Subverting Network Intrusion Detection: Crafting Adversarial Examples Accounting for Domain-Specific Constraints

Martin Teuffenbach(✉), Ewa Piatkowska(✉), and Paul Smith

AIT Austrian Institute of Technology, Vienna, Austria
{martin.teuffenbach,ewa.piatkowska,paul.smith}@ait.ac.at

Abstract. Deep Learning (DL) algorithms are being applied to network intrusion detection, as they can outperform other methods in terms of computational efficiency and accuracy. However, these algorithms have recently been found to be vulnerable to adversarial examples – inputs that are crafted with the intent of causing a Deep Neural Network (DNN) to misclassify with high confidence. Although a significant amount of work has been done to find robust defence techniques against adversarial examples, they still pose a potential risk. The majority of the proposed attack and defence strategies are tailored to the computer vision domain, in which adversarial examples were first found. In this paper, we consider this issue in the Network Intrusion Detection System (NIDS) domain and extend existing adversarial example crafting algorithms to account for the domain-specific constraints in the feature space. We propose to incorporate information about the difficulty of feature manipulation directly in the optimization function. Additionally, we define a novel measure for attack cost and include it in the assessment of the robustness of DL algorithms. We validate our approach on two benchmark datasets and demonstrate successful attacks against state-of-the-art DL network intrusion detection algorithms.

Keywords: Network intrusion detection · Deep neural networks · Adversarial examples · Adversarial robustness

1 Introduction

A Network Intrusion Detection System (NIDS) can be used to monitor network traffic and detect suspicious patterns that could be part of an attack. To address the challenges of modern network architectures, researchers have investigated the use of machine learning techniques for network intrusion detection [5,14]. In particular, Deep Learning (DL) methods, such as Deep Neural Networks (DNNs), are receiving substantial interest as they outperform shallow networks in accuracy. Several DL algorithms have been proposed for network intrusion detection, some reaching an average accuracy score up to 99% for specific datasets [11,20].

Published by Springer Nature Switzerland AG 2020
A. Holzinger et al. (Eds.): CD-MAKE 2020, LNCS 12279, pp. 301–320, 2020.
https://doi.org/10.1007/978-3-030-57321-8_17

However, DL algorithms have been found to be vulnerable to so-called *adversarial examples* [21] – inputs that are crafted to cause a misclassification. A great deal of research on this issue has been performed for DL algorithms that are used for image recognition. The aim is to introduce a small perturbation, imperceptible to a human, that causes an algorithm to misclassify an image. In a white-box setting, in which the architecture and gradients of a model are known, adversarial example attacks reach misclassification rates that are close to 100%. Considering the use of DL in computer vision for self-driving cars, adversarial examples are a serious threat, as they can cause misclassification of street signs.

In this work, we consider the concept of adversarial examples in the NIDS domain. Unlike in computer vision, in which features are independent from each other and can be changed somewhat arbitrarily, network-flow data has underlying compliance constraints [9]. Nevertheless, we show how an adversary can craft adversarial examples, even when constraints are placed on how network flow features can be manipulated. We reproduce state-of-the-art DL models and formulate restrictions for attacking them in a NIDS setting. To ensure comparability over different datasets and models, we provide a rule-set for the grouping of features. Building on this, we derive a novel crafting algorithm that preserves compliance with these rules. An evaluation has been performed that examines how effective these attacks are under a given budget. We demonstrate that by perturbing less than 10% of the features used by the classifier by less than an average of 0.2 we can achieve up to 100% success rate for an attack.

The remainder of this paper is structured as follows: In Sect. 2, we review state-of-the art models, attacks and metrics on the topic of adversarial examples. Section 3 describes the models that we have used in our studies and a threat model. In Sect. 4, we propose our approach to attack the models and the metrics we will use for evaluation; in Sect. 5, we present the results of our attacks.

2 Related Work

Initial research that investigated adversarial examples for NIDSs, e.g. the work by Yang *et al.* [24], often considered network traffic datasets arbitrarily – i.e. they do not constrain the perturbations that can be introduced to craft an adversarial example. Consequently, adversarial capabilities [2] – constraints on permitted perturbations – were not thoroughly considered for most of the proposed attacks. This is appropriate for attacking an image-based input, but not for network traffic-based input – unlike pixels, which can take any value from 0–255 and be independently changed, features in network flow-datasets contain dependencies and have compliance constraints. Recently, Zhang *et al.* [25] investigated a reinforcement learning approach to match IDS dataset-specific restrictions. The action space of their algorithm only contains valid actions, meaning those actions that would not degrade the validity of the instance. Hashemi *et al.* [9] introduced the idea of treating features differently, based on their properties, when crafting adversarial examples for flow-based NIDSs. They also take dependencies between features into account. We have extended this research by

assigning weights to feature groupings, which reflect perturbation constraints, and incorporating these weights into the optimization function that is used to craft the adversarial example.

There are several approaches to measuring the robustness of algorithms against adversarial examples. As the concept of adversarial examples for image recognition algorithms involves imperceptibility from the original image for a human observer, most robustness metrics use a distance metric, using the L1/L2 norm. This is appropriate in this domain, because the distance between the original image input and adversarial example is correlated with the visible difference. Moosavi *et al.* [15] define the expectation value of the *minimal perturbation* over a test dataset as a measure of robustness. Using a theoretical approach, Weng *et al.* [23] have developed the 'CLEVER' score, which is also an estimation for the minimal distance required to subvert a neural network classifier. Papernot *et al.* [16] have proposed the *adversarial distance*, which measures distances between different labels using gradient information to indicate the risk of misclassification between classes. Meanwhile, in the NIDS domain, Hartl *et al.* [8] have developed the *Adversarial Risk Score (ARS)*, which is a distance-based robustness score for classifiers against adversarial examples. In their work, they use a Recurrent Neural Network (RNN) for classification, and investigate the feature sensitivity of their classifier. We argue that without considering the properties of features, e.g. to what extent they can be manipulated, a distance-based approach for measuring the effectiveness of adversarial examples against NIDSs does not accurately reflect the threat. We will elaborate an empirical score for robustness that is based on the relative number of adversarial examples within a given (constrained) feature subspace.

3 Preliminaries

Based on previous research, we present a taxonomy for adversarial examples. We use this taxonomy to define the threat model that we have used in our work. Subsequently, we introduce network intrusion detection using DL algorithms and summarize the algorithms that we have used.

3.1 Adversarial Example Threat Model

The following taxonomy, which describes a threat model for adversarial examples, is adopted from previous work by Carlini *et al.* [2]. The threat model of an adversarial attack can be defined by the *adversarial goal*, *adversarial knowledge*, and *adversarial capabilities*. Based on this taxonomy, the *goal* of the adversary is to perform a network attack that is known (correct classification with high confidence) to the NIDS and change its features to remain undetected (a source/target misclassification problem). It is assumed that the adversary has *white-box knowledge*. Specifically, the type of attack that is performed is one contained in the training dataset of the NIDS, as these instances are known to the model and have higher accuracy. Furthermore, the adversary has full access

to the gradient information of the model. We consider two restrictions regarding *adversarial capabilities*: *(i)* the feature space is reduced to a subset of features (referred to as 'considered features') that an attacker can perturb; and *(ii)* a maximum distance δ_{max} to the original sample (L1-norm) is imposed. We use the L1-norm for the maximum distance constraint, as this norm reflects the average absolute change per feature. With this approach, we want to show the worst-case vulnerability of a given NIDS.

3.2 Deep Learning Algorithms

To get a representative sample of DL algorithms for our experiments, we use several state-of-the-art models: a Deep Neural Network (DNN) for binary and multi-class classification [18], in a supervised and semi-supervised fashion; a Deep Belief Network (DBN) for multi-class classification (semi-supervised) [6]; and an Auto Encoder (AE) that is trained to perform outlier detection for unsupervised anomaly detection [10].

Deep Neural Network. To extend the functionality of a regular DNN, which usually works as a supervised classifier, Rezvy *et al.* [18] proposed the idea to stack an Auto Encoder on top of a two-layer classifier network. This algorithm trains the model in three stages: *(i)* the AE is trained unsupervised; *(ii)* the classifier is trained with the output of the AE as input to perform a classification, in a supervised fashion; and *(iii)* the network as a whole is tuned with a few training samples, also supervised. We use this DNN to perform binary classification, to get a comparable result to the AE-based outlier detection (see below).

Deep Belief Network. A model with a similar approach to the AE-DNN is a Deep Belief Network, which is a perceptron network that consists of several layers of Restricted Boltzmann Machines (RBMs). RBMs are often used for dimensionality reduction or feature extraction, which is similar to the algorithm proposed by Gao *et al.* [6]. The model consists of several layers of RBMs with a single-layer classifier stacked on top. The RBMs are trained unsupervised, the classifier then supervised, using the output of the final RBM as input. The idea is that the RBMs in sequence reduce the input dimension to achieve a better representation of the features.

Auto Encoder. Hawkins *et al.* first proposed the use of an AE to perform outlier detection [10]. The approach is to train an AE on benign data (train the network to reproduce the input, unsupervised) and then use the reconstruction error to measure for outliers. The Outlier Factor (OF) of a record i is defined as the average of the squared reconstruction error. To get a more general outlier detection, we extended this concept with a method proposed by Azami *et al.*, which converts distance-based outlier detection methods to probabilities using a sigmoid function [1]:

$$P(\text{i is outlier}) = (1 + exp(\frac{\text{OF}_i - \gamma}{\sigma}))^{-1} \tag{1}$$

where γ is the anomaly threshold ($\mathrm{OF}_i \geq \gamma \Rightarrow \mathrm{P(i\ is\ outlier)} \geq 50\%$) and σ a scaling constant. We define the threshold as the 95-percentile and σ as the standard derivation of the outlier factors of the training dataset.

4 Methodology

The majority of work on adversarial examples has targeted computer vision algorithms – specifically, classifiers that take raw images as an input. Applying crafting algorithms that are designed for image processing to other domains might not be appropriate. There are several domain-specific aspects to be considered when crafting adversarial examples for NIDS. First, the features extracted from network traffic are heterogeneous, including flows, context, statistical and categorical features. The dependencies between features imposes additional limitations on how their values can be modified. Second, adversarial perturbations should be crafted considering the way the attack will be implemented, e.g. via the modification of network flows. Therefore, perturbations should render correct flows and ensure compliance with protocols. Third, the concept of *imperceptibility*, which is applicable in the image domain, does not readily apply in the NIDS domain. Nevertheless, minimising perturbation may help to ensure that the original malicious intent of the network attack is preserved. In addition, restrictions on perturbation magnitude could also be important to remain stealthy, i.e. introducing high perturbation to some features might trigger alerts. For instance, increasing packet numbers to hide one attack could result in it being detected as another, either by a NIDS or complementary anomaly detection algorithms.

In the following, we present how we address these aspects. Based on the properties of the features, we divide them into groups (as in [9]) and incorporate feature constraints in the adversarial crafting algorithm. We propose weighted optimisation for crafting adversarial examples in order to ensure feature constraints. As with previous work, we apply perturbation constraints to remain stealthy.

4.1 Feature Analysis and Grouping

Hashemi *et al.* [9] proposed the idea of grouping flow-based features by their *feasibility*, i.e. grouping flows based on whether they can be modified by an adversary and still yield correct flows. We follow a similar approach and group features, as presented in Table 1.

We exclude features that are related to backward flows, as it is reasonable to assume that an adversary generally does not have control over traffic that is generated by other hosts in the network (unless they are compromised). Features that are derived from other features are also not considered. These two groups are merged into a group (G0). Next, we consider features that are independent, e.g. the total number of forwarded packets or maximum length of the forwarded packets. These are further divided into features that are not used to derive other features (G1) and those which are (G2). Changes to features of the latter group

requires recalculating dependent features; therefore, we consider them harder to change. The last group contains features that are difficult or impossible to change directly. For instance, to influence statistical summaries of the flows requires the modification of multiple packets. The last group (G3) also includes features with underlying physical constraints (e.g. Inter-Arrival-Time (IAT) features), whose modification could cause potential violations and break the communication.

Table 1. The proposed feature grouping, including those from Hashemi *et al.* [9]

Group	[9]	Description	Weight w
G0	(1)	Features extracted from backward flows	0
	(3)	Features whose values depend on the other features and can be calculated directly by a set of them	0
G1	(2)	Independent and not used to derive other features	1
G2	(2)	Independent and used to derive other features	2
G3	(4)	Features dependent on batches of packets (e.g., mean and frequency based features)	3
		Features with underlying physical constraints (e.g., IAT)	3

Based on the proposed grouping of features, we assign a weight w to each of the groups (as shown in Table 1). The weights give an intuition about how difficult it is for an adversary to perturb a feature, wherein $w = 0$ denotes that the feature cannot be changed and $w = 1, 2, 3$ indicates increasing difficulty. The crafting algorithm is intended to favour features with $w = 1$ over higher weights.

These weights are used in our crafting algorithm. They can be assigned to groups of features or individually to each feature. In this work, we have used constraints that are imposed by the implementation of adversarial perturbation through changes in network flows. Assuming a different threat model, these constraints can differ. For example, weights could also reflect the risk – to the adversary – associated with their change (e.g. due to complementary security measures). Moreover, weights can be assigned empirically using a feature sensitivity analysis [8].

4.2 Crafting Algorithm

For crafting adversarial examples, we extend the Carlini and Wagner (C&W) attack [3]. This method is among the most powerful crafting algorithms and a benchmark technique to evaluate the robustness of deep learning algorithms. Carlini and Wagner formulated an optimisation-based approach to craft adversarial examples. They derived an objective function that maximizes the desired target prediction, while minimising the size of the perturbation:

$$\min_{\delta}(||\delta||_p) \; + \; c \cdot g(x+\delta) \quad \text{s.t. } x+\delta = x^* \in [0,1]^n, \tag{2}$$

where $\delta = x - x^*$ is the distance between the input x and the adversarial sample x^*, c is a coupling constant and $g(x)$ is a target function. This approach ensures minimal perturbation while minimizing the desired target function. For the L_2 (i.e. Euclidean) norm attack, p is set to 2. The constant c in this equation links the minimization of the distance with the minimization of the target function – smaller values of c result in a bias toward minimizing the distance. The optimisation is solved with the Adam algorithm [12].

As this crafting algorithm performs a distance minimization, it is possible to extend it with our proposed weighted features approach. Features are assigned weights w according to their grouping (0 to 3). Using these weights, along with a mask that specifies the set of features that are considered by the algorithm, we extend the C&W attack, as follows:

$$\min_{\delta}(||\delta \odot w||_2) \; + \; c \cdot g(x + \delta \odot mask) \quad \text{s.t. } x + \delta = x^* \in [0,1]^n, \tag{3}$$

where $\odot$ indicates an element-wise vector-vector multiplication. The weights added to the distance in Eq. 3 forces the algorithm to favour modifications on low-weighted features and avoid adding too much perturbation to high-weighted features. The *mask* represents our restriction on the feature space, e.g. it sets the perturbation of undesired features to 0 (in our case features from G0). As the function $g(x)$ in Eq. 3, we used the f_5 function, as presented in [3]:

$$g(x) = log(2 - 2 \cdot F(x)_t), \tag{4}$$

where $F(x)_t$ is the prediction of the classification model of target label t. This function becomes minimal when $F(x)_t \approx 1$.

4.3 Attack Budget (Parameters) and Metrics

The proposed crafting algorithm performs optimisation with respect to two types of constraints: feature space (feature budget) and the magnitude of feature change (perturbation budget). We define a feature budget f, based on the *mask* and w, and calculate it as follows:

$$f = \frac{mask \cdot w}{|w|_1} \tag{5}$$

where $mask \cdot w$ denotes a vector-vector product. If all features of groups G1-3 are used, the value of f is equal to 1, whereas with fewer features it decreases. This value should give an indication of the cost of an attack. Therefore, if an adversary can craft adversarial example with only a few, low-weight features, f is close to 0, hence the cost of the attack is low. Alongside the reduction on the feature space, we restrict our attacks with a maximum perturbation δ_{max}.

To assess the strength of the attack for a given budget f and δ_{max}, we compute the success rate as follows:

$$s_{j \to t} = \frac{1}{N} \sum_{x \in D_j} \mathbb{1}_{F(x+\delta_x)=t} \quad \text{s.t. } \forall \delta_x : \bar{\delta_x} \leq \delta_{max}, \tag{6}$$

where F is a given classifier with $F : D \rightarrow Y$ (Y being a set of labels), t is the desired label ($t \in Y$) and D_j is the data distribution of instances with true and predicted label j ($D_j \subset D$ with $F(x) = j$ for all $x \in D_j$ and $|D_j| = N$). $\mathbb{1}_{\text{condition}}$ is an indicator function, which is 1 if the condition is true, 0 otherwise. The perturbation δ_x, which aims to turn x into an adversarial example is restricted with the *mask*. The perturbation budget (δ_{max}) restricts the average perturbation per feature $\bar{\delta_x}$ of each instance x. As we only consider attacks that aim to be stealthy (target label t is benign), we are going to denote $s_{j \rightarrow t}$ as s_j. The overall success rate s would then be the average over all labels $j \in Y$. The success rate only makes sense for a sufficiently large number of test samples N.

Goodfellow *et al.* [7] interpreted adversarial examples as 'blind spots' due to incomplete training data. Using this metaphor the success rate can intuitively be seen as the relative number of detected 'blind-spots' within a sphere with radius δ_{max} in a hyperspace defined by *mask*. The success rate is dependent on the technique used for crafting adversarial examples and the set of input samples. Therefore, we restrict our evaluation to the *empirical success rate*, which is the overall success rate for a given crafting algorithm that is tested on particular set of samples.

To give an intuition about the vulnerability of the deep learning model against a given attack, we introduce a *vulnerability score*:

$$vs = \frac{2 \cdot (1/f) \cdot s}{(1/f) + s}, \tag{7}$$

which is the harmonic mean between the success rate s and the inverse feature budget f. The vs metric takes values in the range $[0, 2]$. The closer vs is to 0, the less vulnerable is the model under test. This metric is intended to reflect the trade-off between the empirical success rate s and the hyperspace defined by the attack budget.

5 Experimental Analysis

In this section, we introduce the experimental setup. We explain and justify the choice of datasets and how they are used for both anomaly detection (binary) and multi-label classification. Furthermore, we present our approach to preprocessing and feature grouping on the datasets. Subsequently, the models are evaluated (their accuracy) and then attacks on those models, to compare their vulnerability (robustness) against the adversarial examples, are performed.

5.1 Datasets and Preprocessing

In our experiments, we use two publicly available IDS datasets: *(i)* NSL-KDD [22], a relatively old but still widely-used benchmark dataset; and *(ii)* CICIDS2017 [19], which is a more recent network flow-based dataset.

Datasets. The NSL-KDD dataset is a refined version of the well-known KDD CUP 99 dataset that was developed by DARPA. The dataset includes information that has been extracted from network traffic (e.g. TCP connections), as well as high-level features (e.g. the number of failed logins). The data records are attributed with 41 features (intrinsic, content, time-based and host-based). The dataset contains 49 different labels that are grouped into five categories: Normal, Denial of Service (DoS), Probing, User to Root (U2R) and Remote to Local (R2L) attack. The NSL-KDD dataset is divided into training and testing sets. The distribution of the attacks in each of the subsets are presented in Fig. 1.

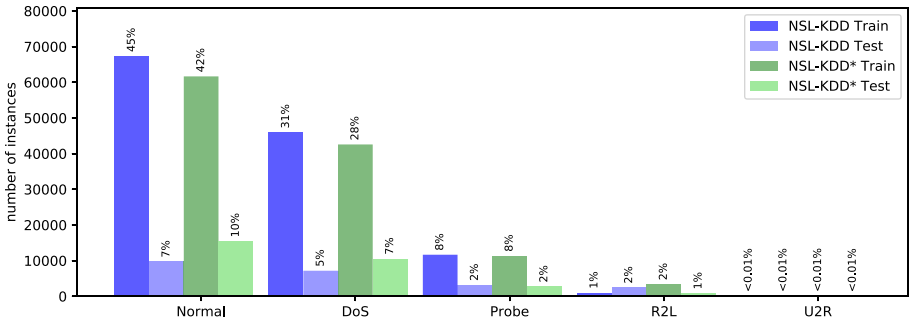


Fig. 1. The distribution of samples in the NSL-KDD dataset

A challenge when using the NSL-KDD dataset is the high diversity of testing samples – the testing set contains labels that are not included in the training set. In addition, for U2R and R2L attacks, the training set is smaller than the testing set, which can have a negative impact on the accuracy of trained classifiers. To address this issue, Rezvy *et al.* [18] defined their own training and testing subsets (further referred to as NSL-KDD*) to train their classifier with more samples of U2R and R2L attacks (see Fig. 1). In our experiments, we aim to reproduce the results of the Rezvy *et al.* DNN model; therefore, we use both variants of the NSL-KDD dataset.

The second dataset (CICIDS2017) [19] contains five days of network traffic that were recorded in an emulated environment with 25 hosts. The data is provided as network packets and bidirectional network flows. In total, this set contains about 2.8 million records, which are described with 80 features and grouped into 15 label categories (14 attacks plus benign). The dataset is quite imbalanced – some attacks are very sparse, e.g. the *Heartbleed* attack appears only eleven times in the whole dataset, whereas almost 84% of labels are normal samples. In our experiments, we used a subset of the CICIDS2017 dataset (due to computational constraints). We used an 80/20 random split to create training and testing sets. For multi-label classification, we utilized an oversampling technique for the training phase, by adding redundant data to low-frequency labels.

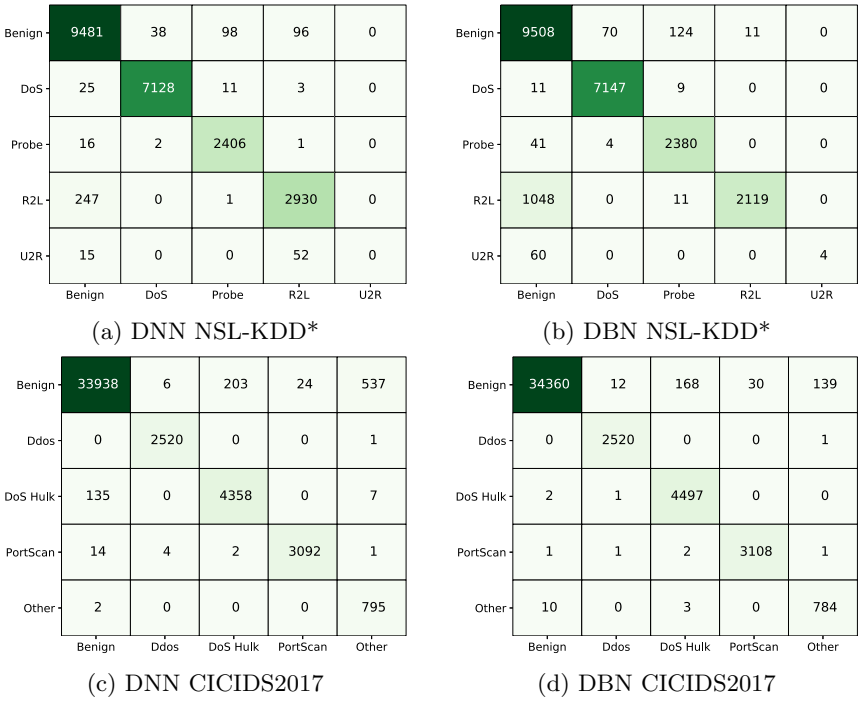
We also removed random samples from the benign label (undersampling). For the anomaly-based detection, we kept the original distribution of the CICIDS2017 dataset, as the benign and total attack samples are sufficiently balanced.

Data Preprocessing. Similar to Gao *et al.* [6], we have applied three preprocessing methods; these are necessary for the datasets to be used as input to machine learning models. First, we have performed *one-hot-encoding* to map categorical features to numerical values. For instance, the 'Protocol' feature can be of three distinct categories, namely 'TCP', 'UDP' or 'ICMP'. In one-hot-encoding, this feature is represented by a 3×1 vector, in which the value 'TCP' protocol translates to [1, 0, 0], 'UDP' to [0, 1, 0], etc. This procedure has been applied to the NSL-KDD dataset, encoding 41 categorical features as 122 numerical features. This step was not necessary for the CICIDS2017 dataset, as it does not include categorical features. Second, since the range of the features in the IDS datasets may vary significantly (e.g. some packet-size related features range from 0 to 10^8, others from 0 to 10^3), we apply the following log-transformation to all features: x_i: $x_i = log(1 + x_i)$. Consequently, a value $x_i \in [0, 10^8]$ is transformed to $x_i \in [0, 18]$. This approach was applied by Hawkins *et al.* [10]. Finally, we normalize the data using a *min-max transform*, such that the range of all the features is set to $[0, 1]$. Without the log-transformation that is performed in the previous step, outliers would cause the majority of features to be set to zero.

Feature Groups. The features in the datasets were arranged into groups, as discussed in Sect. 4.1. For the CICIDS2017 dataset, we assigned all the flows in a backward direction to G0, along with features that relate to both backward and forward directions (total flow bytes/s). Groups 1, 2 and 3 are features in the forward direction. The independent features are placed in G1, those that are used to derive other features are placed in G2, and derived features in G3. Additionally 'hard to access' features and those with physical limitations (IAT features) are placed into G3. Meanwhile, for the NSL-KDD dataset, we put all the categorical and binary features (flags) into G0. Content-based features that are based on flows in the forward direction are located in G1, alongside (independent) counters. Group 2 contains counters that are used to derive other features. Finally, frequency-based features (e.g. error rates) are placed into G3. We consider 30 out of 122 (25%) features for the NSL-KDD set and 32 out of 78 (41%) for the CICIDS2017 as accessible features, i.e. features that could be manipulated by an adversary. Therefore, the maximum number of $f = 1$ in Eq. 5 can only be achieved by using all 30 and 32 accessible features for NSL-KDD and CICIDS2017 datasets, respectively.

5.2 NIDS Model Evaluation

We start our experiments by establishing the baseline performance of the considered algorithms for network intrusion detection (described in Sect. 3.2). We evaluate both variants of NIDS models: attack detection (multi-label classification models) and anomaly detection (binary classification models).



(a) DNN NSL-KDD* (b) DBN NSL-KDD*

(c) DNN CICIDS2017 (d) DBN CICIDS2017

Fig. 2. The confusion matrix for the multi-class models

Attack Detection (Multi-Class Models). Two classifiers have been trained and tested: a Deep Neural Network (DNN) [18] and a Deep Belief Network (DBN) [6]. The architecture and training parameters were chosen to be similar to the proposed models, with slight modifications to optimize the performance. Table 2 shows the performance of the models in terms of overall Accuracy, Recall, Precision and F1-score. The performance of both of the models are acceptable and aligned with the original papers [6,18]. It can also be noted that, as expected, both models achieve significantly better accuracy on the NSL-KDD* dataset than NSL-KDD.

Table 2. The results of the attack detection models

Model	Dataset	Accuracy	Precision	Recall	F1
DNN	CICIDS2017	0.979	0.933	0.983	0.957
	NSL-KDD	0.742	0.914	0.603	0.777
	NSL-KDD*	0.973	0.982	0.971	0.976
DBN	CICIDS2017	0.991	0.969	0.996	0.982
	NSL-KDD	0.762	0.929	0.630	0.751
	NSL-KDD*	0.938	0.983	0.908	0.944

Figure 2 depicts the confusion matrix of the multi-label classification models. For the NSL-KDD* dataset, both classifiers show poor performance for the U2R instances, most likely due to the low frequency of that attack. The majority of the other labels yield high numbers of true positives. The confusion matrix for CICIDS2017 includes details for the most frequent labels (i.e. Benign, DDos, DoS Hulk and PortScan) and summarizes the rest as 'Other'. Both models appear to perform best on the DDoS and PortScan attacks.

Anomaly Detection (Binary Models). For anomaly detection, the DNN model [18] was used again. However, in this case, it was trained for binary classification, i.e. there are only two output classes: benign or attack. The DNN was trained on the NSL-KDD, NSL-KDD* and original subset of the CICIDS2017 (without oversampling) datasets. Table 3 compares the semi-supervised DNN anomaly detection with the unsupervised AutoEncoder (AE) model, which was trained only on the set of benign samples from NSL-KDD and CICIDS2017. The AE threshold was set to 80%.

The binary DNN achieves slightly better results than the multi-class DNN for all of the datasets. The improvement is especially visible for the original split of the NSL-KDD dataset, which demonstrates that supervised classification is very sensitive to the distribution of labels in the training set. The AE model achieves surprisingly good results for the NSL-KDD data, with precision and recall above 0.88; however, its accuracy on the CICIDS2017 data is drastically lower. The recall of attack detection is just slightly above 50% and low precision indicates a high number of false positives. The reconstruction error of the anomalous instances seems to be insufficient to efficiently differentiate attacks from benign samples. In our evaluation of the adversarial example attacks, we focus only on sufficiently accurate models; hence, we will not consider the AutoEncoder (AE) trained on the CICIDS2017 dataset.

Table 3. The results of the anomaly detection models

Model	Dataset	Accuracy	Precision	Recall	F1
DNN	CICIDS2017	0.990	0.965	0.996	0.980
	NSL-KDD	0.794	0.921	0.698	0.798
	NSL-KDD*	0.975	0.984	0.971	0.978
AE	NSL-KDD	0.870	0.890	0.880	0.885
	CICIDS2017	0.628	0.273	0.548	0.365

Our goal with these models is to reproduce a representation of the established deep learning algorithms and test their robustness against adversarial samples. The accuracy and F1-score in Tables 2 and 3 demonstrate that the classifiers are reliable. In further experiments, we will use the strongest variants of the classifiers – for the DNN and DBN models, we choose those trained on the

CICIDS2017 and NSL-KDD* datasets (as they yield higher accuracy than the ones trained on the original NSL-KDD dataset).

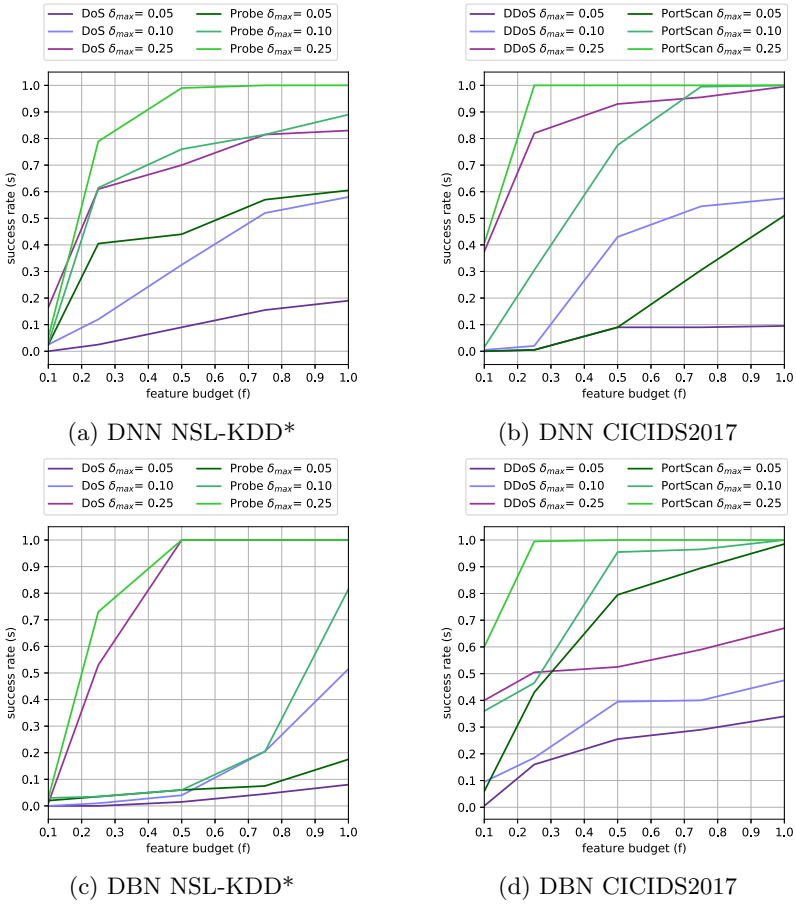
5.3 Attack Evaluation

In this section, we evaluate the proposed algorithm for crafting adversarial examples. As mentioned in Sect. 3.1, our threat model assumes a white-box attack, i.e. the architecture and parameters of the NIDS model are known. The attacker's goal is to launch a stealthy attack through the use of adversarial examples. The idea is to modify the attack sample in such way that a NIDS classifies it as benign. For the multi-label classifiers, we chose the top two attacks (labels with the highest accuracy), as indicated in Fig. 2. The best accuracy of the DNN and DBN models was reported for DoS (99.6%/99.6%) and Probe (99.4%/99.1%) attacks from the NSL-KDD dataset, and Distributed Denial of Service (DDoS) (99.1%/99.6%) and PortScan (99.8%/99.0%) from the CICIDS2017 dataset. These four labels are considered for crafting adversarial examples. For anomaly detection, we selected a random subset of the NSL-KDD attack samples. Some assumptions were made in the evaluation of the proposed adversarial example attacks:

- Adversarial examples are only crafted for inputs that are correctly classified with at least 80% confidence.
- The parameter c in Eq. 3, which is the trade-off between minimizing the perturbation (distance) and maximizing the loss function, is set using a binary search (see [3]).
- We always list the worst-case attack, thus if not stated explicitly, the parameters are always set to maximise the success rate.
- The perturbation constraint δ_{max} should be understood as the average distance per feature; therefore, the actual perturbation δ must be smaller than $\delta_{max} \cdot |mask|_1$.
- The list of *considered features*, formally defined by $mask$, specifies which features can be modified by the adversarial crafting algorithm. The $mask$ is created by randomly adding features from groups in the order of their weights (starting with G1 up to G3), until the budget is reached.
- If all features of G1 to G3 are considered then $f = 1$, whereas $f \approx 0$ indicates that only a few features from G1 were selected. For all of the classifiers, we used the same $mask$ to get comparable results.

In the following, we experimentally evaluate the strength of the adversarial example attacks for the considered deep learning algorithms under a given budget, expressed in terms of feature space constraints (f) and perturbation magnitude (δ_{max}).

Attacks on Multi-class Models. We measure the success rate (see Eq. 6) of the adversarial examples that are crafted by the proposed algorithm, using different combinations of attack parameters. Figure 3 depicts the highest achieved



(a) DNN NSL-KDD*

(b) DNN CICIDS2017

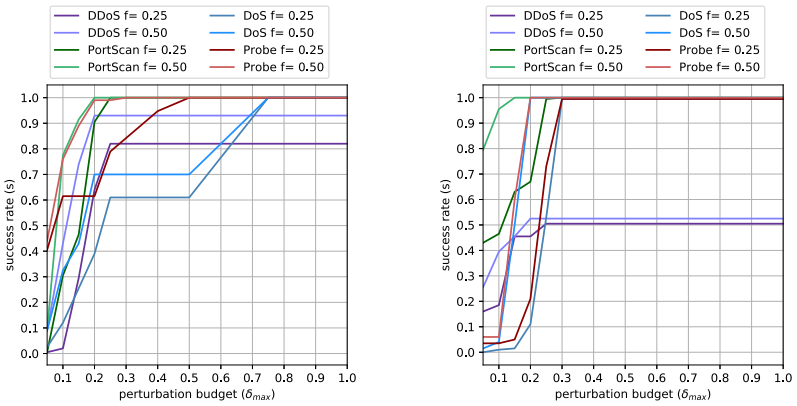
(c) DBN NSL-KDD*

(d) DBN CICIDS2017

Fig. 3. Evaluation of the attack parameters for multi-class models

success rate for the DNN and DBN models for $\delta_{max} = \{0.05, 0.1, 0.25\}$ against varying $f = \{0.1, 0.25, 0.5, 0.75, 1\}$. The overall success rate is still quite high, given the fact that we consider only 41% of the features in the CICIDS2017 dataset as accessible and 25% of the NSL-KDD dataset (see Sect. 5.1). The results indicate that it is more difficult to craft adversarial examples for DoS attacks. Surprisingly, the DBN model seems to be less vulnerable for low budget attacks than the DNN model for the NSL-KDD* set, even though the DNN model outperforms the former in detection (see Table 2). As expected, the success rate increases with higher values of f – the attacks are more effective if an adversary is allowed to manipulate more features. The plots (Fig. 3) reveal some intuition about a trade-off between the attack parameters. We can analyse for which δ_{max} an attacker can reach the optimum success rate with a small feature budget. For instance, the most successful attacks can be noted for PortScan

against both the DNN and DBN models with $\delta_{max} = 0.25$. Furthermore, we can observe that the success rate of the attack is more sensitive to changes of δ_{max} rather than f. Sometimes we can observe that with sufficient δ_{max} adding more features does not improve the success rate (e.g. for Probe and PortScan, the optimal feature budget can be found below 0.5). Analogically, for DoS and DDoS attacks, reasonable success rates can only be achieved for the highest δ_{max} value. For the majority of attacks, the optimal feature budget can be found below 0.5; afterwards there is only a gradual, small increase in success rate.

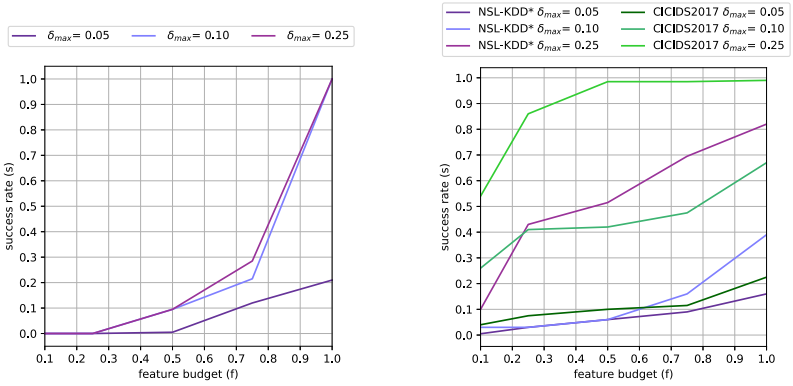


(a) DNN (b) DBN

Fig. 4. The success rate of attacks against the DNN and DBN models for different values of δ_{max} parameter

We have investigated different values of perturbation budget δ_{max} for $f = 0.25$ and $f = 0.5$. Figure 4 visualizes the influence of δ_{max} on the success rate. We observe that for most graphs there exists a certain threshold after which the success rate stays constant. For the DBN model, the optimal value of perturbation budget is reached around $\delta_{max} = 0.3$, whereas for DNN it is on average slightly higher $\delta_{max} = 0.5$. Overall, this threshold appears to be slightly higher for the NSL-KDD dataset. This is due to the fact that the relative amount of features we consider is larger for the CICIDS2017 dataset. Therefore, a smaller perturbation-per-feature is necessary to reach the adversarial goal. An average perturbation above 0.3 seems rather extensive; however, we managed to achieve considerable success rates for δ_{max} below this value with $f = 0.5$. This budget for the NSL-KDD* dataset, for example, means that we only perturb around 10% of the features by ≈ 0.1 and still find adversarial examples for many instances.

Attacks on Anomaly-Based (Binary) Models. We performed a similar parameter analysis for adversarial example attacks against binary models. We



(a) AutoEncoder (b) Binary DNN

Fig. 5. Evaluation of the attack parameters for binary models

recorded the highest achieved success rate for given perturbation restrictions. The results are depicted in Fig. 5.

The AE model treats anomalies as outliers and detects them by measuring the reconstruction error (see Sect. 3.2). As Fig. 5a suggests, a rather high budget is necessary to achieve the adversarial goal. Surprisingly, the feature budget (f) seems to have more influence on the success rate than perturbation budget (δ_{max}). Even though the AE achieves lower accuracy than the other (supervised) models, it has a significant advantage of being more robust to adversarial examples. In Fig. 5b the success rates of adversarial example attacks against the binary DNN model are presented. We can clearly observe that attacks using NSL-KDD* dataset show better performance than CICIDS2017. Furthermore, for the CICIDS2017 dataset it is possible to craft strong attacks even with a small feature budget and relatively small δ_{max}. This might indicate that the distance between benign and anomalous samples is rather low for that dataset, making it challenging for outlier-based anomaly detection. This is why the AE model performed so poorly on the CICIDS2017 dataset (as shown in Table 3).

Vulnerability Analysis. In Fig. 6, we compare the algorithms in terms of their vulnerability to adversarial examples. The vulnerability score vs (see Eq. 7) is listed per attack against each model. The experiments were performed for three different values of δ_{max}.

We can observe that vs confirms our findings from the previous experiments. The classification models are relatively robust against the DoS attack of the NSL-KDD dataset, but rather vulnerable to the Probe and PortScan attacks. The low vs for the DoS attack may be explained by the fact that the most discriminative features for this type of attack are the IAT and frequency-based features. However, these features are assigned to G3 in the proposed grouping, thus considered hard to access and modify by an attacker. Considering features

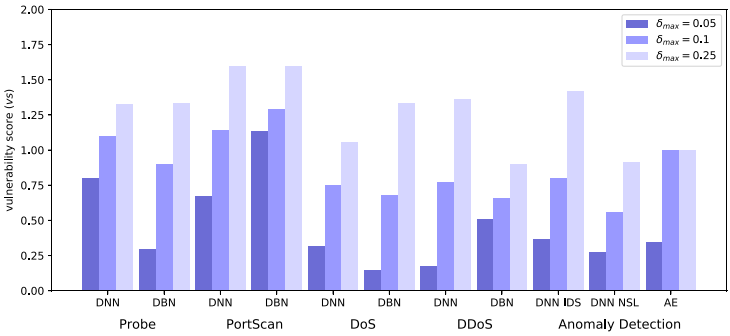


Fig. 6. The vulnerability score of all the models for both datasets

with high weights to craft successful adversarial examples targeted towards DoS attacks would increase the overall feature budget f and, therefore, decrease the vulnerability score. As previously shown, the AE model is robust against low-budget adversarial attacks. This is reflected by $vs \leq 1$ for all values of δ_{max}.

5.4 Feature Perturbation Analysis

To show the impact of the feature weights in the proposed algorithm (see Eq. 3), we performed a feature perturbation analysis. For this experiment, we used the DNN classifier trained on the CICIDS2017 dataset and crafted adversarial examples for the DDoS label. We compare the proposed crafting algorithm with the original C&W [4] and iterative Fast Gradient Sign Method (iFGSM) [13] attacks. The latter attack performs iterative FGSM [7], which maximises the loss using gradient sign. This algorithm terminates when either the desired label is reached or the limit of maximum iterations is exceeded; hence, it does not perform any distance (perturbation) minimisation. Therefore, we neglect the restriction on perturbation budget (maximum distance constraint) in the C&W and our method, since this restriction is not directly relevant for the analysis. As *considered features*, four features of each group have been chosen arbitrarily. The resulting feature budget sums up to $f = 0.31$. We measure the average perturbation per each feature for the successfully created adversarial examples. The results are depicted in Fig. 7.

As expected, the average perturbation for iFGSM and C&W does not depend on the feature group but feature importance with respect to the classifier. Furthermore, for C&W and our proposed extension, we compare the average perturbation per feature using two different values of the hyperparameter c, which controls the impact between distance and target function minimisation. A low value of $c = 0.1$, assigns less importance to target function minimisation and enforces a small distance over desired attack, hence the weights of the extended C&W algorithm strongly influence the results. As can be observed in Fig. 7 (left), our algorithm favours features from G1 over the other groups. This effect is less visible with higher $c = 10$, in which adversarial examples are optimised

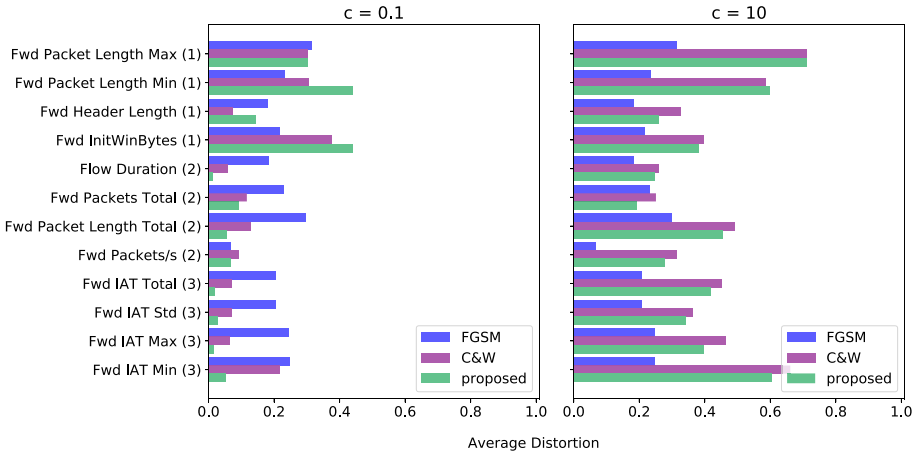


Fig. 7. The average distortion per feature for the DDoS attack label – feature group numbers are shown in parentheses

mostly with respect to target function. The success rate of the iFGSM in Fig. 7 was 73.3% (left and right). With emphasis on the distance minimization (left, $c = 0.1$) our proposed algorithm reached a success rate of 50.8 %, the original C&W 55.6%, even though the proposed algorithm applied only a perturbation close to 0 to the G3 features. By increasing c to 10, we achieved 87% for the proposed method and 87.7% for the C&W.

For an adversary that intends to attack a NIDS through adversarial examples it is vital to account for not only the feature space constraints (e.g. only a subset of features might be accessible) but also perturbation restrictions that are specific for some features. We demonstrate that using our weighted C&W approach, we are able to control the amount of perturbation per group of features. Our results show that the proposed method kept the perturbation of IAT features (G3) as low as possible, since these features are important for maintaining the goal of the DDoS attacks. In our experiments, we assigned weight values to $v = \{1, 2, 3\}$ for Group 1, 2 and 3, respectively. However, the impact of weighted perturbation could be amplified by using different weight ratios.

6 Conclusions

In this paper, we have proposed a novel approach to crafting adversarial examples that accounts for domain-specific constraints in the feature space. Features are assigned weights to reflect the difficulty of their modification. To achieve this, we have extended a well-established C&W attack [3] to perform optimisation with respect to feature weights and perturbation constraints. To express the difficulty of an attack, we consider two types of constraints: feature budget and perturbation budget. We have presented different methods to analyse the impact of attack budget on the strength of adversarial attacks against considered DL

models. We demonstrate that by modifying only a few features by less than an average of 0.2, we can achieve considerable success rates for adversarial example attacks. Our weighted crafting algorithm is able to restrict the perturbation mostly to independent and easy to access features, which increases the chance of creating a valid adversarial flow instance. We introduced a vulnerability score to provide an empirical, yet realistic, measure of robustness of network intrusion detection models against attacks of a given perturbation budget. The measure gives an intuition about the difficulty of the attacks that are required to achieve considerable success rates. Our results indicate that DNN and DBN models are more robust to adversarial examples that are targeted to hide DoS attacks than Probe or PortScan attacks. Furthermore, an AutoEncoder (AE) model, even though it achieves slightly worse accuracy in anomaly detection than other methods, seem to be the least vulnerable to adversarial examples. In future work, we will explore different threat models, investigate the transferability of adversarial examples [17] and test the feasibility of black box attacks, given a more restrictive validity constraints in the NIDS domain.

Acknowledgments. The research leading to this publication was supported by the EU H2020 project SOCCRATES (833481) and the Austrian FFG project Malori (873511).

References

1. Azami, M.E., Lartizien, C., Canu, S.: Converting SVDD scores into probability estimates: application to outlier detection. Neurocomputing **268**, 64–75 (2017)
2. Carlini, N., et al.: On evaluating adversarial robustness. arXiv:1902.06705 [cs, stat] (2019)
3. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. arXiv:1608.04644 [cs] (2016)
4. Carlini, N., Wagner, D.: Adversarial examples are not easily detected: bypassing ten detection methods. arXiv:1705.07263 [cs] (2017)
5. Dong, B., Wang, X.: Comparison deep learning method to traditional methods using for network intrusion detection. In: 2016 8th IEEE International Conference on Communication Software and Networks (ICCSN), pp. 581–585. IEEE (2016)
6. Gao, N., Gao, L., Gao, Q., Wang, H.: An intrusion detection model based on deep belief networks. In: 2014 Second International Conference on Advanced Cloud and Big Data, pp. 247–252 (2014)
7. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv:1412.6572 [cs, stat] (2014)
8. Hartl, A., Bachl, M., Fabini, J., Zseby, T.: Explainability and adversarial robustness for RNNs. arXiv:1912.09855 [cs, stat] (2020)
9. Hashemi, M.J., Cusack, G., Keller, E.: 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 - Big-DAMA 2019, pp. 14–21 (2019)
10. Hawkins, S., He, H., Williams, G., Baxter, R.: Outlier detection using replicator neural networks. In: Kambayashi, Y., Winiwarter, W., Arikawa, M. (eds.) DaWaK 2002. LNCS, vol. 2454, pp. 170–180. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46145-0_17

11. Kim, J., Shin, N., Jo, S.Y., Kim, S.H.: Method of intrusion detection using deep neural network. In: 2017 IEEE International Conference on Big Data and Smart Computing (BigComp), pp. 313–316 (2017)
12. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv:1412.6980 [cs] (2017)
13. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial machine learning at scale. arXiv:1611.01236 [cs, stat] (2017)
14. Mishra, P., Varadharajan, V., Tupakula, U., Pilli, E.S.: A detailed investigation and analysis of using machine learning techniques for intrusion detection. IEEE Commun. Surv. Tutorials **21**, 686–728 (2019)
15. Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: DeepFool: a simple and accurate method to fool deep neural networks. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2574–2582 (2016)
16. Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A.: The limitations of deep learning in adversarial settings. In: arXiv:1511.07528 [cs, stat], pp. 372–387, March 2016
17. Papernot, N., McDaniel, P., Goodfellow, I.: Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. arXiv:1605.07277 [cs] (2016)
18. Rezvy, S., Petridis, M., Lasebae, A., Zebin, T.: Intrusion detection and classification with autoencoded deep neural network. In: Lanet, J.-L., Toma, C. (eds.) SECITC 2018. LNCS, vol. 11359, pp. 142–156. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-12942-2_12
19. Sharafaldin, I., Habibi Lashkari, A., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proceedings of the 4th International Conference on Information Systems Security and Privacy, pp. 108–116 (2018)
20. Shone, N., Ngoc, T.N., Phai, V.D., Shi, Q.: A deep learning approach to network intrusion detection. IEEE Trans. Emerg. Top. Comput. Intell. **2**, 41–50 (2018)
21. Szegedy, C., et al.: Intriguing properties of neural networks. arXiv:1312.6199 [cs] (2013)
22. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the KDD CUP 99 data set. In: Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1–6 (2009)
23. Weng, T.W., et al.: Evaluating the robustness of neural networks: an extreme value theory approach. arXiv:1801.10578 [cs, stat] (2018)
24. Yang, K., Liu, J., Zhang, C., Fang, Y.: Adversarial examples against the deep learning based network intrusion detection systems. In: MILCOM 2018–2018 IEEE Military Communications Conference (MILCOM), pp. 559–564 (2018)
25. Zhang, X., Zhou, Y., Pei, S., Zhuge, J., Chen, J.: Adversarial examples detection for XSS attacks based on generative adversarial networks. IEEE Access **8**, 10989–10996 (2020)