNN**	**CNN**	**AE**	**DNN**	**CNN**	**AE**	**DNN**	**CNN**	**AE**
Attack	0.85	0.85	0.85	0.99	0.99	0.98	0.96	0.96	0.95
Benign	0.99	0.99	0.99	1.00	1.00	1.00	0.99	0.99	0.99
macro-averaged	0.92	0.92	0.92	0.99	0.99	0.99	0.98	0.97	0.97

of the *Attack* class. However, for the CSE-CIC-IDS2018 dataset, the number of attack samples that are falsely classified as *Benign* is higher, resulting in a lower recall.

5.3 Adversarial Attack Results

We test all our attack approaches using test data, which was not used for training of the victim models. We only consider samples that are correctly classified to obtain the rate of changing the prediction of these samples to *Benign*, which is the attack success rate (ASR). First, we present the success rate of white-box attacks with subsets based on domain constraints and single feature perturbation. Then, we follow the same approach to present the transferability of adversarial examples generated by white-box attacks using substitute models, which are trained with parts of the original training data. Finally, we evaluate the vulnerability of the trained classifiers to black-box attacks. Additionally, we shortly present adversarial attacks against binary classifiers, and further analyze successful adversarial examples.

5.3.1 White-Box Attack Results

The next two paragraphs describe the results from our white-box adversarial attacks using a) feature subsets according to domain constraints and b) perturbations of only a single feature. The used attacks are BIM, PGD, DeepFool and the C&W attack with two implementations, one from the *Adversarial Robustness Toolbox* (*ART*) and the other using *Foolbox* (*FB*). Due to computational limitations, we limit the number of samples of each class that are used to generate adversarial examples to 10000.

Attacks using Domain Constraints We test the vulnerability of the different datasets and classifier architectures in accordance with domain constraints using two subset (time- and source-based) created for features of the CICFlowMeter, which is used by both the CIC-IDS2017 and CSE-CIC-IDS2018 dataset. Regarding the UNSW-NB15 dataset, we use a subset which is based on the restrictions formulated by the source-based subset (see Section 4.4.5).

Figure 21 shows the resulting attack success rates (ASR) for the different adversarial attack methods. The macro-averaged ASR is the highest for the CIC-IDS2017 and CSE-CIC-IDS2018 datasets with around 70%, while the ASR for the UNSW-NB15 dataset is slightly under 50% and shows that it is more robust to attacks. Both BIM and PGD are the most effective for UNSW-NB15, but also perform well for the other datasets. DeepFool and the C&W attack, on the other hand, perform significantly worse for the UNSW-NB15 dataset and slightly worse for CIC-IDS2017 and CSE-CIC-IDS2018. The *ART* implementation of C&W achieves the lowest ASR for all datasets. We cannot observe noteworthy differences regarding model architectures, and there are only small differences for the two feature subsets. However, the source-based set seems to achieve higher ASRs than the time-based variant of the CSE-CIC-IDS2018 dataset, for almost all attack methods and models.

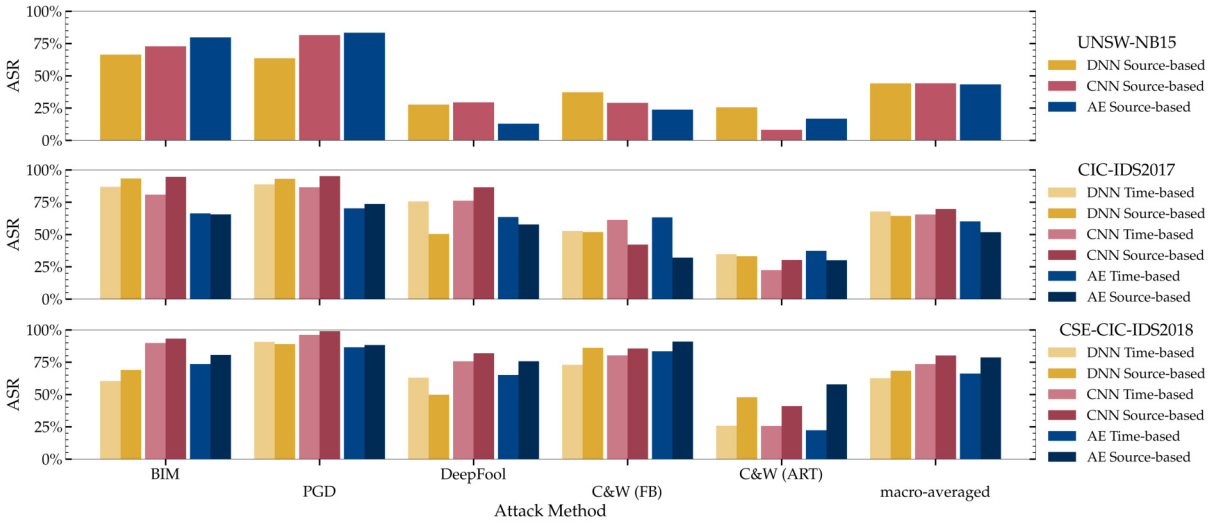


Figure 21: Attack Success Rate of White-Box Attacks with Domain Constraints

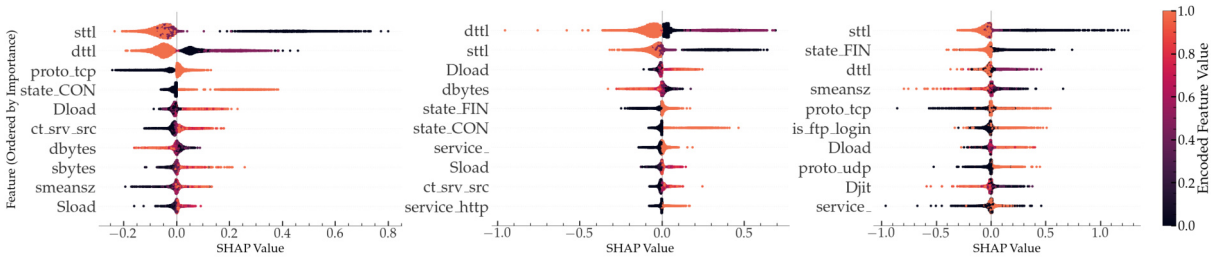
Single Feature Attacks on UNSW-NB15 For further analysis, we apply the same attacks as before but with the restriction that only the one most effective feature can be changed, and all other features of a sample stay the same. The most effective feature is the feature that achieves the highest ASR and is individually selected for each class. Therefore, we analyze the resulting ASR in more detail and evaluate differences between classes.

Table 13 shows the selected feature for each class of the UNSW-NB15 dataset, which achieved the highest ASR. The best feature is evaluated individually for each class, model architecture and attack method, but since this data is quite extensive, we summarize the results by only presenting the features that are selected most often by the different attack methods. For the complete results, see Figures A.4, A.5 and A.6. We observe that the manipulation of either *sttl* or *dttl* is most effective to change the prediction from any attack class to *Benign*. Some classes (e.g., *Backdoors*) seem more related to *dttl*, and other classes (e.g., *Shellcode*) to *sttl*, but most classes can be changed using either of them. Only for the AE-based model and the classes *Generic* and *Worms*, other features (*smeansz* and *Sload*, respectively) are selected. For further investigation, we compare the selected features with the feature importances of the class *Benign* according to SHAP. Figure 22 shows the SHAP values for the features by their order of importance. Each dot represents a sample, and its color the normalized value of the corresponding feature. We can see that the features *sttl* and *dttl* have the highest importance for all model architectures for the classification of benign samples. Furthermore, we can see that mostly low or midrange TTL values lead to a *Benign* prediction, while high values in these features indicate attack classes. As already mentioned in Section 5.2.1, TLL values are not a common indicator for attacks, but rather depend on the operating system or network infrastructure. Since the TTL values of packets coming from the attacker (*sttl*) can be changed, single feature attacks using this feature would be most likely possible, even in a real-world setting.

Figure 23 depicts the ASR for each attack method using the individually selected feature. All models of the UNSW-NB15 dataset are generally vulnerable to single feature white-box attacks with an ASR of around 40%, which is slightly lower compared to the feature subsets according to domain constraints, but still considerably high. There are a few observations we can make about differences between the different algorithms, models, and classes. Considering the macro-averaged ASR across classes, BIM, PGD and DeepFool perform the best, and along them DeepFool with 44%, 39% and 39%. The C&W (FB) attack achieves slightly lower ASRs, but the results of C&W (ART) are significantly worse with only 26%, 11% and 13% for the

Table 13: Selected Features for Single Feature Perturbation of UNSW-NB15

Class	DNN	CNN	AE
Analysis	dttl	sttl	dttl
Backdoors	dttl	dttl	dttl
DoS	dttl	sttl	sttl
Exploits	dttl	dttl	dttl
Fuzzers	dttl	sttl	sttl
Generic	sttl	sttl	smeansz
Reconnaissance	dttl	dttl	dttl
Shellcode	sttl	sttl	sttl
Worms	sttl	sttl	Sload



(a) SHAP Results DNN (b) SHAP Results CNN (c) SHAP Results AE

Figure 22: SHAP Feature Importance of DNN, CNN and AE Models for Class Benign on UNSW-NB15

DNN, CNN and AE models, respectively. The results show no large differences between the model architectures, but AEs seem to be slightly more robust (ASR around 5% lower than DNN for BIM, PGD, DeepFool and C&W (FB)) and CNNs slightly more vulnerable to attacks (ASR around 5% higher than DNN for BIM, PGD and C&W (FB)). Regarding the vulnerability of distinct classes, *Generic* has the highest ASR followed by *Analysis* and *Reconnaissance*, though *Generic* with *DNN* has a low ASR for all attacks except DeepFool and C&W (ART). The classes *Shellcode* regarding AEs and *Worms* for DNN and CNN appear almost impossible to attack, and have ASRs of nearly 0%.

Single Feature Attacks on CIC-IDS2017 Table 14 shows the selected feature for each model and class of the CIC-IDS2017 dataset, which occurred most often regarding the different attack methods (see the Tables A.7, A.8 and A.9 for more details). We can observe a wider variety of features in comparison to the UNSW-NB15 dataset and more differences between model architectures. The most prominent features are *Flow Duration*, *RST Flag Count* and *Bwd Packets/s* (number of packets per second coming from the destination). When considering the SHAP values for the class *Benign* (see Figure 24), we can neither find *Flow Duration* nor *RST Flag Count*. Even though the change of only one of these features is sufficient to change the prediction to *Benign*, SHAP does not list them as important features. A possible reason for this observation is that the SHAP values depend on available samples for their explanations. Therefore, adversarial attacks can show vulnerabilities in the decision-making process of classifiers, which otherwise would be hidden. However, the feature *Bwd Packets/s* is listed under the most important features. According to the SHAP analysis of the class *Benign* regarding this feature, a high number of packets per second sent in the backward direction is an indication for benign traffic. But the manipulation of this feature in a real-world scenario might be infeasible, since it depends on the number of packets the victim sends.

The resulting ASRs using single feature perturbation can be seen in Figure 25. The attack methods achieve nearly 100% macro-averaged success rates and perform even better than the

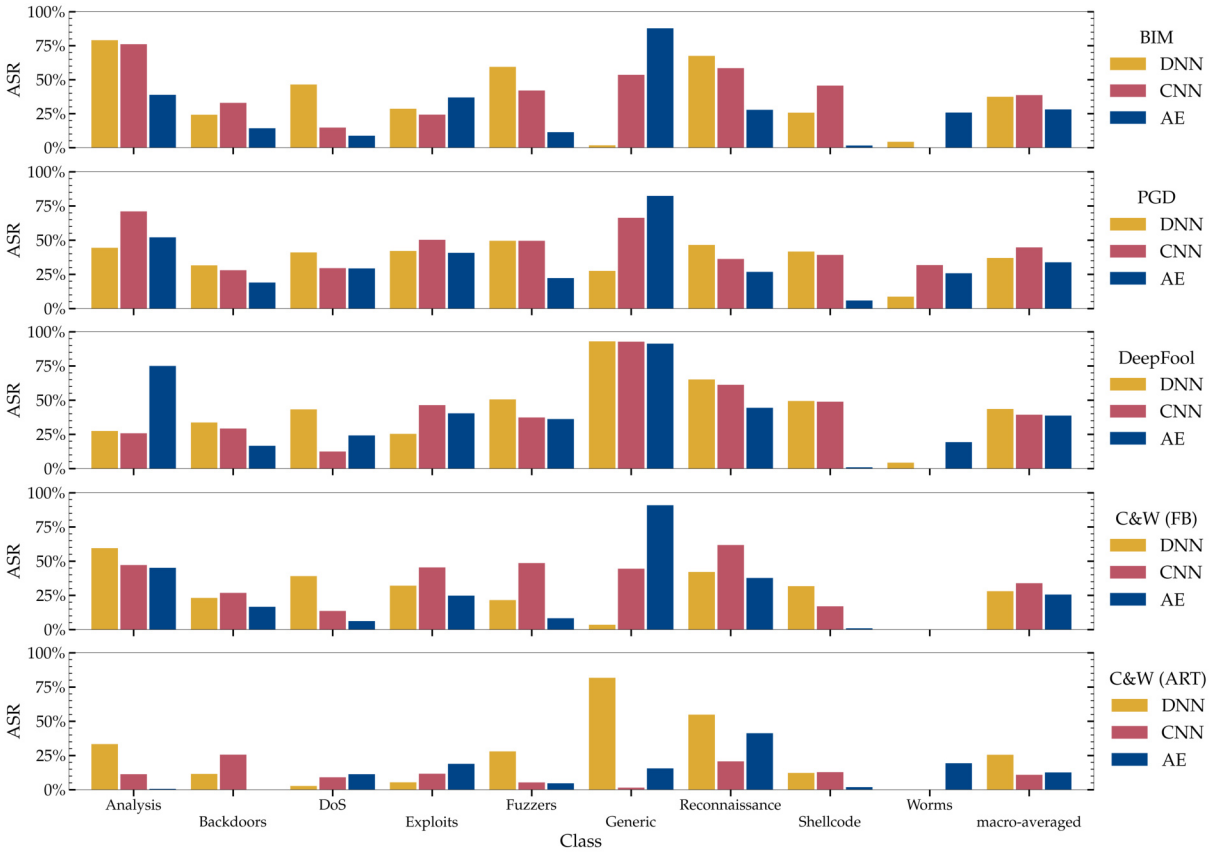
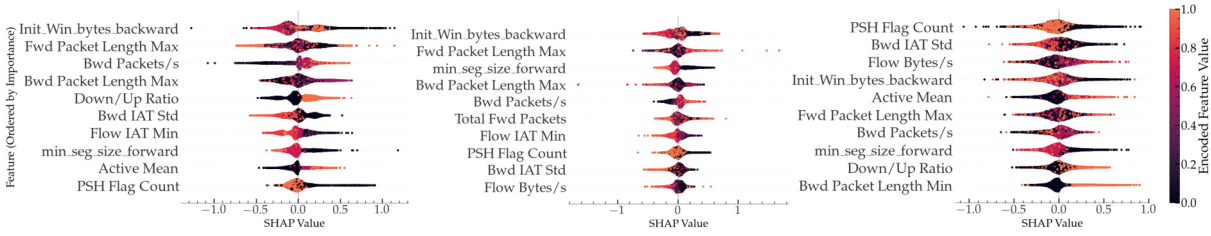


Figure 23: Attack Success Rate of White-Box Attacks with Single Feature Perturbation on UNSW-NB15

Table 14: Selected Features for Single Feature Perturbation of CIC-IDS2017

Class	DNN	CNN	AE
Bot	Flow Packets/s	Bwd Packets/s	min_seg_size_forward
DDoS	Active Std	Fwd Packet Length Min	Fwd Packet Length Max
DoS GoldenEye	Bwd Packets/s	Bwd Packets/s	Flow Duration
DoS Hulk	Fwd PSH Flags	Bwd Packet Length Max	Flow Packets/s
DoS Slowhttptest	min_seg_size_forward	Bwd Packets/s	Flow Duration
DoS slowloris	RST Flag Count	PSH Flag Count	Flow Duration
FTPPatator	RST Flag Count	Init_Win_bytes_backward	Init_Win_bytes_backward
PortScan	RST Flag Count	PSH Flag Count	Bwd Packets/s
SSHPatator	min_seg_size_forward	Flow Duration	Flow Bytes/s
Web Attack Brute Force	Flow Duration	min_seg_size_forward	Flow Duration
Web Attack XSS	Flow Duration	Flow Duration	Flow Duration



(a) SHAP Results DNN (b) SHAP Results CNN (c) SHAP Results AE

Figure 24: SHAP Feature Importance of DNN, CNN and AE Models for Class Benign on CIC-IDS2017

approach using subsets of features. This is generally surprising, but since the algorithms use the same hyperparameters, including the number of iterations, as in the other approach, they might be able to converge quicker with the much smaller search space of one feature. Another explanation could be the choice of features that are included in neither subset and are simply more effective than the other features. Features that were selected but are not included in either subset based on domain constraints are, among others, *Flow Duration*, *RST Flag Count* and *Bwd Packets/s*. Therefore, according to these subsets, most of the single feature attacks would be impossible in a real-world scenario. When further analyzing the resulting ASRs, we can observe that especially the classes *Bot*, *SSHPatator*, *Web Attack Brute Force* and *Web Attack XSS* have a high success rate of nearly 100% for all model architectures and attack methods except C&W (ART), which generally achieves lower results. The class *DDoS* seems to be the most robust, but still almost all ASRs of that class are above 50%. Regarding differences between the model architectures, DNNs seem slightly more vulnerable than the other models, followed by CNNs and then AEs. However, these are only small differences, and in general, all model architectures are vulnerable to single feature perturbations.

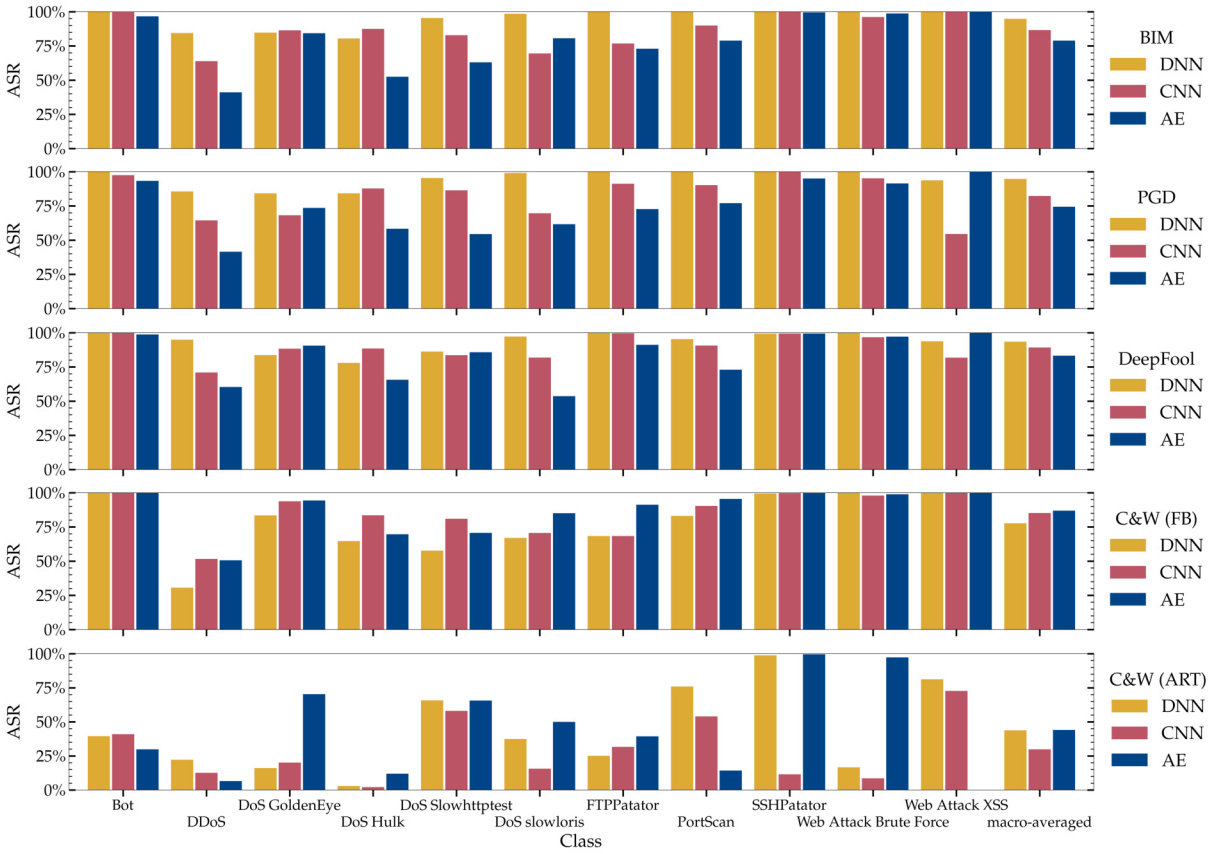


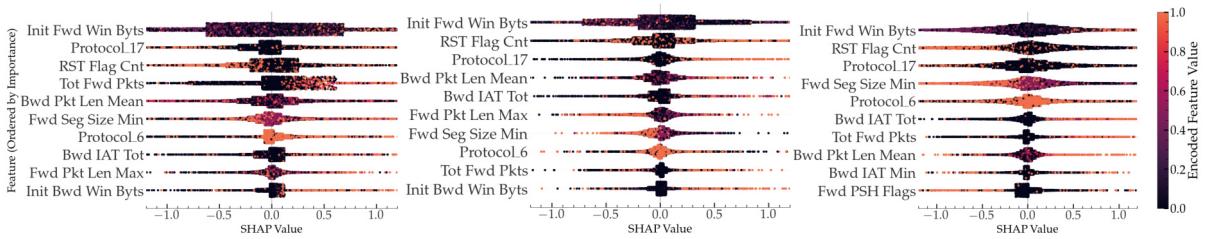
Figure 25: Attack Success Rate of White-Box Attacks with Single Feature Perturbation on CIC-IDS2017

Single Feature Attacks on CSE-CIC-IDS2018 We apply the same techniques for single feature perturbations using the CSE-CIC-IDS2018 dataset. The selected features can be seen in Table 15, for the selected features for each attack method, see Tables A.10, A.11 and A.12. Similarly to the CIC-IDS2017 dataset, the features are more diverse and there are more differences between the model architectures when comparing to the UNSW-NB15 dataset. The most prominent features are *Fwd Pkt Len Max* (maximum packet size from source), *Init Fwd*

Win Byts (number of bytes sent in the initial window) and *Fwd Seg Size Min* (minimum segment size from source). These features are also among the list of most important features for the class *Benign* according to SHAP, see Figure 26. The SHAP analysis shows that *Init Fwd Win Byts* is the most important feature for the classification as benign traffic, while the importance of *Fwd Pkt Len Max* and *Fwd Seg Size Min* varies across models. Unfortunately, we cannot derive clear feature values which are an indicator for benign traffic. According to Hashemi et al. [37], the manipulation of both *Init Fwd Win Byts* and *Fwd Pkt Len Max* is most likely possible as an adversary. Therefore, the single feature perturbations that are applied to this dataset pose a serious threat.

Table 15: Selected Features for Single Feature Perturbation of CSE-CIC-IDS2018

Class	DNN	CNN	AE
Bot	Fwd Pkt Len Min	Bwd IAT Tot	Fwd Pkt Len Max
Brute Force Web	Bwd Pkt Len Mean	Fwd Pkt Len Max	Fwd Pkt Len Min
Brute Force XSS	Tot Bwd Pkts	Init Fwd Win Byts	Fwd Pkt Len Min
DDOS attackHOIC	Fwd Pkt Len Min	Fwd Pkt Len Max	Init Fwd Win Byts
DDOS attackLOICUDP	Tot Fwd Pkts	Flow Duration	Flow Duration
DDoS attacksLOICHTTP	Fwd Pkt Len Max	Init Fwd Win Byts	Bwd Pkt Len Mean
DoS attacksGoldenEye	Init Fwd Win Byts	Init Fwd Win Byts	Fwd Seg Size Min
DoS attacksHulk	Flow Byts/s	Fwd Seg Size Min	Fwd Seg Size Min
DoS attacksSlowloris	Init Fwd Win Byts	Init Fwd Win Byts	Active Mean
Infilteration	Fwd PSH Flags	Init Fwd Win Byts	Fwd Seg Size Min
SQL Injection	Fwd Pkt Len Max	Fwd Pkt Len Max	-
SSHBruteforce	Down/Up Ratio	Flow Duration	Flow Byts/s



(a) SHAP Results DNN (b) SHAP Results CNN (c) SHAP Results AE

Figure 26: SHAP Feature Importance of DNN, CNN and AE Models for Class Benign on CSE-CIC-IDS2018

The attack results of the CSE-CIC-IDS2018 dataset are similar to the CIC-IDS2017 dataset, see Figure 27. All attack methods, except C&W (ART), achieve macro-averaged ASRs of over 75%. The highest macro-averaged ASR for the DNN model is 93% with C&W (FB), and for CNN and AE it is 88% and 91%, respectively, with DeepFool. The class *SQL Injection* is not considered for the AE in this calculation, since this model correctly classifies no samples of this class. The model architectures show no clear difference in their robustness and are similarly vulnerable. Overall, C&W (ART) achieves significantly lower ASRs, but interestingly, it has great success for *DDOS attackLOICUDP* while the other methods fail to generate successful adversarial examples for the CNN and AE architectures regarding this class. Another interesting observation can be made for the DNN model and the classes *Brute Force XSS* and *DDOS attackHOIC*, for which all methods show lower ASRs except C&W (FB) with 100% and 80%. Apart from that, most classes have an ASR of around 90-100%, independent of the attack method (except C&W (ART)) or model architecture.

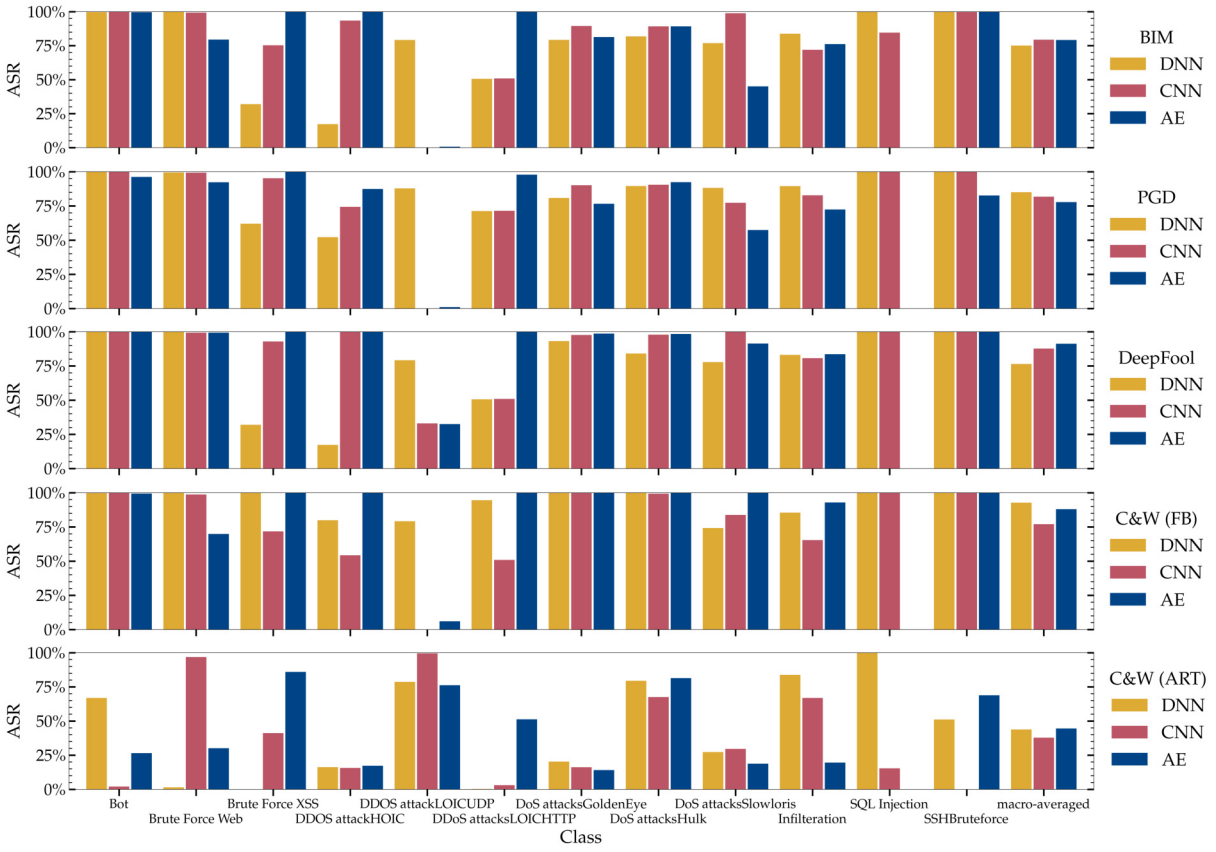


Figure 27: Attack Success Rate of White-Box Attacks with Single Feature Perturbation on CSE-CIC-IDS2018

5.3.2 Substitute Attack Results

We test the transferability of adversarial examples created by white-box attacks using substitute models, which we train with 10%, 1% and 0.1% of the original training data. After generation of the adversarial examples based on the weights of the substitute model, we test them for their attack success on the victim model.

Substitute Attacks on UNSW-NB15 Figure 28 depicts the ASR for the UNSW-NB15 dataset for single feature perturbation and the source-based subset. In general, the substitute attacks behave similar to the direct application of white-box attacks. BIM and PGD show the best results, while both implementations of the C&W attack achieve only low success rates. When comparing the single with the source-based approach, we can observe that the differences between attack methods and models are similar, but the source-based approach has an overall higher ASR. In terms of differences between model architectures, the AE is seemingly more robust to the substitute attacks than the other classifiers, with a macro-averaged ASR of around 20% for the source-based subset, while DNN and CNN reach over 40%. Regarding the different sizes of training data, there are notable differences, but we cannot infer a clear trend. The ASR appears to be dependent on the substitute model but less on the size of the training data that was used, which indicates that even less than 0.1% would be sufficient for the generation of successful adversarial examples.

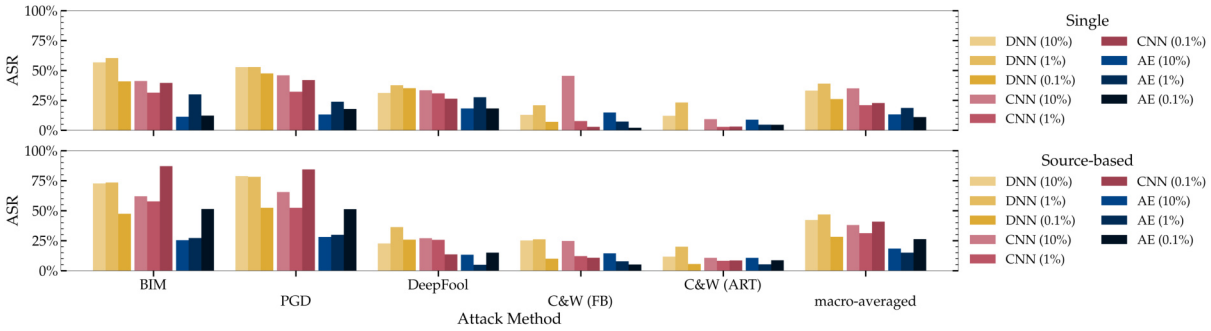


Figure 28: Attack Success Rate of Substitute Attacks with varying Percentages of Training Data on UNSW-NB15

Substitute Attacks on CIC-IDS2017 We apply the same substitute attack methodology on the CIC-IDS2017 dataset and evaluate the resulting ASRs for the three victim models, which can be seen in Figure 29. We observe high success rates, especially for the DNN and CNN models using BIM and PGD with the source-based subset, which achieve ASRs of around 90%. Similarly to the UNSW-NB15 dataset, the ASRs of the source-based approach are generally higher than for single feature perturbation. However, the time-based subset performs comparable to single feature perturbation, and has a lower success than using source-based features. For all three approaches, we observe that the AE architecture is more robust than the other models, e.g., for source-based the macro-averaged ASRs are around 30% for the AE, while being double as high for the other models with around 60%. Regarding the different training sizes, the single feature tests indicate a slight trend to higher ASRs for more training data. But when including the results of both feature subsets, we cannot confirm this observation and ASR and training size seems not directly correlated.

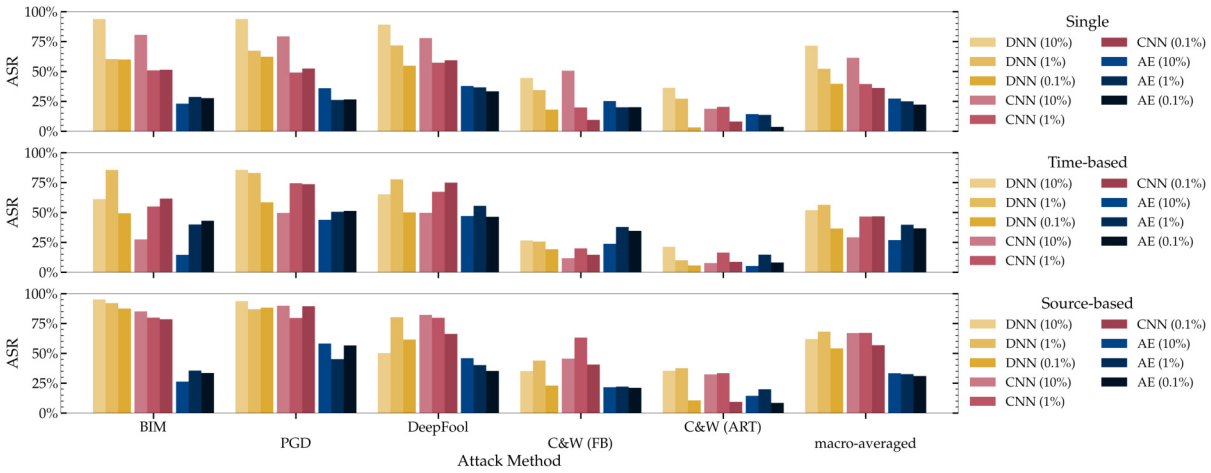


Figure 29: Attack Success Rate of Substitute Attacks with varying Percentages of Training Data on CIC-IDS2017

Substitute Attacks on CSE-CIC-IDS2018 In this paragraph, we describe the results of substitute attacks on the CSE-CIC-IDS2018 dataset. Similar to the white-box attack results, BIM, PGD, DeepFool and C&W (FB) show high ASRs of 50% or higher, while C&W (ART) only has little success in the generation of adversarial examples. Regarding the AE model, the macro-averaged ASR scores of the three approaches (single feature, time-based, and source-based

subset) are equally around 40%, while for both DNN and CNN, single feature perturbation achieves slightly lower results. Interestingly, we observe a slight trend that the adversarial examples generated using substitute models with less data perform better, which is against our hypothesis that transferability becomes less effective when only using a small proportion of data. It might be the case that these differences are only dependent on the specific substitute model, and even less than 0.1% training data would be sufficient. That explanation would be consistent with the results from the other two datasets, which also show model specific differences but no clear correlation regarding the training data size. Future work could examine such substitute attacks further, by using less training data and multiple substitute models for each training size to average out possible model specific differences.

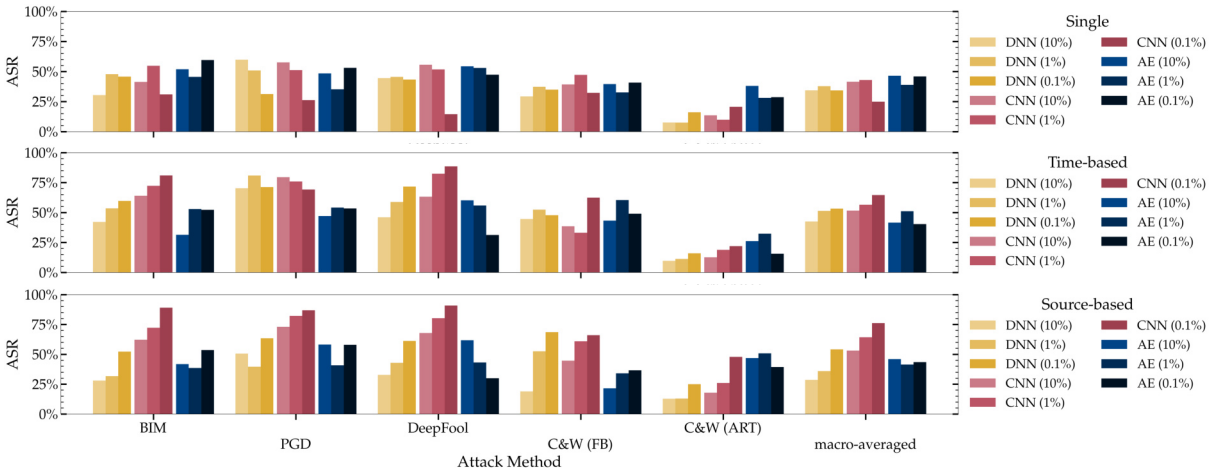


Figure 30: Attack Success Rate of Substitute Attacks with varying Percentages of Training Data on CSE-CIC-IDS2018

Considering these results, we can conclude that all three model architectures on all datasets are subject to substitute attacks based on the transferability of adversarial examples generated by either white-box attack method. This observation holds for both single feature perturbations and attacks using feature subsets according to domain constraints, and shows success rates, which are only slightly lower than the corresponding direct application of white-box attacks. Additionally, we show that such attacks are possible with only 0.1% training data and expect that even less data is sufficient.

5.3.3 Black-Box Attack Results

In this section, we describe the results when applying black-box attacks including ZOO, the Boundary attack and HopSkipJump. Such attacks do not require internal access to the underlying deep learning model and therefore are more realistically applicable in a real-world setting. However, these black-box attacks are computational expensive when compared to white-box attacks, which is the reason we limit the number of original samples that are used for the generation to 100 for each class. The next three paragraphs show the resulting ASRs for the UNSW-NB15, CIC-IDS2017 and CSE-CIC-IDS2018 dataset with corresponding feature subsets according to domain constraints.

Black-Box Attacks on UNSW-NB15 Figure 31 shows the black-box attack results of the UNSW-NB15 dataset. The ZOO attack cannot produce any successful adversarial examples for the DNN and CNN model, but has a macro-averaged ASR of 23% for the AE-based model. The

Boundary attack performs better with macro-averaged ASRs of 27%, 45% and 31%, for DNN, CNN and AE, respectively. This attack is especially successful for the class *Analysis* with ASRs of over 60% for all models. However, HopSkipJump outperforms both methods, and achieves ASRs for all classes of 75% and above, only the AE model shows a higher robustness for a few classes. Therefore, the HopSkipJump attack has a higher ASR than attacks based on substitute models, and even higher than white-box attacks. When considering the computational effort necessary for these results, however, HopSkipJump is less effective in finding adversarial examples than, e.g., BIM. We do not empirically analyze the computational effort for each method, but to give a rough figure, HopSkipJump takes for the computation of 100 samples per class about 50 times as long as BIM with 10000 samples per class. Regarding the ASR of individual classes, the results show no significant differences and all classes are equally effective to manipulate using HopSkipJump. When comparing the success of different model architectures, the CNN model appears to be slightly more vulnerable to the Boundary attack and HopSkipJump, but only by a small percentage.

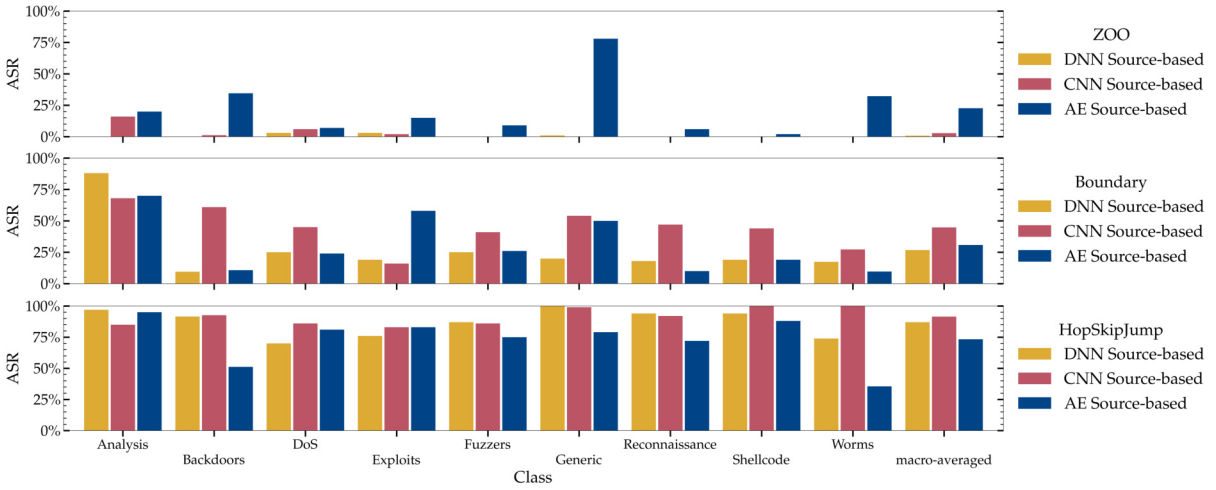


Figure 31: Attack Success Rate of Black-Box Attacks on UNSW-NB15

Black-Box Attacks on CIC-IDS2017 We analyze the vulnerability of classifiers trained on the CIC-IDS2017 dataset using black-box attacks with two domain-specific feature subsets (time- and source-based). The ZOO attack can produce more successful adversarial examples than on the UNSW-NB15 dataset, with macro-averaged ASRs of around 20% for the time-based and 30% for the source-based subset. However, dissimilar to the performance on the UNSW-NB15 dataset, the Boundary attack achieves lower ASRs than ZOO for both DNN and AE, but has similar results for the CNN model. Overall, the HipSkipJump attack performs significantly better and reaches ASRs of 60% for DNN with the time-based subset and CNN with both subsets. The AE seems more robust, with an ASR of around 20%. Regarding the two feature subsets, we can observe differences in their effect on the prediction, but they seem to be dependent on the class, model and attack method. Therefore, we cannot conclude a better effectiveness of changing the class prediction for either subset. Analyzing each class in more detail, we see rather low success rates for *DDoS*, *DoS Hulk* (except DNN time-based) and *SSHPatator*, but with large variations of ASRs between 0% and up to 75%. We conclude that black-box attacks are effective to create adversarial examples for classifiers of the CIC-IDS2017 but show large variations regarding the different classes and classifier architectures. White-box and substitute attacks seem to achieve more reliable results for this dataset.

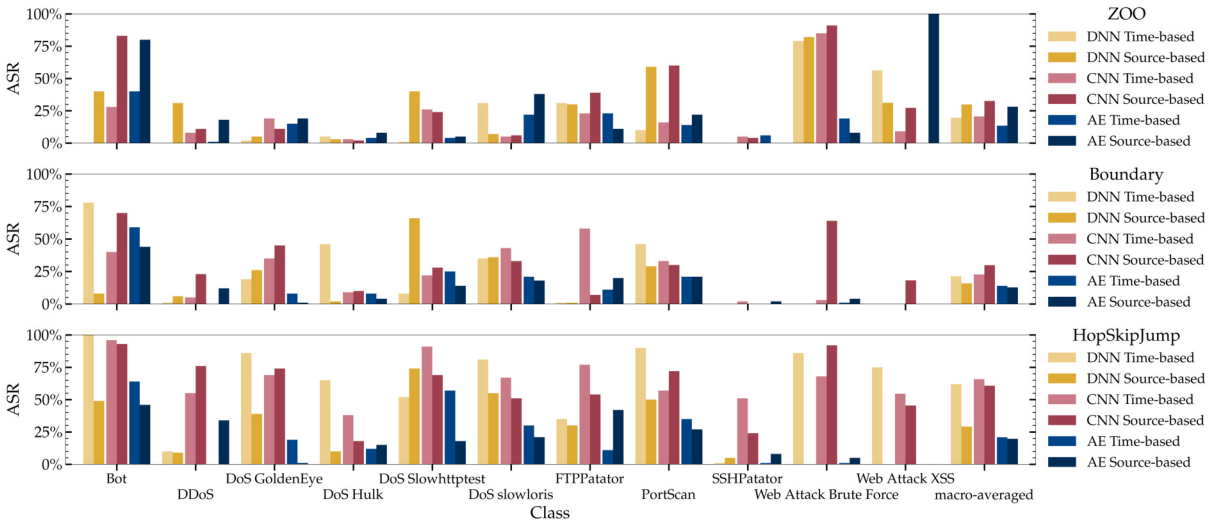


Figure 32: Attack Success Rate of Black-Box Attacks on CIC-IDS2017

Black-Box Attacks on CSE-CIC-IDS2018 The black-box attack results on the CSE-CIC-IDS2018 dataset are considerably different when compared to the other dataset. Foremost, the attack methods achieve significantly better results, with macro-averaged ASRs of around 75% for both ZOO and the Boundary attack. However, HipSkipJump can produce a success rate of 100% for all classes, models, and feature subsets. This means that the attack was able to find restricted perturbations that fool the model for every given sample. Note that this calculation does not include the class *SQl Injection* for the AE classifier, since no samples were correctly classified before the attack. In comparison to the CIC-IDS2017 dataset, we can observe that the results of the Boundary attack are less dependent on the class, model architecture or feature subset, and the attack achieves ASRs of over 50% for all classes. Only for *SQL Injection* this attack has a lower ASR of less than 50%. The ZOO attack shows larger variation between classes, but overall performs similarly well.

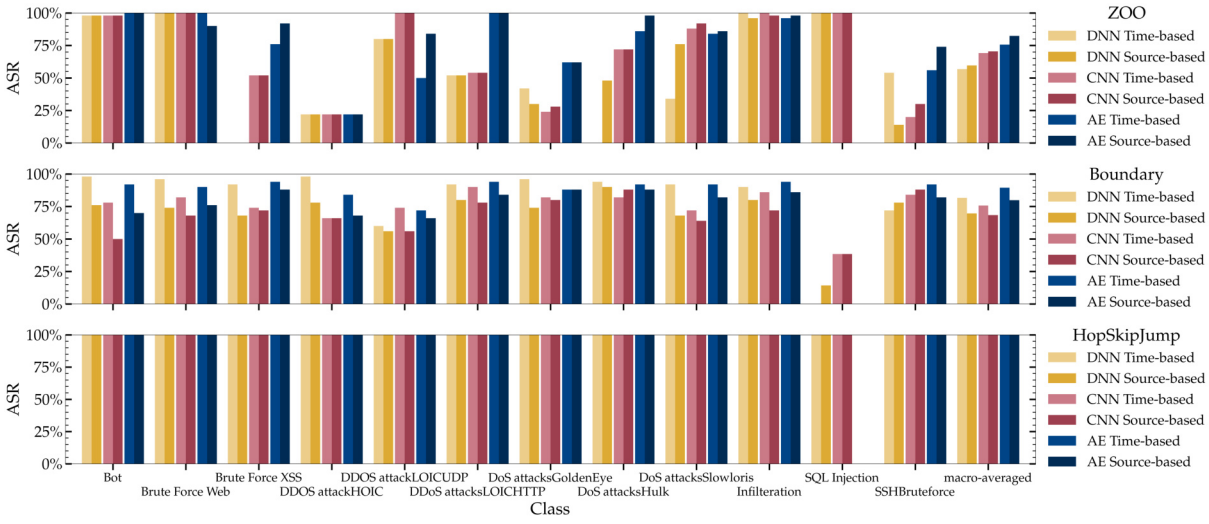


Figure 33: Attack Success Rate of Black-Box Attacks on CSE-CIC-IDS2018

5.3.4 Attacks against Binary Classifiers

While the focus of our work is in multiclass classifiers, we also test the possibility of adversarial attacks against binary classifiers. Figure 34 shows a summary of the attack results when applied to the binary classifiers of each dataset. The results are macro-averaged over both the time-based and source-based subsets and all three model architectures to give a comprehensive summary.

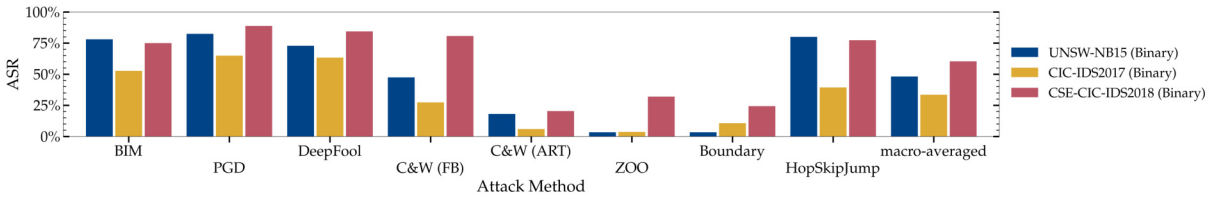


Figure 34: Attack Success Rate on Binary Classifiers

We can observe that binary classifiers are similarly vulnerable to adversarial attacks as multiclass classifiers. The macro-averaged ASRs across all methods are 48.2%, 33.5% and 60.4% for the UNSW-NB15, CIC-IDS2017 and CSE-CIC-IDS2018 dataset, respectively. The binary classifiers of the CIC-IDS2017 dataset are generally more robust and have lower ASRs for almost all attack methods. Regarding white-box attacks, BIM, PGD and DeepFool achieve the best results, while C&W (FB) has an especially high ASR for the CSE-CIC-IDS2018 dataset. C&W (ART) performs, similar to the attacks on multiclass models, the worst for all datasets. Considering black-box attacks, both ZOO and the Boundary attack are unable to reliably produce successful adversarial examples, but HopSkipJump reaches high ASRs of over 75% for UNSW-NB15 and CSE-CIC-IDS2018, and 40.5% for CIC-IDS2017.

5.3.5 Analysis of Adversarial Examples

In this section, we further analyze successful adversarial examples, which can change the prediction of an attack class to *Benign*. While we restrict the features that are manipulated in an attack to either a single feature or a subset, which only includes features that are most likely possible to change according to domain constraints, we do not further restrict the size of the perturbation. Figure 35 summarizes the perturbation size for each attack method and dataset using the macro-averaged mean absolute error (MAE). An average MAE of 0.4, which is the maximum for all attacks, corresponds to a change of 0.4 for the normalized value (capped between 0 and 1), e.g., a feature value which was 1 is after the manipulation 0.6. Note that this manipulation does not necessarily correspond to a 40% change in the original feature value because it depends on the applied normalization technique. Since the perturbed feature value is restricted to stay within 0 and 1, an adversarial attack cannot create values that do not exist in the original data.

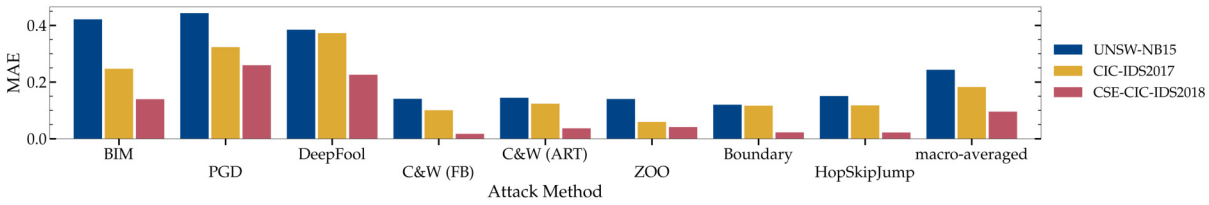


Figure 35: Perturbation Size of Successful Adversarial Examples

The results show that BIM, PGD and DeepFool produce the largest perturbations with MAEs of up to 0.4, while both C&W implementations stay under 0.2. This is not surprising, since the C&W attack tries to minimize the perturbation in its optimization process. BIM and PGD, on the other hand, apply step-wise fixed-size perturbations and do not aim to minimize them. DeepFool should theoretically also produce minimal perturbations, but the reason that is not the case here could be our hyperparameter optimization process. We optimize the parameters of the white-box attacks using ASR as the only evaluation metric, which might lead to larger perturbations. Regarding the black-box setting, all three attacks create perturbations which are of similar size as the C&W attack. All these attacks use minimization optimizations, with ZOO even using the C&W algorithm internally. Furthermore, we can observe large differences between the datasets. The UNSW-NB15 dataset seems to require the largest perturbations for successful attacks. By far, the lowest MAE is measured for the CSE-CIC-IDS2018 dataset, especially for the attack methods using perturbation minimization. The MAE of HopSkipJump is only 0.02, indicating minimal changes in the features. Combining the results from this analysis with the 100% success rate of the HopSkipJump attack on the CSE-CIC-IDS2018 dataset, it underlines the vulnerability of classifiers trained using this dataset, even with minimal changes in the input.

5.3.6 Summary

We analyzed the effect of targeted changes in the model input of a single feature and feature subsets with direct model access (white-box attacks) and through training of substitute models. Furthermore, we tested the vulnerability regarding black-box attacks, which require no access to the model. Based on our analysis, we can conclude that deep learning based IDS are generally vulnerable to adversarial attacks, with high success rates of above 75%. We cannot find significant differences between model architectures, but find large differences in the vulnerability between datasets. In particular, models trained on the CSE-CIC-IDS2018 dataset are especially susceptible to evasion attacks. The UNSW-NB15 dataset shows a higher robustness to white-box attacks and the CIC-IDS2017 dataset to black-box attacks, but it is more vulnerable to single feature perturbations. When applying attacks based on substitute models, we discover that 0.1% training data is sufficient to create transferable adversarial examples that fool the victim model. Furthermore, we examine the perturbation size of the attack methods and conclude that the HopSkipJump attack is most effective in finding minimal adversarial examples. Regarding white-box attacks, BIM and PGD outperform the other methods but create larger changes to the input than the C&W attack. Figure 36 summarizes the ASR of each attack method for the three datasets by showing the macro average across model architectures using perturbable feature subsets.

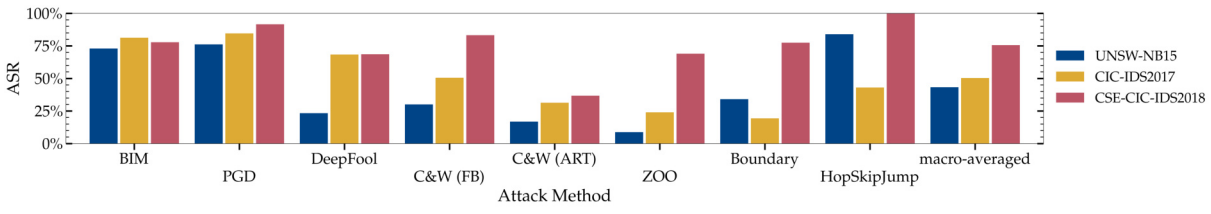


Figure 36: Comparison of Attack Success Rates across Datasets

6 Discussion

In the next sections, we will further discuss our findings. First, we compare our classification and attack results to other publications. Then, we discuss our results and the meaning of them for the application of deep learning based IDS in the real world.

6.1 Comparison of Classification Results

We compare our classification results with work that uses the same datasets and deep learning architectures. Even though the focus of our work lies in the success of adversarial attacks, it is important to achieve comparable results with our trained classifiers. State-of-the-art models are the basis to our study and needed to give a reasonable evaluation of the vulnerability through adversarial attacks. To make the results as comparable as possible, we only include work that uses roughly the same methodology, but still, the direct comparison is difficult. This is mostly due to the following reasons: First, some work only considers binary classifiers or multiclass classifiers with a small subset of classes, which makes them difficult to compare to our results. Another difficulty is due to the usage of different feature subsets. Especially the feature *Destination Port* is in a proportion of studies removed and in others kept, in which it shows a high importance for classification of the attacks [86]. We decided to remove it, since it can lead to overfitting regarding this feature, e.g., attacks are only recognized by the port since corresponding benign traffic is missing. The third problem consists of different metrics that are reported. A large proportion of the work in this domain only reports accuracy or weighted scores. Such weighted scores are calculated using the number of samples in each class, which is an unfair metric in the case of imbalanced data, and all IDS datasets are highly imbalanced [13]. Macro-averaged scores, on the other hand, consider each class equally and are better suited for data with large differences in the number of samples in each class. Finally, some papers report the results directly from the cross validation process and not from a separate test set. Since the optimization of model hyperparameters is done using the same data, knowledge is leaked between train and test folds, which can lead to overfitting and overly high result metrics [34]. Table 16 shows our results in comparison to results from other work. Unfortunately, none of the considered studies report their macro-averaged F1 scores. Therefore, we calculate this metric if the F1 scores for each class are provided. In other cases, we can only compare the accuracy and weighted F1 score with our results.

The authors of the UNSW-NB15 dataset analyze their dataset by comparing classification results with the KDDCUP'99 dataset. They achieve an accuracy of 0.81 with a DNN classifier [71]. Yang et al. [109] also use a DNN-based approach and reach an accuracy of 0.93 and a weighted F1 score of 0.94. Another method using a CNN by Ashiku and Dagli [10] has an accuracy of 0.96, but unfortunately, they do not report other metrics of their model. The CNN model from Jiang et al. [48] achieves a considerably lower accuracy of 0.75, but since they report the individual F1 scores for each class, we can calculate a macro-averaged F1 score of 0.4 for their model. Regarding AEs as a feature extraction technique for the UNSW-NB15 dataset, Khan et al. [51] use a deep AE and reach a weighted F1 score of 0.91. We calculate a macro-averaged score of 0.53 from their individual F1 scores, but it has to be noted that their results seem to be inconsistent with the reported precision and recall metrics. Andresini et al. [9] give a broader overview by considering the UNSW-NB15 and the CIC-IDS2017 dataset using DNNs, CNNs and AEs, but only for binary classification. They present accuracy scores of 0.86, 0.88 and 0.79 for DNN, CNN and AE, respectively. Our classifiers for the UNSW-NB15 dataset perform in most cases better than the compared studies, and are therefore well suited for this dataset.

The second dataset we use in our research is the CIC-IDS2017 dataset. Sharafaldin et al. [93] present this dataset in their paper and apply a DNN classifier that achieves a weighted F1 score

Table 16: Comparison of Classification Results
B = binary classification, * = calculated from available class F1 scores

Paper	Dataset	Method	Accuracy	Weighted F1	Macro F1
Moustafa and Slay (2016) [71]	UNSW-NB15	DNN	0.81		
Andresini et al. (2020) [9]	UNSW-NB15	DNN	0.86_B	0.91_B	
Yang et al. (2020) [109]	UNSW-NB15	DNN	0.93	0.94	
This research	UNSW-NB15	DNN	0.98	0.98	0.57
Andresini et al. (2020) [9]	UNSW-NB15	CNN	0.88_B	0.92_B	
Jiang et al. (2020) [48]	UNSW-NB15	CNN	0.75	0.78	0.40^*
Ashiku and Dagli (2021) [10]	UNSW-NB15	CNN	0.96		
This research	UNSW-NB15	CNN	0.98	0.98	0.56
Andresini et al. (2020) [9]	UNSW-NB15	AE	0.79_B	0.83_B	
Khan et al. (2019) [51]	UNSW-NB15	AE	0.89	0.91	0.53^*
This research	UNSW-NB15	AE	0.98	0.98	0.53
Sharafaldin et al. (2018) [93]	CIC-IDS2017	DNN		0.76	
Andresini et al. (2020) [9]	CIC-IDS2017	DNN	0.96_B	0.90_B	
Gamage and Samarabandu (2020) [31]	CIC-IDS2017	DNN	1.00	1.00	
Bulavas et al. (2021) [13]	CIC-IDS2017	DNN		0.98	
This research	CIC-IDS2017	DNN	0.99	0.99	0.83
Andresini et al. (2020) [9]	CIC-IDS2017	CNN	0.96_B	0.89_B	
Zhang et al. (2019) [115]	CIC-IDS2017	CNN	1.00	1.00	
This research	CIC-IDS2017	CNN	1.00	1.00	0.84
Andresini et al. (2020) [9]	CIC-IDS2017	AE	0.97_B	0.92_B	
Gamage and Samarabandu (2020) [31]	CIC-IDS2017	AE	0.98	0.98	
This research	CIC-IDS2017	AE	0.99	0.99	0.76
Kanimozhi and Jacob (2019) [49]	CSE-CIC-IDS2018	DNN	1.00_B	1.00_B	
Farhan et al. (2020) [27]	CSE-CIC-IDS2018	DNN	0.90		0.61^*
Gamage and Samarabandu (2020) [31]	CSE-CIC-IDS2018	DNN	0.98	0.98	
Zhang et al. (2020) [112]	CSE-CIC-IDS2018	DNN	0.99	0.96	
Bulavas et al. (2021) [13]	CSE-CIC-IDS2018	DNN		0.96	
This research	CSE-CIC-IDS2018	DNN	0.98	0.98	0.70
Zhang et al. (2020) [112]	CSE-CIC-IDS2018	CNN	0.99	0.96	
This research	CSE-CIC-IDS2018	CNN	0.96	0.96	0.68
Gamage and Samarabandu (2020) [31]	CSE-CIC-IDS2018	AE	0.98	0.98	
This research	CSE-CIC-IDS2018	AE	0.98	0.98	0.68

of 0.76. Most other research, which uses this dataset for classification, reports considerably higher weighted F1 scores of between 0.98 and 1. Bulavas et al. [13] achieve 0.98 with a DNN, and Zhang et al. [115] 1.0 with their CNN-based method. Gamage and Samarabandu [31] compare their DNN and AE models for both the CIC-IDS2017 and CSE-CIC-IDS2018 datasets and reach weighted scores of above 0.98 for all of them. Our classifiers perform similarly well, but the comparison is less informative because none of the papers report macro-averaged or class F1 scores.

The CSE-CIC-IDS2018 dataset is the final dataset we consider, and shows similar weighted scores as work using the CIC-IDS2017 dataset. Kanimozhi and Jacob [49] reach an accuracy of nearly 100%, but only consider Bot attacks using binary classification. Using only a subset of the attack classes, Farhan et al. [27] achieve an accuracy of 0.9 and a macro-averaged F1 score of 0.61 (calculated from individual class scores) with their DNN. Zhang et al. [112] also use DNNs but with the goal of testing adversarial robustness and report an accuracy and F1 score of 0.99 and 0.96, respectively.

Considering the results from this comparison, we can conclude that our classifiers achieve state-of-the-art performance, and are therefore suited for tests of their vulnerability against adversarial attacks.

6.2 Comparison of Attack Results

In this section, we aim to compare our attack results with other publications that use the same attack method, model architecture and dataset. Unfortunately, publications which are directly comparable to our methodology are rare, and even for them, a direct comparison is difficult. Since the evaluation of adversarial attacks requires a classification system, the same difficulties as described for the comparison of classification results apply in addition to different attack methodologies. Nevertheless, we try to present a small comparison of our results with the results of three other papers, which is shown in Table 17.

Table 17: Comparison of Attack Results
* = calculated from accuracy reduction

Paper	Dataset	Classifier	Method	ASR
Abou Khamis and Matrawy (2020) [3]	UNSW-NB15	DNN	BIM	55.0%*
This research	UNSW-NB15	DNN	BIM	66.4%
Abou Khamis and Matrawy (2020) [3]	UNSW-NB15	DNN	PGD	55.0%*
This research	UNSW-NB15	DNN	PGD	63.6%
Abou Khamis and Matrawy (2020) [3]	UNSW-NB15	DNN	DeepFool	60.0%*
This research	UNSW-NB15	DNN	DeepFool	27.7%
Abou Khamis and Matrawy (2020) [3]	UNSW-NB15	DNN	C&W	26.0%*
This research	UNSW-NB15	DNN	C&W	37.3%
Abou Khamis and Matrawy (2020) [3]	UNSW-NB15	CNN	BIM	51.0%*
This research	UNSW-NB15	CNN	BIM	72.8%
Abou Khamis and Matrawy (2020) [3]	UNSW-NB15	CNN	PGD	51.0%*
This research	UNSW-NB15	CNN	PGD	81.6%
Abou Khamis and Matrawy (2020) [3]	UNSW-NB15	CNN	DeepFool	28.0%*
This research	UNSW-NB15	CNN	DeepFool	29.4%
Abou Khamis and Matrawy (2020) [3]	UNSW-NB15	CNN	C&W	30.0%*
This research	UNSW-NB15	CNN	C&W	29.1%
Teuffenbach et al. (2020) [100]	CIC-IDS2017	DNN	BIM	73.3%
This research	CIC-IDS2017	DNN	BIM	86.9%
Teuffenbach et al. (2020) [100]	CIC-IDS2017	DNN	C&W	55.6%
This research	CIC-IDS2017	DNN	C&W	52.7%
Zhang et al. (2020) [112]	CSE-CIC-IDS2018	DNN	Boundary	21.8%
This research	CSE-CIC-IDS2018	DNN	Boundary	81.7%
Zhang et al. (2020) [112]	CSE-CIC-IDS2018	DNN	HopSkipJump	27.2%
This research	CSE-CIC-IDS2018	DNN	HopSkipJump	100%
Zhang et al. (2020) [112]	CSE-CIC-IDS2018	CNN	Boundary	25.3%
This research	CSE-CIC-IDS2018	CNN	Boundary	75.5%
Zhang et al. (2020) [112]	CSE-CIC-IDS2018	CNN	HopSkipJump	29.8%
This research	CSE-CIC-IDS2018	CNN	HopSkipJump	100%

Abou Khamis and Matrawy [3] test the vulnerability of DNN and CNN models against white-box attacks on the UNSW-NB15 dataset. Their approach is based on binary classifiers, discriminating only between benign and attack traffic. Unfortunately, they do not report their success rate, but we derive this metric from the reported reductions of accuracy for a rough estimate. The ASRs they achieve are similar to our results and differ only slightly. The most significant difference can be observed for DeepFool on DNNs, for which Abou Khamis and Matrawy reach an ASR of about 60%, while our approach only has an ASR of 27.7%.

Regarding the CIC-IDS2017 dataset, Teuffenbach et al. [100] test the vulnerability of DNNs using BIM and the C&W attack. While their ASRs only consider the class *DDoS*, they use feature subsets based on domain constraints, similar to our approach. However, they propose a more advanced grouping-based approach with an underlying cost function rather than strict feature subsets. The results of Teuffenbach et al. are similar to the ASRs we obtained. They also

report a higher success rate of BIM than C&W but with the cost of larger perturbations, which is equal to our findings.

Finally, we compare our results of black-box attacks on the CSE-CIC-IDS2018 dataset. Zhang et al. [112] test the success rate of the Boundary attack and HopSkipJump using DNNs and CNNs. Our results show a similar difference between these two attack methods but are significantly higher with 81.7% and 100% for Boundary and HopSkipJump using DNNs, while they achieved ASRs of only 21.8% and 27.2%, respectively. A possible reason for this difference could be further restrictions the authors put on their perturbations. In addition to feature subsets, which we also used, they constrain perturbations in their size and relations to other features, e.g., in the case of statistical features of the same metric.

6.3 Discussion of Attack Results

Our results indicate that deep learning based IDS are generally vulnerable to adversarial attacks of different kinds. But our approach has a few shortcomings, which we will discuss in this section.

6.3.1 Domain-Specific Restrictions

All the attack methods we utilize in our study were developed for the computer vision domain. Generally, the methods are applicable for any data type, but data for intrusion detection is different from image data and the attack methods are not optimized for that. One of these differences is the restriction of changes, which depends on the specific feature. These restrictions include features that cannot be changed at all, features that can be changed but only for certain attacks, and restrictions on the size of the perturbation that is added. Furthermore, not all features are independent but must reflect changes of other features, e.g., when the total packet size is increased, so does the mean. The attack algorithms do not allow for such complex constraints, and even allowing only a subset of features to be changed can be challenging. To incorporate the domain-specific constraints, we partially change the attack methods to only consider a predefined subset of features but enforce these restrictions by removing any perturbation on other than the allowed features, see Section 4.4.7 for more details. Regarding the size of perturbation, we do not enforce a strict limit but analyze the resulting differences, see Section 5.3.5. However, we do not restrict perturbations which might require changes in other features. Therefore, our attack results might be higher than when considering this restriction. Future work should extend the capabilities of existing adversarial attack algorithms to solve the described limitations.

6.3.2 Adversarial Attacks in the Real World

Even when considering most of the domain-specific restrictions, the question remains whether these adversarial attacks are also possible for real-world IDS. Based on our work alone, we cannot definitively answer this question, but our results indicate that deep learning based IDS are generally highly vulnerable to adversarial attacks. The effort to test if the attacks are also possible in a real-world setting is high and requires setting up an own testing network infrastructure and dataset. When using an existing dataset, checking if the underlying intrusion is still successful is hardly possible, which also makes the entire approach harder to compare to existing work. Some papers cover real-world limitations further, e.g., Kuppa et al. [56], which test the validity of packets by transforming original *pcap* files, or Hashemi et al. [37], which define a set of operations that are possible to apply to real traffic. However, future work should

investigate real-world limitations and the resulting vulnerability of IDS to adversarial attacks further.

7 Conclusion

In our approach to analyze the vulnerability of deep learning based IDS, we first trained three model architectures using three publicly available datasets. We show that our classifiers achieve state-of-the-art performance and can confidently discriminate between benign traffic and attack classes. We then applied adversarial attacks on them in white- and black-box settings, while limiting the perturbations to domain-specific constraints. Additionally, we tested the success of attacks when manipulating only a single feature and further evaluated the most effective features using XAI techniques.

Our main findings are a generally high vulnerability of deep learning based IDS systems to adversarial attacks, with success rates of above 75%. While we cannot find significant differences between model architectures, we observe large deviations between the datasets. Our results indicate that the vulnerability to adversarial attacks lies in the data a model is trained on and less in the structure of the model itself. Furthermore, we show that these models are not only vulnerable to white-box attacks but also to substitute and black-box attacks, making it likely that these flaws could be taken advantage of for real-world systems. Additionally, we observe that even minimal changes in the data can fool the models to think that an attack is benign traffic, making it a serious security threat. While these results indicate a high possibility of real-world adversarial attacks, future work should investigate the feasibility of such attacks further. Finally, the results from our study should be used to make IDS more robust and to implement successful defenses that protect these systems from adversaries.

References

[1] Kjersti Aas, Martin Jullum, and Anders Løland. Explaining individual predictions when features are dependent: More accurate approximations to shapley values. *Artificial Intelligence*, 298:103502, 2021.

[2] Mohammad Hamid Abdulraheem and Najla Badie Ibraheem. A detailed analysis of new intrusion detection dataset. *Journal of Theoretical and Applied Information Technology*, 97(17):4519–4537, 2019.

[3] Rana Abou Khamis and Ashraf Matrawy. Evaluation of adversarial training on different types of neural networks in deep learning-based idss. In *2020 international symposium on networks, computers and communications (ISNCC)*, pages 1–6. IEEE, 2020.

[4] Rana Abou Khamis, M Omair Shafiq, and Ashraf Matrawy. Investigating resistance of deep learning-based ids against adversaries using min-max optimization. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2020.

[5] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[6] Naveed Akhtar, Ajmal Mian, Navid Kardan, and Mubarak Shah. Advances in adversarial attacks and defenses in computer vision: A survey. *IEEE Access*, 9:155161–155196, 2021.

[7] Muataz Salam Al-Daweri, Khairul Akram Zainol Ariffin, Salwani Abdullah, and Mohamad Firham Efendy Md. Senan. An analysis of the kdd99 and unsw-nb15 datasets for the intrusion detection system. *Symmetry*, 12(10):1666, 2020.

[8] Khaled Alrawashdeh and Carla Purdy. Toward an online anomaly intrusion detection system based on deep learning. In *2016 15th IEEE international conference on machine learning and applications (ICMLA)*, pages 195–200. IEEE, 2016.

[9] Giuseppina Andresini, Annalisa Appice, Nicola Di Mauro, Corrado Loglisci, and Donato Malerba. Multi-channel deep feature learning for intrusion detection. *IEEE Access*, 8:53346–53359, 2020.

[10] Lirim Ashiku and Cihan Dagli. Network intrusion detection system using deep learning. *Procedia Computer Science*, 185:239–247, 2021.

[11] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[12] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017.

[13] Viktoras Bulavas, Virginijus Marcinkevičius, and Jacek Rumiński. Study of multi-class classification algorithms' performance on highly imbalanced network intrusion datasets. *Informatica*, 32(3):441–475, 2021.

[14] James Cannady. Artificial neural networks for misuse detection. In *Proceedings of the 1998 National Information Systems Security Conference (NISSC'98)*, pages 443–456, 1998.

[15] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee, 2017.

[16] Identity Theft Resource Center. 2022 Annual Data Breach Report. `https://www.idtheftcenter.org/publication/2022-data-breachreport`, 2022. Accessed: 22nd January 2024.

[17] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*, 2018.

[18] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[19] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. In *2020 ieee symposium on security and privacy (sp)*, pages 1277–1294. IEEE, 2020.

[20] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 15–26, 2017.

[21] Tsung-Huan Cheng, Ying-Dar Lin, Yuan-Cheng Lai, and Po-Ching Lin. Evasion techniques: Sneaking through your intrusion detection/prevention systems. *IEEE Communications Surveys & Tutorials*, 14(4):1011–1020, 2011.

[22] Carlos Garcia Cordero, Sascha Hauke, Max Mühlhäuser, and Mathias Fischer. Analyzing flow-based anomaly intrusion detection using replicator neural networks. In *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, pages 317–324. IEEE, 2016.

[23] Islam Debicha, Thibault Debatty, Jean-Michel Dricot, and Wim Mees. Adversarial training for deep learning-based intrusion detection systems. *arXiv preprint arXiv:2104.09852*, 2021.

[24] Bo Dong and Xue Wang. Comparison deep learning method to traditional methods using for network intrusion detection. In *2016 8th IEEE international conference on communication software and networks (ICCSN)*, pages 581–585. IEEE, 2016.

[25] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.

[26] Communications Security Establishment and Canadian Institute for Cybersecurity. A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018). `https://registry.opendata.aws/cse-cic-ids2018`, 2018. Accessed: 28th December 2023.

[27] Rawaa Ismael Farhan, Abeer Tariq Maolood, and NidaaFlaih Hassan. Performance analysis of flow-based attacks detection on cse-cic-ids2018 dataset using deep learning. *Indones. J. Electr. Eng. Comput. Sci*, 20(3):1413–1418, 2020.

[28] Qusyairi Ridho Saeful Fitni and Kalamullah Ramli. Implementation of ensemble learning and feature selection for performance improvements in anomaly-based intrusion detection systems. In *2020 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*, pages 118–124. IEEE, 2020.

[29] Romain Fontugne, Pierre Borgnat, Patrice Abry, and Kensuke Fukuda. Mawilab: combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking. In *Proceedings of the 6th International COnference*, pages 1–12, 2010.

[30] Ivo Frazão, Pedro Henriques Abreu, Tiago Cruz, Hélder Araújo, and Paulo Simões. Denial of service attacks: Detecting the frailties of machine learning algorithms in the classification process. In *Critical Information Infrastructures Security: 13th International Conference, CRITIS 2018, Kaunas, Lithuania, September 24-26, 2018, Revised Selected Papers 13*, pages 230–235. Springer, 2019.

[31] Sunanda Gamage and Jagath Samarabandu. Deep learning methods in network intrusion detection: A survey and an objective comparison. *Journal of Network and Computer Applications*, 169:102767, 2020.

[32] Pedro Garcia-Teodoro, Jesus Diaz-Verdejo, Gabriel Maciá-Fernández, and Enrique Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security*, 28(1-2):18–28, 2009.

[33] Amirhossein Gharib, Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. An evaluation framework for intrusion detection dataset. In *2016 International Conference on Information Science and Security (ICISS)*, pages 1–6. IEEE, 2016.

[34] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[35] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[36] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern recognition*, 77:354–377, 2018.

[37] Mohammad J Hashemi, Greg Cusack, and Eric Keller. Towards evaluation of nidss in adversarial setting. In *Proceedings of the 3rd ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks*, pages 14–21, 2019.

[38] Simon Hawkins, Hongxing He, Graham Williams, and Rohan Baxter. Outlier detection using replicator neural networks. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 170–180. Springer, 2002.

[39] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

[40] Haitao He, Xiaobing Sun, Hongdou He, Guyu Zhao, Ligang He, and Jiadong Ren. A novel multimodal-sequential approach based on multi-view features for network intrusion detection. *IEEE Access*, 7:183207–183221, 2019.

[41] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[42] Hannes Holm. Signature based intrusion detection for zero-day attacks:(not) a closed chapter? In *2014 47th Hawaii international conference on system sciences*, pages 4895–4904. IEEE, 2014.

[43] Yuming Hua, Junhai Guo, and Hua Zhao. Deep belief networks and deep learning. In *Proceedings of 2015 International Conference on Intelligent Computing and Internet of Things*, pages 1–4. IEEE, 2015.

[44] Olakunle Ibitoye, Omair Shafiq, and Ashraf Matrawy. Analyzing adversarial attacks against deep learning for intrusion detection in iot networks. In *2019 IEEE global communications conference (GLOBECOM)*, pages 1–6. IEEE, 2019.

[45] Félix Iglesias and Tanja Zseby. Analysis of network traffic features for anomaly detection. *Machine Learning*, 101:59–84, 2015.

[46] Tharmini Janarthanan and Shahrzad Zargari. Feature selection in unsw-nb15 and kddcup'99 datasets. In *2017 IEEE 26th international symposium on industrial electronics (ISIE)*, pages 1881–1886. IEEE, 2017.

[47] Ahmad Javaid, Quamar Niyaz, Weiqing Sun, and Mansoor Alam. A deep learning approach for network intrusion detection system. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, pages 21–26, 2016.

[48] Kaiyuan Jiang, Wenya Wang, Aili Wang, and Haibin Wu. Network intrusion detection combined hybrid sampling with deep hierarchical network. *IEEE access*, 8:32464–32476, 2020.

[49] V Kanimozhi and T Prem Jacob. Calibration of various optimized machine learning classifiers in network intrusion detection system on the realistic cyber dataset cse-cic-ids2018 using cloud computing. *International Journal of Engineering Applied Sciences and Technology*, 4(6):2455–2143, 2019.

[50] Richard A Kemmerer. Nstat: a model-based real-time network intrusion detection system. *Computer Science Department, University of California, Santa Barbara, Report TRCS97-18, http://www. cs. ucsb. edu/TRs/TRCS97-18. html*, 1997.

[51] Farrukh Aslam Khan, Abdu Gumaei, Abdelouahid Derhab, and Amir Hussain. A novel two-stage deep learning model for efficient network intrusion detection. *IEEE Access*, 7:30373–30385, 2019.

[52] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1):1–22, 2019.

[53] Jin Kim, Nara Shin, Seung Yeon Jo, and Sang Hyun Kim. Method of intrusion detection using deep neural network. In *2017 IEEE international conference on big data and smart computing (BigComp)*, pages 313–316. IEEE, 2017.

[54] Jiyeon Kim, Jiwon Kim, Hyunjung Kim, Minsun Shim, and Eunjung Choi. Cnn-based network intrusion detection against denial-of-service attacks. *Electronics*, 9(6):916, 2020.

[55] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[56] Aditya Kuppa, Slawomir Grzonkowski, Muhammad Rizwan Asghar, and Nhien-An Le-Khac. Black box attacks on deep anomaly detectors. In *Proceedings of the 14th international conference on availability, reliability and security*, pages 1–10, 2019.

[57] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pages 99–112. Chapman and Hall/CRC, 2018.

[58] Arash Habibi Lashkari, Gerard Draper Gil, Mohammad Saiful Islam Mamun, and Ali A Ghorbani. Characterization of tor traffic using time based features. In *International Conference on Information Systems Security and Privacy*, volume 2, pages 253–262. SciTePress, 2017.

[59] Dohyun Lee and Kyoungok Kim. An efficient method to determine sample size in oversampling based on classification complexity for imbalanced data. *Expert Systems with Applications*, 184:115442, 2021.

[60] Jian Li, Guo-Yin Zhang, and Guo-Chang Gu. The research and implementation of intelligent intrusion detection system based on artificial neural network. In *Proceedings of 2004 international conference on machine learning and cybernetics (IEEE Cat. No. 04EX826)*, volume 5, pages 3178–3182. IEEE, 2004.

[61] Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[62] Peng Lin, Kejiang Ye, and Cheng-Zhong Xu. Dynamic network anomaly detection system by using deep learning techniques. In *Cloud Computing–CLOUD 2019: 12th International Conference, Held as Part of the Services Conference Federation, SCF 2019, San Diego, CA, USA, June 25–30, 2019, Proceedings 12*, pages 161–176. Springer, 2019.

[63] Zilong Lin, Yong Shi, and Zhi Xue. Idsgan: Generative adversarial networks for attack generation against intrusion detection. In *Pacific-asia conference on knowledge discovery and data mining*, pages 79–91. Springer, 2022.

[64] Teng Long, Qi Gao, Lili Xu, and Zhangbing Zhou. A survey on adversarial attacks in computer vision: Taxonomy, visualization and future directions. *Computers & Security*, page 102847, 2022.

[65] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[66] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[67] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

[68] Mehdi Moradi and Mohammad Zulkernine. A neural network based system for intrusion detection and classification of attacks. In *Proceedings of the IEEE international conference on advances in intelligent systems-theory and applications*, pages 15–18. IEEE Luxembourg-Kirchberg, Luxembourg, 2004.

[69] Nour Moustafa and Jill Slay. The significant features of the unsw-nb15 and the kdd99 data sets for network intrusion detection systems. In *2015 4th international workshop on building analysis datasets and gathering experience returns for security (BADGERS)*, pages 25–31. IEEE, 2015.

[70] Nour Moustafa and Jill Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *2015 military communications and information systems conference (MilCIS)*, pages 1–6. IEEE, 2015.

[71] Nour Moustafa and Jill Slay. The evaluation of network anomaly detection systems: Statistical analysis of the unsw-nb15 data set and the comparison with the kdd99 data set. *Information Security Journal: A Global Perspective*, 25(1-3):18–31, 2016.

[72] Srinivas Mukkamala, Guadalupe Janoski, and Andrew Sung. Intrusion detection using neural networks and support vector machines. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*, volume 2, pages 1702–1707. IEEE, 2002.

[73] Reyadh Shaker Naoum, Namh Abdula Abid, and Zainab Namh Al-Sultani. An enhanced resilient backpropagation artificial neural network for intrusion detection system. *International Journal of Computer Science and Network Security (IJCSNS)*, 12(3):11, 2012.

[74] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, Ian Molloy, and Ben Edwards. Adversarial robustness toolbox v1.2.0. *CoRR*, 1807.01069, 2018.

[75] Quamar Niyaz, Weiqing Sun, and Ahmad Y Javaid. A deep learning based ddos detection system in software-defined networking (sdn). *arXiv preprint arXiv:1611.07400*, 2016.

[76] Sinem Osken, Ecem Nur Yildirim, Gozde Karatas, and Levent Cuhaci. Intrusion detection systems with deep learning: A systematic mapping study. In *2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT)*, pages 1–4. IEEE, 2019.

[77] Suad Mohammed Othman, Nabeel T Alsohybe, Fadl Mutaher Ba-Alwi, and Ammar Thabit Zahary. Survey on intrusion detection system types. *International Journal of Cyber-Security and Digital Forensics*, 7(4):444–463, 2018.

[78] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[79] Dan Pelleg, Andrew W Moore, et al. X-means: Extending k-means with efficient estimation of the number of clusters. In *Icml*, volume 1, pages 727–734, 2000.

[80] Xiao Peng, Weiqing Huang, and Zhixin Shi. Adversarial attack against dos intrusion detection: An improved boundary-based method. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1288–1295. IEEE, 2019.

[81] Ye Peng, Jinshu Su, Xiangquan Shi, and Baokang Zhao. Evaluating deep learning based network intrusion detection system in adversarial environment. In *2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, pages 61–66. IEEE, 2019.

[82] Aritran Piplai, Sai Sree Laya Chukkapalli, and Anupam Joshi. Nattack! adversarial attacks to bypass a gan based classifier trained to detect network intrusion. In *2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, pages 49–54. IEEE, 2020.

[83] Sasanka Potluri and Christian Diedrich. Accelerated deep neural networks for enhanced intrusion detection system. In *2016 IEEE 21st international conference on emerging technologies and factory automation (ETFA)*, pages 1–8. IEEE, 2016.

[84] Han Qiu, Tian Dong, Tianwei Zhang, Jialiang Lu, Gerard Memmi, and Meikang Qiu. Adversarial attacks against network intrusion detection in iot systems. *IEEE Internet of Things Journal*, 8(13):10327–10335, 2020.

[85] Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. In *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*, 2017.

[86] Kezhou Ren, Yifan Zeng, Zhiqin Cao, and Yingchao Zhang. Id-rdrl: a deep reinforcement learning-based feature selection intrusion detection model. *Scientific Reports*, 12(1):15370, 2022.

[87] Maria Rigaki. Adversarial deep learning against intrusion detection classifiers, 2017.

[88] Arnaud Rosay, Eloïse Cheval, Florent Carlier, and Pascal Leroux. Network intrusion detection: A comprehensive analysis of cic-ids2017. In *8th International Conference on Information Systems Security and Privacy*, pages 25–36. SCITEPRESS-Science and Technology Publications, 2022.

[89] Mahdis Saharkhizan, Amin Azmoodeh, Ali Dehghantanha, Kim-Kwang Raymond Choo, and Reza M Parizi. An ensemble of deep recurrent neural networks for detecting iot cyber attacks using network traffic. *IEEE Internet of Things Journal*, 7(9):8852–8859, 2020.

[90] Mostafa A Salama, Heba F Eid, Rabie A Ramadan, Ashraf Darwish, and Aboul Ella Hassanien. Hybrid intelligent intrusion detection scheme. In *Soft computing in industrial applications*, pages 293–303. Springer, 2011.

[91] Mohammed Sammany, Marwa Sharawi, Mohammed El-Beltagy, and Imane Saroit. Artificial neural networks architecture for intrusion detection systems and classification of attacks. In *The 5th international conference INFO2007*, pages 24–26, 2007.

[92] Iman Sharafaldin, Amirhossein Gharib, Arash Habibi Lashkari, and Ali A Ghorbani. Towards a reliable intrusion detection benchmark dataset. *Software Networking*, 2017(1):177–200, 2017.

[93] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1:108–116, 2018.

[94] Nathan Shone, Tran Nguyen Ngoc, Vu Dinh Phai, and Qi Shi. A deep learning approach to network intrusion detection. *IEEE transactions on emerging topics in computational intelligence*, 2(1):41–50, 2018.

[95] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[96] Deris Stiawan, Mohd Yazid Bin Idris, Alwi M Bamhdi, Rahmat Budiarto, et al. Cicids-2017 dataset feature analysis with information gain for anomaly detection. *IEEE Access*, 8:132911–132921, 2020.

[97] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[98] Tuan A Tang, Lotfi Mhamdi, Des McLernon, Syed Ali Raza Zaidi, and Mounir Ghogho. Deep learning approach for network intrusion detection in software defined networking. In *2016 international conference on wireless networks and mobile communications (WINCOM)*, pages 258–263. IEEE, 2016.

[99] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*, pages 1–6. Ieee, 2009.

[100] Martin Teuffenbach, Ewa Piatkowska, and Paul Smith. Subverting network intrusion detection: Crafting adversarial examples accounting for domain-specific constraints. In *Machine Learning and Knowledge Extraction: 4th IFIP TC 5, TC 12, WG 8.4, WG 8.9, WG 12.9 International Cross-Domain Conference, CD-MAKE 2020, Dublin, Ireland, August 25–28, 2020, Proceedings 4*, pages 301–320. Springer, 2020.

[101] Jonathan Uesato, Brendan O'donoghue, Pushmeet Kohli, and Aaron Oord. Adversarial risk and the dangers of evaluating against weak attacks. In *International Conference on Machine Learning*, pages 5025–5034. PMLR, 2018.

[102] Muhammad Usama, Muhammad Asim, Siddique Latif, Junaid Qadir, et al. Generative adversarial networks for launching and thwarting adversarial attacks on network intrusion detection systems. In *2019 15th international wireless communications & mobile computing conference (IWCMC)*, pages 78–83. IEEE, 2019.

[103] Zheng Wang. Deep learning-based intrusion detection with adversaries. *IEEE Access*, 6:38367–38384, 2018.

[104] Arkadiusz Warzyński and Grzegorz Kołaczek. Intrusion detection systems vulnerability on adversarial examples. In *2018 Innovations in Intelligent Systems and Applications (INISTA)*, pages 1–4. IEEE, 2018.

[105] Yihan Xiao, Cheng Xing, Taining Zhang, and Zhongkai Zhao. An intrusion detection model based on feature reduction and convolutional neural networks. *IEEE Access*, 7:42210–42219, 2019.

[106] Ryo Yamada and Shigeki Goto. Using abnormal ttl values to detect malicious ip packets. *Proceedings of the Asia-Pacific Advanced Network*, 34(0):27, 2013.

[107] Qiao Yan, Mingde Wang, Wenyao Huang, Xupeng Luo, and F Richard Yu. Automatically synthesizing dos attack traces using generative adversarial networks. *International journal of machine learning and cybernetics*, 10(12):3387–3396, 2019.

[108] Kaichen Yang, Jianqing Liu, Chi Zhang, and Yuguang Fang. Adversarial examples against the deep learning based network intrusion detection systems. In *MILCOM 2018-2018 ieee military communications conference (MILCOM)*, pages 559–564. IEEE, 2018.

[109] Yanqing Yang, Kangfeng Zheng, Bin Wu, Yixian Yang, and Xiujuan Wang. Network intrusion detection based on supervised adversarial variational auto-encoder with regularization. *IEEE access*, 8:42169–42184, 2020.

[110] Xinwei Yuan, Shu Han, Wei Huang, Hongliang Ye, Xianglong Kong, and Fan Zhang. A simple framework to enhance the adversarial robustness of deep learning-based intrusion detection system. *Computers & Security*, page 103644, 2023.

[111] Junhai Zhai, Sufang Zhang, Junfen Chen, and Qiang He. Autoencoder and its various variants. In *2018 IEEE international conference on systems, man, and cybernetics (SMC)*, pages 415–419. IEEE, 2018.

[112] Chaoyun Zhang, Xavier Costa-Pérez, and Paul Patras. Tiki-taka: Attacking and defending deep learning-based intrusion detection systems. In *Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop*, pages 27–39, 2020.

[113] Chaoyun Zhang, Xavier Costa-Perez, and Paul Patras. Adversarial attacks against deep learning-based network intrusion detection systems and defense mechanisms. *IEEE/ACM Transactions on Networking*, 30(3):1294–1311, 2022.

[114] Sicong Zhang, Xiaoyao Xie, and Yang Xu. A brute-force black-box method to attack machine learning-based systems in cybersecurity. *IEEE Access*, 8:128250–128263, 2020.

[115] Yong Zhang, Xu Chen, Lei Jin, Xiaojuan Wang, and Da Guo. Network intrusion detection: Based on deep hierarchical network and original flow data. *IEEE Access*, 7:37004–37016, 2019.

[116] Yu Zhang, Peter Tiňo, Aleš Leonardis, and Ke Tang. A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(5):726–742, 2021.

A Supporting Figures & Tables

A.1 Hyperparameters of Classifiers

Table A.1: Selected Parameters (DNN)

Parameter	Search Space	UNSW-NB15	CIC-IDS2017	CSE-CIC-IDS2018
hidden_layer_sizes	[64], [128, 64], [256, 129, 64]	[256, 128]	[256]	[256, 128]
epochs	10 - 100	44	69	58
learning_rate	0.0001 - 0.01	0.000272	0.000147	0.006957

Table A.2: Selected Parameters (CNN)

Parameter	Search Space	UNSW-NB15	CIC-IDS2017	CSE-CIC-IDS2018
filter_sizes	[32], [32, 64], [32, 64, 64]	[32]	[32, 64]	[32, 64, 64]
kernel_size	3, 5	3	3	5
hidden_layer_sizes	[64], [128, 64], [256, 129, 64]	[128, 64]	[128]	[128]
dropout	0 - 0.5	0.363336	0.408001	0.311925
epochs	10 - 100	63	22	69
learning_rate	0.0001 - 0.01	0.000229	0.000339	0.000316

Table A.3: Selected Parameters (AE)

Parameter	Search Space	UNSW-NB15	CIC-IDS2017	CSE-CIC-IDS2018
autoencoder loss	CosineSimilarity, MeanSquaredError	CosineSimilarity	MeanSquaredError	CosineSimilarity
autoencoder hidden_layer_sizes	[64], [128, 64], [256, 129, 64]	[256]	[256, 128, 64]	[256]
autoencoder l1	0.001 - 0.1	0.084955	0.033661	0.003722
autoencoder latent_size	0.2 - 0.8	0.633527	0.563861	0.630102
autoencoder epochs	10 - 100	14	13	59
autoencoder learning_rate	0.0001 - 0.01	0.003563	0.001417	0.000154
classifier hidden_layer_sizes	[64], [128, 64], [256, 129, 64]	[64, 32, 16]	[64, 32, 16]	[64, 32, 16]
classifier epochs	10 - 100	28	82	89
classifier learning_rate	0.0001 - 0.01	0.002399	0.000599	0.000478

A.2 Comprehensive SHAP Results

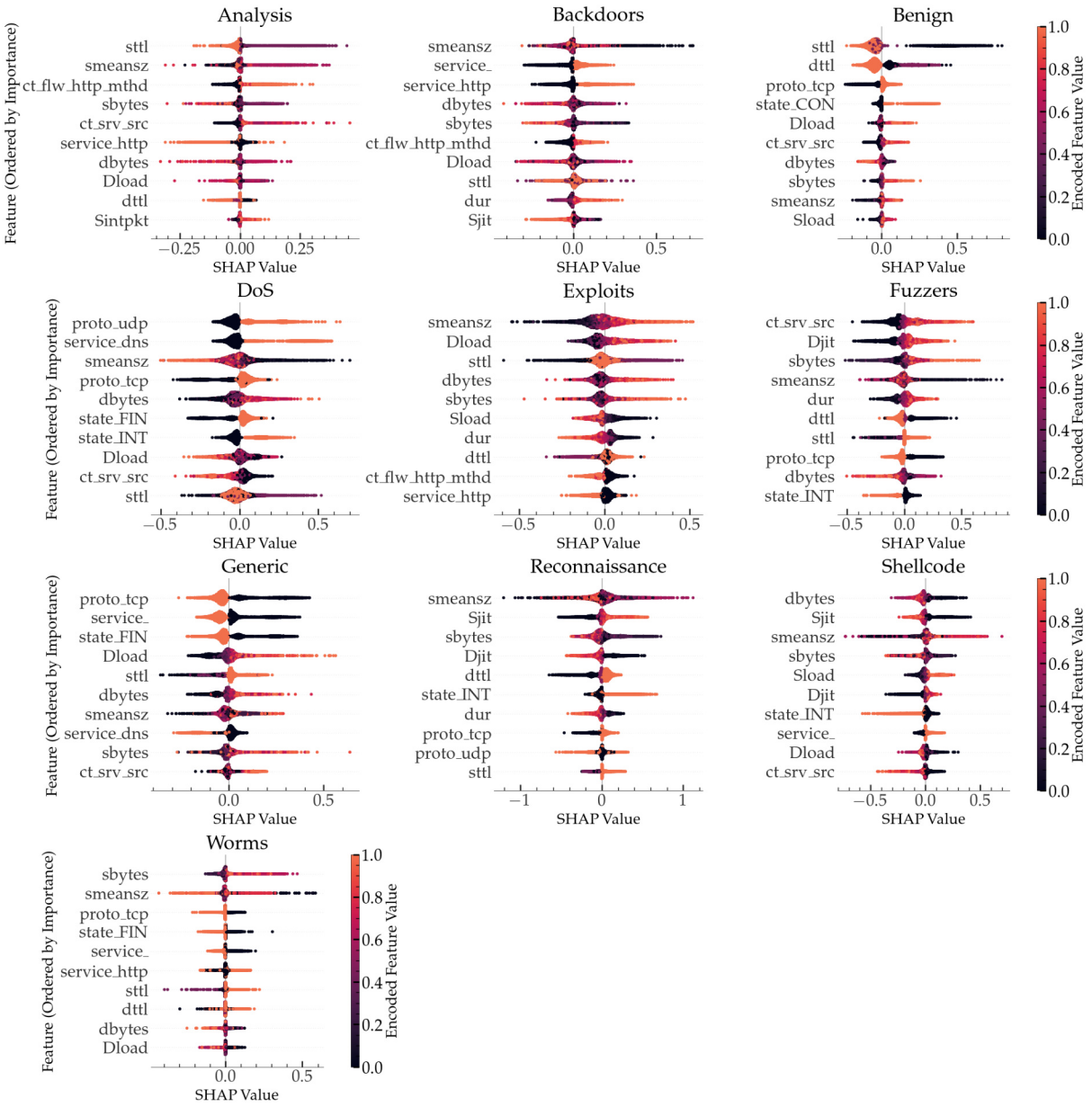


Figure A.1: SHAP Feature Importance of DNN on UNSW-NB15

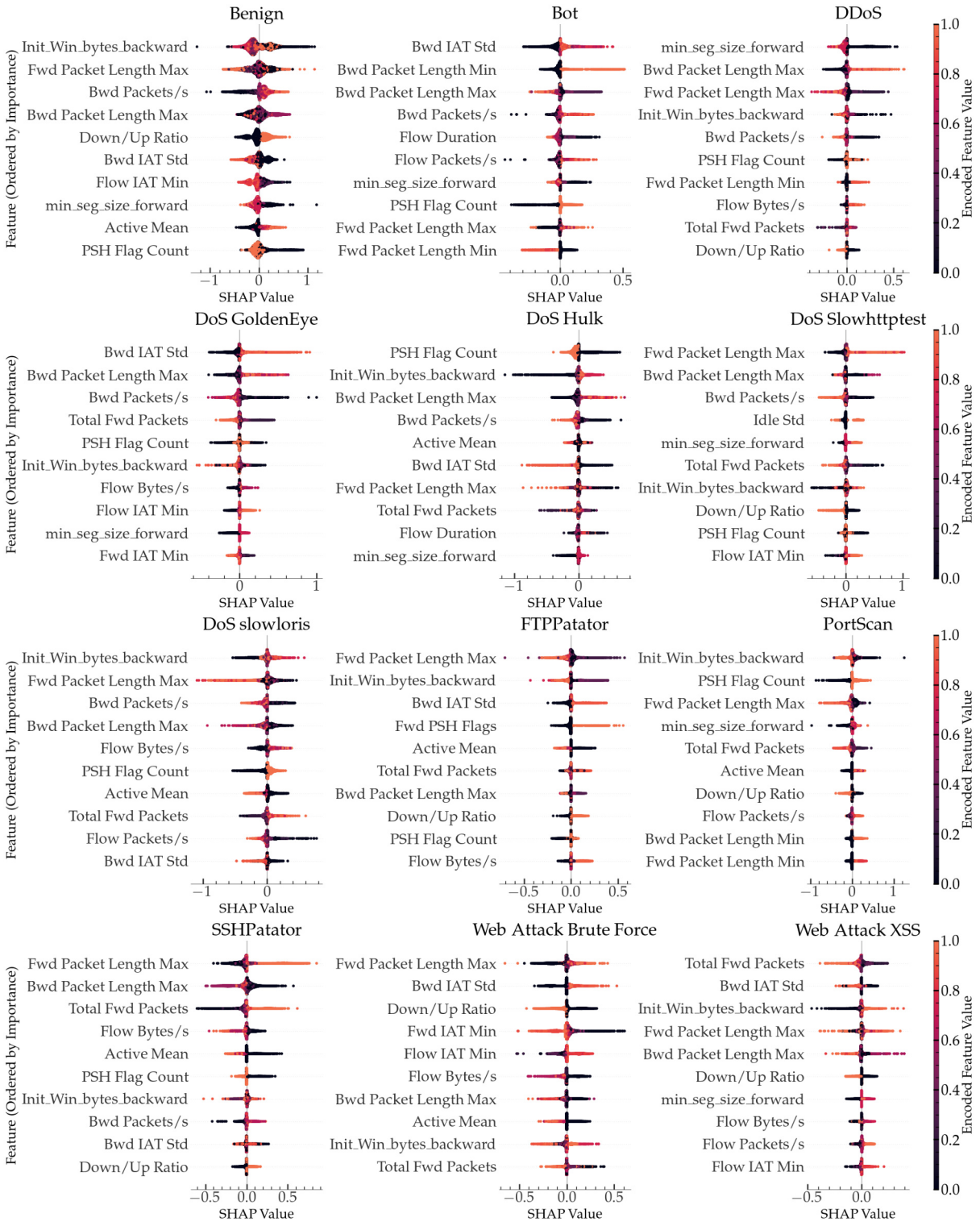


Figure A.2: SHAP Feature Importance of DNN on CIC-IDS2017

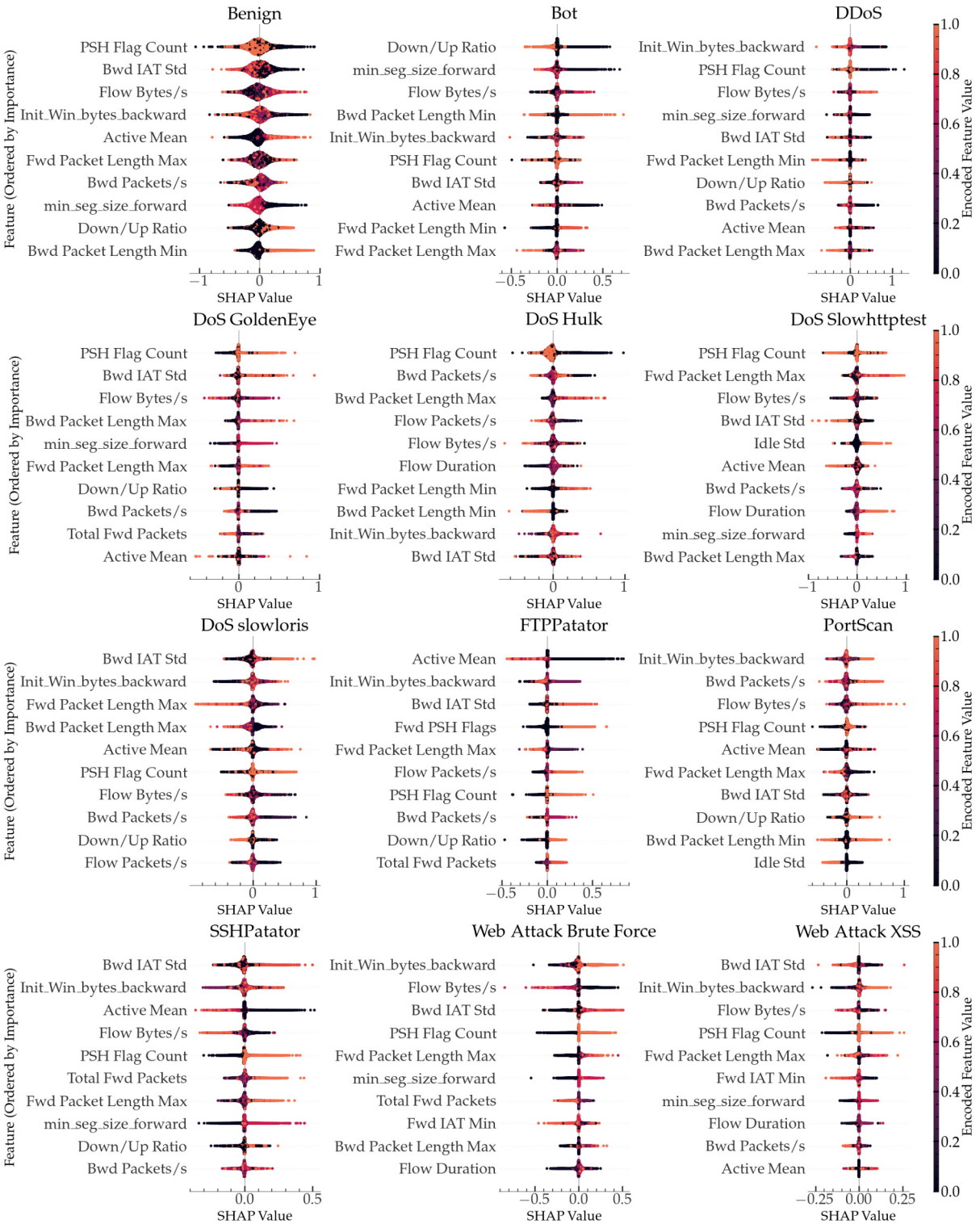


Figure A.3: SHAP Feature Importance of AE on CIC-IDS2017

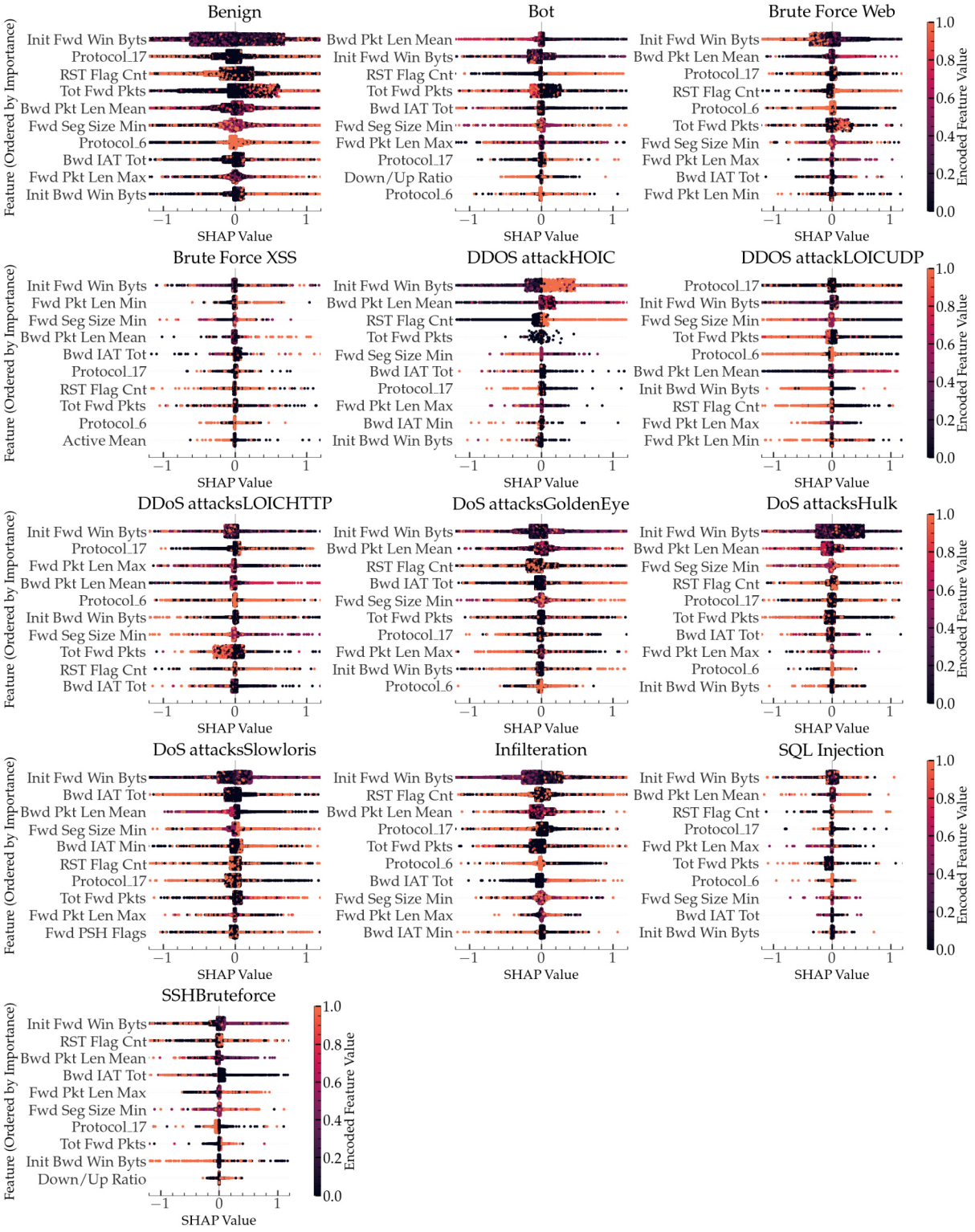


Figure A.4: SHAP Feature Importance of DNN on CSE-CIC-IDS2018

A.3 Selected Features for Single Feature Perturbation

Table A.4: Selected Features for Single Feature Perturbation of UNSW-NB15 (DNN)

	DNN				
Class	**BIM**	**PGD**	**DeepFool**	**C&W (FB)**	**C&W (ART)**
Analysis	dttl	dttl	sttl	dttl	sttl
Backdoors	dttl	dttl	dttl	dttl	dttl
DoS	dttl	dttl	dttl	dttl	sttl
Exploits	dttl	dttl	dttl	dttl	sttl
Fuzzers	dttl	dttl	sttl	dttl	dttl
Generic	sttl	sttl	sttl	sttl	sttl
Reconnaissance	dttl	dttl	dttl	dttl	dttl
Shellcode	dttl	sttl	sttl	sttl	sttl
Worms	sttl	dttl	sttl	-	-

Table A.5: Selected Features for Single Feature Perturbation of UNSW-NB15 (CNN)

	AE				
Class	**BIM**	**PGD**	**DeepFool**	**C&W (FB)**	**C&W (ART)**
Analysis	dttl	dttl	sttl	Dload	sttl
Backdoors	dttl	dttl	dttl	dttl	dttl
DoS	sttl	sttl	sttl	sttl	sttl
Exploits	dttl	dttl	dttl	dttl	dttl
Fuzzers	dttl	sttl	sttl	dttl	sttl
Generic	sttl	sttl	sttl	sttl	sttl
Reconnaissance	dttl	dttl	dttl	dttl	dttl
Shellcode	sttl	sttl	sttl	sttl	sttl
Worms	-	sttl	-	-	-

Table A.6: Selected Features for Single Feature Perturbation of UNSW-NB15 (AE)

	AE				
Class	**BIM**	**PGD**	**DeepFool**	**C&W (FB)**	**C&W (ART)**
Analysis	dttl	dttl	dttl	dttl	dttl
Backdoors	dttl	dttl	dttl	dttl	-
DoS	dttl	sttl	sttl	sttl	sttl
Exploits	dttl	dttl	dttl	dttl	sttl
Fuzzers	sttl	sttl	sttl	sttl	sttl
Generic	Sintpkt	Sintpkt	smeansz	smeansz	smeansz
Reconnaissance	dttl	dttl	dttl	dttl	dttl
Shellcode	dttl	sttl	sttl	dur	sttl
Worms	sttl	dttl	Sload	-	Sload

Table A.7: Selected Features for Single Feature Perturbation of CIC-IDS2017 (DNN)

	DNN				
Class	**BIM**	**PGD**	**DeepFool**	**C&W (FB)**	**C&W (ART)**
Bot	Flow Packets/s	RST Flag Count	Flow Packets/s	Flow Packets/s	Init_Win_bytes_backward
DDoS	Active Std	Active Std	Active Std	min_seg_size_forward	Fwd Packet Length Max
DoS GoldenEye	Bwd Packets/s	Fwd Packet Length Min	Bwd Packets/s	Bwd Packets/s	min_seg_size_forward
DoS Hulk	Fwd PSH Flags	Fwd PSH Flags	Fwd PSH Flags	Bwd Packets/s	Bwd Packet Length Max
DoS Slowhttptest	min_seg_size_forward	min_seg_size_forward	Bwd Packets/s	Flow IAT Min	Flow IAT Min
DoS slowloris	RST Flag Count	RST Flag Count	RST Flag Count	min_seg_size_forward	Init_Win_bytes_backward
FTPPatator	RST Flag Count	RST Flag Count	RST Flag Count	Flow Duration	Fwd Packet Length Max
PortScan	RST Flag Count	RST Flag Count	RST Flag Count	PSH Flag Count	Flow IAT Min
SSHPatator	min_seg_size_forward	RST Flag Count	min_seg_size_forward	Flow Packets/s	Flow Duration
Web Attack Brute Force	Flow Duration	RST Flag Count	Flow Duration	Flow Duration	Fwd IAT Min
Web Attack XSS	Flow Duration	Flow Duration	Bwd IAT Std	Flow Duration	Flow Duration

Table A.8: Selected Features for Single Feature Perturbation of CIC-IDS2017 (CNN)

	CNN				
Class	**BIM**	**PGD**	**DeepFool**	**C&W (FB)**	**C&W (ART)**
Bot	Bwd Packets/s	URG Flag Count	RST Flag Count	Bwd Packets/s	min_seg_size_forward
DDoS	Fwd Packet Length Min	Fwd Packet Length Min	Bwd Packet Length Max	Bwd Packet Length Max	Total Fwd Packets
DoS GoldenEye	min_seg_size_forward	Bwd Packets/s	Bwd Packets/s	Bwd Packets/s	min_seg_size_forward
DoS Hulk	Bwd Packet Length Max	Bwd Packet Length Max	Bwd Packet Length Max	Bwd Packet Length Max	Init_Win_bytes_backward
DoS Slowhttptest	Bwd Packets/s	Bwd Packets/s	Bwd Packets/s	PSH Flag Count	Fwd IAT Min
DoS slowloris	PSH Flag Count	PSH Flag Count	Bwd Packet Length Max	PSH Flag Count	Bwd Packet Length Max
FTPPatator	Init_Win_bytes_backward	Active Mean	Flow Bytes/s_infinite	min_seg_size_forward	Fwd Packet Length Max
PortScan	PSH Flag Count	PSH Flag Count	URG Flag Count	PSH Flag Count	Bwd Packets/s
SSHPatator	Flow Duration	Active Std	min_seg_size_forward	Flow Duration	Flow Bytes/s
Web Attack Brute Force	min_seg_size_forward	PSH Flag Count	min_seg_size_forward	min_seg_size_forward	Fwd IAT Min
Web Attack XSS	Bwd Packets/s	Flow Duration	Active Mean	PSH Flag Count	Flow Duration

Table A.9: Selected Features for Single Feature Perturbation of CIC-IDS2017 (AE)

	AE				
Class	**BIM**	**PGD**	**DeepFool**	**C&W (FB)**	**C&W (ART)**
Bot	Bwd Packet Length Max	min_seg_size_forward	Init_Win_bytes_backward	Flow Bytes/s	min_seg_size_forward
DDoS	Init_Win_bytes_backward	Down/Up Ratio	Fwd Packet Length Max	Fwd Packet Length Max	Fwd Packet Length Max
DoS GoldenEye	Flow Duration	Flow Duration	Flow Duration	Flow Duration	Flow Duration
DoS Hulk	Flow Packets/s	Flow Packets/s	Flow Packets/s	Flow Packets/s	Flow Packets/s
DoS Slowhttptest	Flow Duration	Bwd IAT Std	min_seg_size_forward	Flow Duration	Flow Duration
DoS slowloris	Flow Duration	Init_Win_bytes_backward	Bwd Packet Length Max	Flow Duration	Flow Duration
FTPPatator	Flow Duration	Init_Win_bytes_backward	Init_Win_bytes_backward	Flow Duration	Init_Win_bytes_backward
PortScan	Bwd Packets/s	Bwd Packets/s	min_seg_size_forward	Bwd Packets/s	Fwd Packet Length Min
SSHPatator	Flow Bytes/s	Flow Duration	Flow Bytes/s	Flow Bytes/s	Flow Bytes/s
Web Attack Brute Force	Flow Duration	Init_Win_bytes_backward	Init_Win_bytes_backward	Flow Duration	Flow Duration
Web Attack XSS	Flow Duration	Flow Duration	Flow Duration	Flow Duration	Flow Duration

Table A.10: Selected Features for Single Feature Perturbation of CSE-CIC-IDS2018 (DNN)

	DNN				
Class	**BIM**	**PGD**	**DeepFool**	**C&W (FB)**	**C&W (ART)**
Bot	Bwd IAT Tot	Fwd Pkt Len Min	Fwd Pkt Len Min	Fwd Seg Size Min	Down/Up Ratio
Brute Force Web	Bwd Pkt Len Mean	Fwd Pkt Len Max	Flow Byts/s	Bwd Pkt Len Mean	Init Fwd Win Byts
Brute Force XSS	Tot Bwd Pkts	Flow Byts/s	Tot Bwd Pkts	Flow Duration	-
DDOS attackHOIC	Fwd Pkt Len Min	Init Bwd Win Byts	Fwd Pkt Len Min	Init Fwd Win Byts	Init Fwd Win Byts
DDOS attackLOICUDP	Bwd IAT Min	Bwd IAT Min	Tot Fwd Pkts	Tot Fwd Pkts	Tot Fwd Pkts
DDoS attacksLOICHTTP	Fwd Pkt Len Max	Init Fwd Win Byts	Tot Bwd Pkts	Flow Duration	Fwd Seg Size Min
DoS attacksGoldenEye	URG Flag Cnt	Down/Up Ratio	Fwd Pkt Len Min	Init Fwd Win Byts	Init Fwd Win Byts
DoS attacksHulk	Flow Byts/s	Flow Byts/s	RST Flag Cnt	Fwd Seg Size Min	Fwd Seg Size Min
DoS attacksSlowloris	Down/Up Ratio	Fwd Pkt Len Max	Fwd Seg Size Min	Init Fwd Win Byts	Init Fwd Win Byts
Infilteration	Fwd PSH Flags	Fwd PSH Flags	Bwd Pkt Len Mean	Init Fwd Win Byts	Init Fwd Win Byts
SQL Injection	Fwd Pkt Len Max	Fwd Pkt Len Max	Fwd Pkt Len Max	Fwd Pkt Len Max	Fwd Pkt Len Max
SSHBruteforce	Down/Up Ratio	Down/Up Ratio	Fwd Pkt Len Max	Fwd Seg Size Min	Init Fwd Win Byts

Table A.11: Selected Features for Single Feature Perturbation of CSE-CIC-IDS2018 (CNN)

	DNN				
Class	**BIM**	**PGD**	**DeepFool**	**C&W (FB)**	**C&W (ART)**
Bot	Bwd IAT Tot	Bwd IAT Tot	Flow Pkts/s	RST Flag Cnt	Init Bwd Win Byts
Brute Force Web	Fwd Pkt Len Max	Fwd Pkt Len Max	Init Bwd Win Byts	Flow Duration	Fwd Pkt Len Max
Brute Force XSS	Fwd Pkt Len Min	Fwd PSH Flags	Init Fwd Win Byts	Fwd Pkt Len Max	Init Fwd Win Byts
DDOS attackHOIC	Fwd Pkt Len Max	Bwd IAT Tot	FIN Flag Cnt	Fwd Pkt Len Max	Init Fwd Win Byts
DDOS attackLOICUDP	-	-	Flow Duration	-	Fwd Pkt Len Min
DDoS attacksLOICHTTP	Flow Byts/s_infinite	Init Fwd Win Byts	Flow Duration	Fwd Seg Size Min	Init Fwd Win Byts
DoS attacksGoldenEye	Fwd Pkt Len Max	Init Fwd Win Byts	Init Fwd Win Byts	Fwd Seg Size Min	Init Fwd Win Byts
DoS attacksHulk	Tot Bwd Pkts	Fwd URG Flags	Init Fwd Win Byts	Fwd Seg Size Min	Fwd Seg Size Min
DoS attacksSlowloris	Init Fwd Win Byts	Init Fwd Win Byts	Fwd Seg Size Min	Init Fwd Win Byts	Fwd Seg Size Min
Infilteration	RST Flag Cnt	RST Flag Cnt	Init Fwd Win Byts	Init Fwd Win Byts	Init Fwd Win Byts
SQL Injection	Init Fwd Win Byts	Fwd Pkt Len Max	Fwd Pkt Len Max	Fwd Seg Size Min	Fwd Pkt Len Max
SSHBruteforce	Flow Duration	Flow Duration	Flow Duration	Init Fwd Win Byts	-

Table A.12: Selected Features for Single Feature Perturbation of CSE-CIC-IDS2018 (AE)

	DNN				
Class	**BIM**	**PGD**	**DeepFool**	**C&W (FB)**	**C&W (ART)**
Bot	Fwd Pkt Len Max	Fwd Pkt Len Max	Tot Fwd Pkts	Fwd Seg Size Min	Bwd Pkt Len Mean
Brute Force Web	URG Flag Cnt	Fwd Pkt Len Min	Bwd Pkt Len Mean	Init Fwd Win Byts	Fwd Pkt Len Min
Brute Force XSS	Fwd Pkt Len Min	Fwd Pkt Len Min	Fwd Pkt Len Min	Flow Duration	Bwd IAT Tot
DDOS attackHOIC	Bwd IAT Min	Flow Duration	Bwd Pkt Len Mean	Init Fwd Win Byts	Init Fwd Win Byts
DDOS attackLOICUDP	Flow Duration	Flow Duration	Flow Duration	Flow Duration	Fwd Pkt Len Max
DDoS attacksLOICHTTP	Bwd Pkt Len Mean	FIN Flag Cnt	Fwd Pkt Len Min	Init Fwd Win Byts	Fwd Seg Size Min
DoS attacksGoldenEye	FIN Flag Cnt	Fwd PSH Flags	Fwd Seg Size Min	Fwd Seg Size Min	Init Fwd Win Byts
DoS attacksHulk	Init Bwd Win Byts	Init Bwd Win Byts	Fwd Seg Size Min	Fwd Seg Size Min	Fwd Seg Size Min
DoS attacksSlowloris	Active Mean	Active Mean	Bwd Pkt Len Mean	Fwd Seg Size Min	Fwd Seg Size Min
Infilteration	Fwd Seg Size Min	Fwd Seg Size Min	Fwd Seg Size Min	Fwd Seg Size Min	Fwd Seg Size Min
SQL Injection	-	-	-	-	-
SSHBruteforce	Flow Byts/s	Init Fwd Win Byts	Flow Byts/s	Flow Duration	Init Fwd Win Byts