applied sciences

MDPI

*Article*

# Few-Shot Network Intrusion Detection Using Discriminative Representation Learning with Supervised Autoencoder

**Auwal Sani Iliyasu [1,*], Usman Alhaji Abdurrahman [2] and Lirong Zheng [2]**

[1] School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China
[2] School of Information Science and Technology, Fudan University, Shanghai 200433, China; aausman18@fudan.edu.cn (U.A.A.); lrzheng@fudan.edu.cn (L.Z.)
* Correspondence: engrausan@gmail.com

**Abstract:** Recently, intrusion detection methods based on supervised deep learning techniques (DL) have seen widespread adoption by the research community, as a result of advantages, such as the ability to learn useful feature representations from input data without excessive manual intervention. However, these techniques require large amounts of data to generalize well. Collecting a large-scale malicious sample is non-trivial, especially in the modern day with its constantly evolving landscape of cyber-threats. On the other hand, collecting a few-shot of malicious samples is more realistic in practical settings, as in cases such as zero-day attacks, where security agents are only able to intercept a limited number of such samples. Hence, intrusion detection methods based on few-shot learning is emerging as an alternative to conventional supervised learning approaches to simulate more realistic settings. Therefore, in this paper, we propose a novel method that leverages discriminative representation learning with a supervised autoencoder to achieve few-shot intrusion detection. Our approach is implemented in two stages: we first train a feature extractor model with known classes of malicious samples using a discriminative autoencoder, and then in the few-shot detection stage, we use the trained feature extractor model to fit a classifier with a few-shot examples of the novel attack class. We are able to achieve detection rates of 99.5% and 99.8% for both the CIC-IDS2017 and NSL-KDD datasets, respectively, using only 10 examples of an unseen attack.

**Keywords:** network intrusion detection; few-shot learning; deep learning; discriminative autoencoder



**Citation:** Iliyasu, A.S.; Abdurrahman, U.A.; Zheng, L. Few-Shot Network Intrusion Detection Using Discriminative Representation Learning with Supervised Autoencoder. *Appl. Sci.* **2022**, *12*, 2351. https://doi.org/10.3390/app12052351

Academic Editor: Leandros Maglaras

Received: 14 December 2021
Accepted: 10 January 2022
Published: 24 February 2022



## 1. Introduction

Cyber defense is a continuous process that entails tasks, such as prevention, detection, and recovery, which are applied at various system levels. Network intrusion detection is a branch of cyber security that deals with the detection of attacks at the network layer level.

Network intrusion detection techniques can be broadly divided into two types: signature-based and anomaly-based methods [1]. Signature-based methods operate by matching incoming network traffic against a predefined set of known attack signatures. Thus, they perform well in detecting previously known attack signatures; however, signature-based methods fail to detect novel attacks [2]. On the other hand, anomaly-based methods, which entail machine learning methods, operate by modeling normal network traffic data and then flag any network traffic that deviates from the model pattern as an anomaly. However, these approaches sometimes lead to too many false alarm rates (FARs).

Network intrusion detection using machine learning methods has been studied for a long time, with many commercial intrusion detection systems (IDSs) using machine learning algorithms as part of their detection engines [3].

Recently, technologies such as cloud computing, IoT, and 5G have led to an explosion in the volume and diversity of network traffic, which provide fertile ground for applying deep learning (DL) techniques. Deep learning techniques are end-to-end learning models,

capable of learning highly complex non-linear functions, which enable them to learn powerful representations directly from input data [4]. Thus, recent research on intrusion detection system (IDS) methods are mostly focused on this area [5].

However, network IDSs based on supervised deep learning techniques require huge amounts of labeled data in order to generalize well. Collecting a large-scale malicious sample to train DL classifiers is prohibitively expensive, and subject to obsolescence as the landscape is constantly evolving. Regardless, unsupervised anomaly-based methods provide an alternative towards generalization of an unseen malicious sample, and these approaches are highly susceptible to false alarm rates [6]. Hence, there has been an increase in interest from the research community towards approaches that require a handful of samples to achieve detection. Since collecting a few samples of malicious traffic is more realistic in a practical settings, which, for instance, can be realized from a few successfully detected intrusions from a deployed detection system, few-shot learning is emerging as an alternative to conventional supervised learning methods to simulate more realistic settings.

Few-shot learning measures the challenging issue of a model's ability to generalize new tasks using limited data [7]. This was addressed recently, based on the idea of meta-learning or "learning to learn" [8–12]. The meta-learning paradigm consist of two disjointed stages: meta-training and meta-testing. Each of the meta stages consists of a number of classification tasks with limited training data that require fast adaptability by the learner. The goal is to leverage the meta-training stage to learn transferable knowledge from a set of tasks that will enable fast adaptability to novel tasks in the testing stage.

However, recently, it has been established that good learned representations are very powerful for few-shot classification tasks, and perform on par with, or slightly worse than, the current set of complicated meta-learning algorithms [13–15]. Therefore, in this paper we propose a simple framework that relies on learning good representations to achieve few-shot intrusion detection. Our approach consists of a linear model trained on top of a pre-trained feature extractor model. The feature extractor model is trained to learn good representations using a discriminative autoencoder.

In contrast to conventional autoencoders, which are purely unsupervised representation learning methods, discriminative autoencoders are a form of supervised autoencoders that leverages the class information of their inputs. Thus, they combine both reconstruction and classification errors in their objective functions. This makes the representations learned by discriminative autoencoders more discriminative and more suitable for classification tasks [16,17].

The remainder of the paper is organized as follows: Section 2 presents the related work, Section 3 presents our problem formulation of few-shot intrusion detection using discriminative autoencoders, Section 4 presents the results and discussion of our experiments, and Section 5 concludes the paper.

## 2. Related Works

Traditionally, networks are defended against intrusion using signature-based techniques, whereby incoming network traffic is compared against commonly known attack patterns. These approaches perform well against previously known attacks, but fail to detect novel attacks.

Classical machine learning (ML) methods provide an upgrade over traditional signature-based techniques. These methods exploit various features of network traffic, which enable them to detect attack signatures without explicit rule specifications [18]. Thus, popular classical ML approaches, such as K-nearest neighbor (KNN) [19], support vector machines (SVM) [20], decision tree (DT) [21], and random forest (RF) [22], have all been employed as network-based IDSs.

For example, Kutrannont et al. [23] proposed a KNN-based IDS. KNN operates based on the assumption that a sample belongs to the class where most of its top K-neighbors reside. Therefore, parameter K affects the performance of the model. In their work, Kutrannont et al. proposed the integration of a simplified neighborhood classification

using a percentage instead of group rankings. Taking into account the unevenness of data distribution, the improve rule selects a fixed percentage (50%) of neighboring samples as neighbors and its efficiency is enhanced via parallel processing using a graphical processing unit (GPU). The algorithm performs well on sparse data, achieving an accuracy of 99.30%.

Goeschel et al. [24] employed a combination of SVM, decision tree (DT), and naïve Bayes classifiers. The SVM was first trained to perform a binary classification to separate data instances into benign and malicious classes. The malicious classes are then categorized into specific classes of attacks using a DT classifier. However, since DT can only separate known classes of attacks, they further employed a naïve Bayes classifier to identify unknown attacks types. This hybrid method achieved an accuracy of 99.62% and a false alarm rate of 1.57%.

Malik et al. [25] proposed an IDS using random forest (RF) and particle swarm optimization (PSO). They trained the IDS in two stages: feature selection and classification. The PSO serves as feature selection algorithm, which is used to select appropriate features for classifying attacks, while the RF is used as a classifier. They evaluated their approach using the KDD cup99 dataset, and achieved detection rates of 99.92%, 99.49%, and 88.46% on DoS, Probe, and U2R attack classes.

Recently, there has been widespread adoption of DL techniques for network-based IDSs for sizeable numbers of datasets. These techniques can operate directly on raw data, learn features, and perform classifications. Hence, they achieve better performances when compared to classical machine learning methods [26]. Deep learning models, such as multi-layer perceptron (MLP), convolutional neural network (CNN), autoencoders (AE), recurrent neural network (RNN), as well as deep generative networks, such as the deep belief network (DBN) and generative adversarial networks (GANs) have all been applied in the context of network-intrusion detection [27,28].

Min et al. [29] proposed an IDS named TR-IDS, which leverages both statistical features as well as payload features. They employed a CNN to extract important features from the payload. To accomplish this, they first encoded each byte in the payload in to a word vector using skip-gram word embedding, and then applied the CNN to extract the features. The extracted features were then combined with the statistical features generated from each network flow, which included fields from the packet header and statistical attributes of the entire flow. The features were then used to train a random forest classifier, which achieves an accuracy of 99.13%.

In the work by Yin et al. [30], a recurrent neural network (RNN) was directly applied for intrusion detection tasks. The RNN model achieved better performance on a NSL-KDD dataset when compared with classical ML techniques consisting of support vector machines and random forest.

Wang et al. employed a combination of CNN and long short-term memory (LSTM). Intuitively, the CNN learns the low-level spatial features of network traffic, while the LSTM learns the high-level temporal features of the data. The learned features enable the model to improve the false alarm rate of an IDS [31].

In another work, Al Qatf et al. employed a sparse autoencoder (AE) for dimensionality reduction and the reduced features are then retrained using the SVM classifier. This enables the model to outperform classical machine learning methods [32].

Similarly, in a recent work by Narayana et al., a hybrid methodology involving a sparse autoencoder, DNN, and LSTM was employed. In the first stage, the autoencoder is trained in an unsupervised fashion with smoothed l1 regularization to enforce sparsity. This enables the autoencoder to learn sparse representations, which are then used to train the MLP and LSTM classifiers in the second stage. The model performs better than conventional deep learning classifiers in terms of detection rates and low false positive rates [33].

Another hybrid intrusion detection method that employs both classical machine learning and deep learning techniques was proposed by Le, et al. They first built a feature selection model termed a sequence forward selection (SFS) algorithm (SFSD) and a decision

tree. The SFSD algorithm selects the best subset of features, which are then used in the second part to train various forms of RNN (traditional RNN, LSTM and gated recurrent neural network (GRU)). The model achieves significant improvements in detection rates when compared with classical methods [34].

However, these techniques require huge amounts of labeled data during training in order to generalize well. The dynamic nature of the modern-day cyber-threat landscape makes it unfeasible or prohibitively expensive to acquire sufficient enough malicious samples to train deep learning classifiers. Therefore, a trend is developing towards techniques that require only a few shot of malicious examples to achieve detection.

For example, Hindy et al. proposed an intrusion detection model using one-shot learning. The main idea of one-shot learning is to learn patterns and similarities from previously seen classes that enable classifying unseen classes using only one instance. Thus, one-shot learning is an instance of few-shot learning, whereby the number of examples is restricted to only a single example [35]. To model an IDS using one-shot learning, Hindy et al. employed a Siamese neural network, a form of neural network consisting of twin networks. The Siamese network is trained using two pairs of instances to learn patterns and similarities instead of fitting the model to fixed classes. Therefore, during the training stage, the Siamese network learns patterns and discriminate between benign traffic and different classes of a known cyber-attacks. At the evaluation stage, a new traffic instance is compared against all known classes (used during training) without any form of additional training. Although the approach provides a simple framework for one-shot learning, in general, they achieve lower detection rates relative to other works [36].

In another work, Xu et al. proposed an intrusion detection method using few-shot learning. They employed a deep neural network architecture (DNN) named FC-Net, which is composed of two parts: a feature extraction network and a comparison network. FC-Net is trained using a meta-learning approach consisting of two disjointed stages of meta-training and meta-testing. In the meta-training phase, the feature extraction network of FC-Net is trained using several meta-tasks, where a meta-task is comprised of a binary classification between an attack category and benign traffic. This enables FC-Net to learn a pair of feature maps, which are then used by the comparison network in the meta-testing stage to determine whether a new traffic instance belongs to the different classes of attacks learned during training [37]. However, one drawback with their approach is it requires a complex DNN architecture and computationally intensive optimization procedures.

## 3. Our Proposed Few-Shot Intrusion Detection Method

Supervised learning approaches for network intrusion detection require all categories of attacks to be known in advance, with a sufficient number of training examples available for each category. The basic task is to use a classifier, $f$, to infer labels for network traffic samples, $N$. The number of samples, $N$, is often very large and is simply composed of two groups: a training set and a test set. Contrary to this, in real-world settings, new attacks frequently emerge, and only subset of categories are known beforehand, with few examples per category. Therefore, in such scenarios, where the number of samples, $N$, is small, the problem is considered as a few-shot classification. Applying the conditions of a supervised learning method to this problem will encounter overfitting.

Few-shot learning is popularly addressed based on the meta-learning paradigm, which is composed of meta-training and meta-testing. Each one of the meta stages consists of a number of classification tasks, where each task describes a pair: training (support) and testing (query). The meta-training set is described as $T = \left\{\left(D_i^{train}, D_i^{test}\right)\right\}_{t=1}^{I}$ and the meta-testing set is $S = \left\{\left(D_q^{train}, D_q^{test}\right)\right\}_{q=1}^{Q}$, with each dataset containing pairs of data points and their ground-truth labels, i.e., $D^{train} = \{(x_t, y_t)\}_{t=1}^{T}$ and $D^{test} = \left\{\left(x_q, y_q\right)\right\}_{q=1}^{Q}$, which are sampled from the same distribution. The objective is to leverage the meta-training stage to learn good representations, which will enable it to adapt quickly to unseen tasks in the meta-testing stage, using powerful optimization techniques.

In a network intrusion detection context, a task, $T$, can be simply defined as a binary classification between a normal network traffic sample and a category of malicious samples. Supposing that there are five different network traffic samples, $O, A, B, C$ and $E$ such that, sample $O$ is a benign network traffic, samples $A, B, C$ indicate known categories of attacks with sufficient examples, while the remaining sample, $E$, refers to a newly found category of attack with a few examples. The goal is to identify the new attack sample, S, with as few examples as possible. Then, three different tasks can be constructed, $T_1, T_2$ and $T_3$ where $T_1, T_2$ and $T_3$ define a binary classification task between a normal sample $O$ and attack categories $A, B$ and $C$. $T_1, T_2$ and $T_3$ constitute the meta-training set, while the meta-test set consists of a normal sample, $O$, and the novel class, $E$, which has few examples. The idea is to leverage the meta-training stage to learn transferable knowledge from $T_1, T_2$ and $T_3$ that will enable a classifier to accomplish task $T_4$ (a binary classification between normal sample $O$ and attack category $E$) with as few examples as possible during the meta-testing phase. Thus, in our case, a discriminative autoencoder was employed to acquire such transferable knowledge.

*Feature Extraction with Discriminative Autoencoder*

Autoencoders have been proved to be powerful models for learning representations in an unsupervised fashion. However, discriminative autoencoders are a form of autoencoders that, in addition to residual errors, considers class information of the input in its objective function. This ensures that more powerful and discriminative representations are learned than those learned by conventional autoencoders.

We adopted the discriminative autoencoder proposed in [38], which, in its setup, uses data from two distributions, termed positive ($X^+$) and negative ($X^-$), with their labeled information. The discriminative autoencoder then learns a manifold that is good at reconstructing the data from the positive distribution, while ensuring that those of the negative distributions are pushed away from the manifold. This enables it to learn robust patterns and similarities that separate the two distributions.

In our case, the two distributions, $X^+$ and $X^-$, can be generated from benign network traffic classes and malicious traffic classes.

Let $l(x)$ denote the label of an example, $x$, with $l(x) \in \{-1, 1\}$ and $d(x)$ is the distance of that example to the manifold, with $d(x) = \|x - \overline{x}\|$. Then, the loss function is described as:

$$L(X^+ \cup X^-) = \sum_{x \in X^+ \cup X^-} \max(0, l(x) \cdot (d(x) - 1)) \quad (1)$$

Thus, to train the discriminative autoencoder, we merged all the meta-training tasks $D_t^{train}$ from $T$ into a single training set, $D^{new}$, of seen classes:

$$D^{new} = \cup \left\{ D_1^{train}, ..., D_t^{train}, ..., D_T^{train} \right\} \quad (2)$$

We trained the discriminative autoencoder during the meta-training stage (Algorithm 1). After training, the decoder part of the model was discarded, while the encoder module, which then served as our feature extractor was retained. The encoder was then employed in a fixed state (no fine-tuning) in the meta testing stage. The meta testing stage consists of the task of identifying a novel class of attack, which has few examples.

For a given task $\left( D_q^{train}, D_q^{test} \right)$ sampled from the meta-testing set, $S$, we trained a classifier, $f$, on top of the extracted features to recognize the unseen classes using the training dataset, $D_q^{train}$ (Algorithm 2).

**Algorithm 1** Discriminative Autoencoder Training

**Input:** meta-training dataset $D$ containing, $n$ normal data samples and $m$ malicious samples, $l(x)$ the label of the dataset with $l(x) \in \{1, 0\}$,
**Output: encoder $f_e$ and a decoder $f_d$**
$\theta_e \leftarrow$ initialize encoder parameters
$\theta_d \leftarrow$ initialize decoder parameters
**Repeat**
**for** $i = 1$ to $k$ do
Draw a batch of $k$ samples $x^{(1)}, \ldots, x^{(k)}$ from the dataset $D$
$z^i = f_e\left(x^i\right)$
$\hat{x}^i = f_d\left(z^i\right)$
$L_{DAE} = \frac{1}{k}\sum_{i=1}^{k} \max\left(0, l(x).\left(\| x^i - \hat{x}^i \| - 1\right)\right)$
**end for**
// update parameters with gradients
$\theta_e \leftarrow \theta_e - \nabla_{\theta_e} L_{DAE}$
$\theta_d \leftarrow \theta_d - \nabla_{\theta_d} L_{DAE}$
**until** convergence of parameters $\theta_e, \theta_d$

**Algorithm 2** Few-Shot Detection

**Input:** meta-testing dataset $D$ containing, $n$ normal data samples and $m$ few malicious samples with $n \gg m$, $l(x)$ the label of the dataset with $l(x) \in \{1, 0\}$, trained encoder $f_e$
**Output: classifier $c_l$, prediction $l_{pred}$**
$\theta_c \leftarrow$ initialize classifier parameters
**Repeat**
**for** $i =$ to $k$ do
Draw a batch of $k$ samples $x^{(1)}, \ldots, x^{(k)}$ from the dataset $D$
$f^i_{extract} = f_E\left(x^i\right)$
$l_{pred} = C_l\left(f^i_{extract}\right)$
$L_C = binarycrossentropy\left(l, l_{pred}\right)$
**end for**
// update parameters with gradients
$\theta_C \leftarrow \theta_C - \nabla_{\theta_C} L_C$
**until** convergence of parameters $\theta_C$

## 4. Evaluation

### *4.1. Evaluation Metrics*

Accuracy, recall (detection rate) and precision are common metrics normally used when evaluating a classifier. Accuracy, which indicates the percentage of correctly classified data items, is a poor metric for a task where the dataset is largely skewed in favor of normal samples. Similarly, a high detection rate implies fewer chances of a model missing alarming anomalies, while a high precision highlights the ability of a model to classify normal data. We consider detection rate (DR) to be more significant in our case than precision; for example, if a model predicts a normal sample as an attack, it is easier to correct the prediction of the model through domain knowledge, since there are few attack samples. However, if the model fails to detect an attack, it is difficult to find the attack in such a huge dataset. Nonetheless, we consider all three metrics; when comparing our approach with baseline models, we only considered DR since it is more significant in our situation.

### *4.2. Datasets*

We evaluated the performance of our approach against baseline models using the following network intrusion detection datasets: CIC-IDS2017 and NSL-KDD. The CIC-IDS2017 dataset reflects recent attacks, and, to some extent, it satisfies the criteria for reliable intrusion detection datasets proposed by [39], which are anonymity, attack diversity,

complete capture, complete interaction, complete network configuration, available protocol, complete traffic feature set, meta data, heterogeneity, and labeling.

The dataset was developed at the Canadian Institute of Cybersecurity of the University of New Brunswick (UNB) in 2017. The dataset comprises raw PCAP files, as well as 80 statistical features generated from the PCAP files, which were captured on different days of a week (from Monday to Friday). The dataset considers several attacks and sub-attacks, as depicted in Table 1.

**Table 1.** Attack composition of the CIC-IDS2017 datasets.

| Attack Class | Subclasses | Number of Records |
|---|---|---|
| DoS | Dos Hulk | 231,073 |
| | Dos Slow loris | 5796 |
| | Dos Golden Eye | 10,293 |
| | DoS Http Test | 5499 |
| Web Attack | Brute force | 1507 |
| | XSS | 652 |
| | SQL injection | 21 |
| Ports Scan | - | 158,930 |
| DDoS | - | 41,835 |
| Botnet | - | 1966 |
| FTP-Patator | - | 7938 |
| SSH-Patator | - | 5897 |
| Infiltration | - | 36 |
| Benign | - | 2,358,036 |

The NSL-KDD dataset was also generated at the Canadian Institute of Cybersecurity. This dataset was purposely created to solve the problem of the original KDDcup'99 dataset, which has about 78% and 75% of the training and testing set duplicated, respectively [40]. The NSL-KDD rectifies this problem and still retains the original 41 features.

Table 2 depicts a breakdown of some of the attacks that exist in the dataset with more than five samples.

**Table 2.** Attack composition of the NSL-KDD datasets (training set).

| Attack Type | Subclasses | Number of Records |
|---|---|---|
| DoS | Neptune | 41,214 |
| | Smurf | 2646 |
| | Tear drop | 892 |
| | Back | 956 |
| Probe | Ip sweep | 3599 |
| | Nmap | 1493 |
| | Port sweep | 2931 |
| | Satan | 3633 |
| User to root (U2R) | Buffer overflow | 30 |
| | Load module | 9 |
| | rootkit | 10 |
| Remote login (R2L) | Guess password | 53 |
| | Warezmaster | 890 |
| | Imap | 11 |
| | Multihop | 7 |

### *4.3. Implementation*

We employed tensorflow 2.6 as the deep learning framework to implement our approach. We implemented the feature extractor module (discriminative autoencoder), and

the classifier was a standard multi-layer perceptron (MLP) neural network. Table 3 present the architectural details of the modules.

**Table 3.** Architectural details of our models.

| Model | Architectural Details | |
|---|---|---|
| Encoder | Layer 1 | Dense, output: 128, activation: Relu |
| | Layer 2 | Dense, output: 64, activation: Relu |
| | Layer 3 | Dense, output: 32, activation: Relu |
| | Layer 4 | Dense, output: 8, activation: Relu |
| Decoder | Layer 1 | Dense, output: 32, activation: Relu |
| | Layer 2 | Dense, output: 32, activation: Relu |
| | Layer 3 | Dense, output: No, of features, activation: sigmoid |
| Classifier | Layer 1 | Dense, output:1, activation: sigmoid |

### *4.4. Experiments and Results*

We conducted the following experiments to evaluate our few-shot detection approach.

#### 4.4.1. Experiment 1: Detecting Mutants of Existing Attacks

The objective of this experiment was to evaluate our approach in detecting mutants of existing attacks. Malicious users develop mutants of existing attacks to evade detection, as conventional supervised learning models detect such mutants poorly, especially when there is significant variation with the known existing attack.

To accomplish this, we extracted categories of attacks, which we believed to comprise variants of one another in both the NSL-KDD and CIC-2017IDS datasets. Tables 1 and 2 depict the compositions of such attack mutants.

To evaluate our approach in detecting a particular mutant, we utilized all the other mutants of the attack as a data source for the meta-training stage. For instance, if the attack mutant we wanted to detect was DoS hulk, then our meta-training data source will comprise all the other DoS subclasses (golden eye, slow loris, slow http test). The DoS hulk would then serve as the data source for the meta-testing stage. We then trained our feature extractor using the meta-training set. Thereafter, in the few-shot detection stage, we selected 10 samples of the meta-testing set and trained our classifier on top of our feature extractor. We applied the same logic to all attacks in both the NSL-KDD and CIC-IDS2017 datasets. However, we conducted this experiment on classes with sufficient numbers of samples. Therefore, on the NSL-KDD set we were able to perform the experiment on DoS and probe classes. While on the CIC-IDS2017 dataset, only the DoS class has a sufficient number of samples.

Figures 1–3 display the results of our experiment on both the CIC-IDS2017 and NSL-KDD datasets. As can be seen from the results, our model achieves the highest detection rate of 99.9% on DoS attack mutant Neptune (Figure 1), followed by 90.0% and 80.0% on tear drop and smurf attack, respectively. These are good results, considering that we employed few-shots examples to train our classifier. However, our model achieves a DR of 0.0% on DoS attack mutant back. In fact, our model scores 0.0% on all the other performance metrics (accuracy and precision). This indicates that the DoS attack mutant back has significant variations with other DoS attack subclasses.

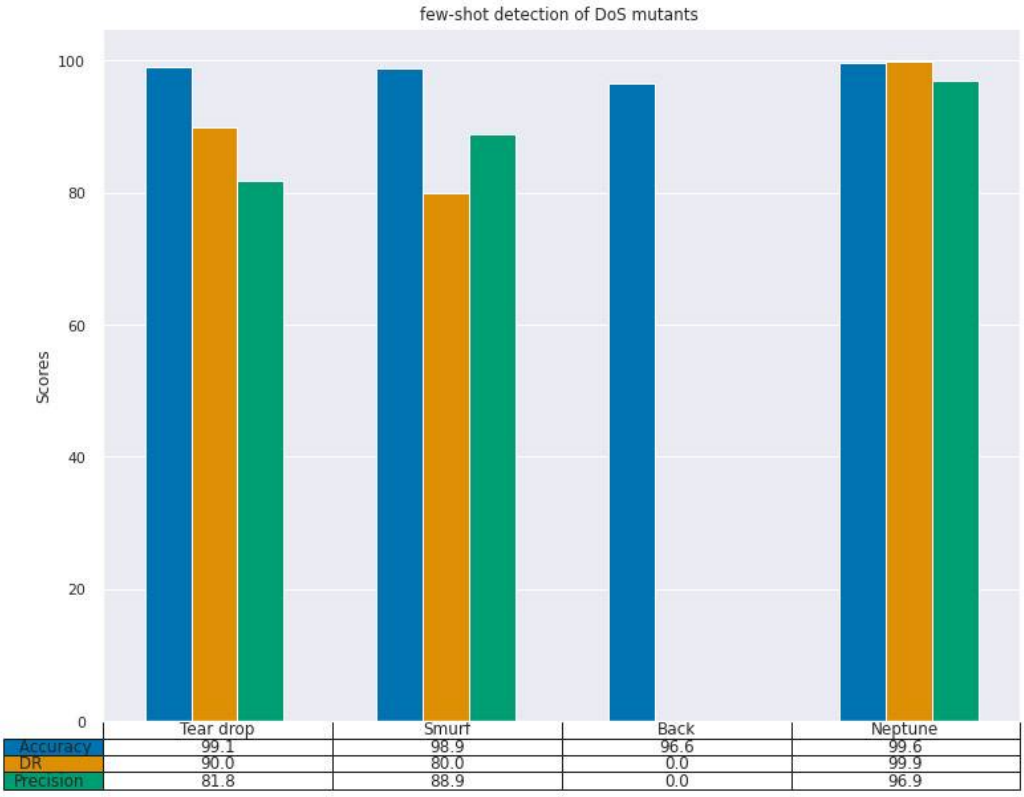



**Figure 1.** Results of few-shot detection of DoS attack mutant for NSL-KDD dataset.

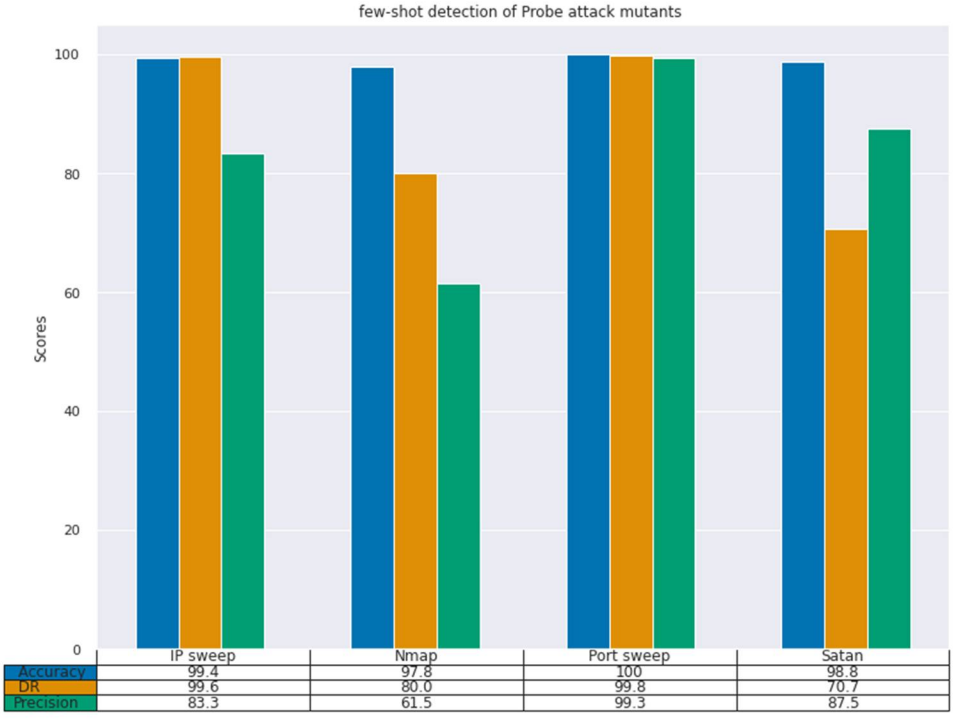



**Figure 2.** Result of few-shot detection of probe attack mutants for NSL-KDD dataset.

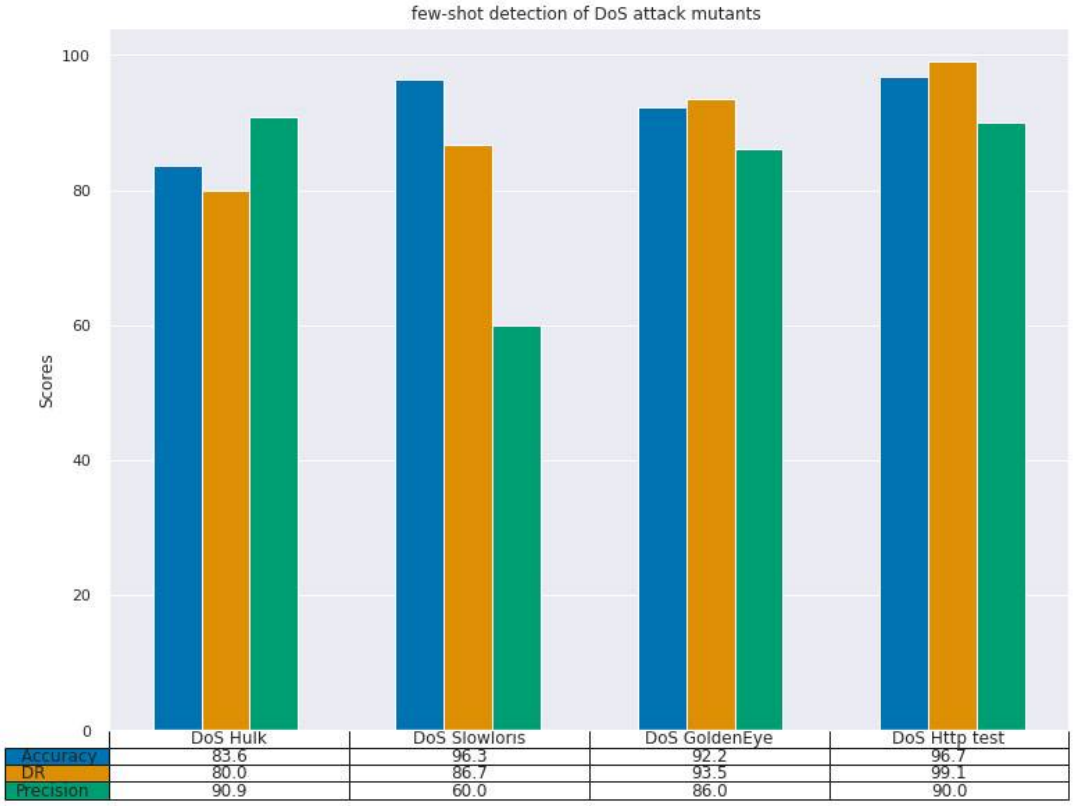


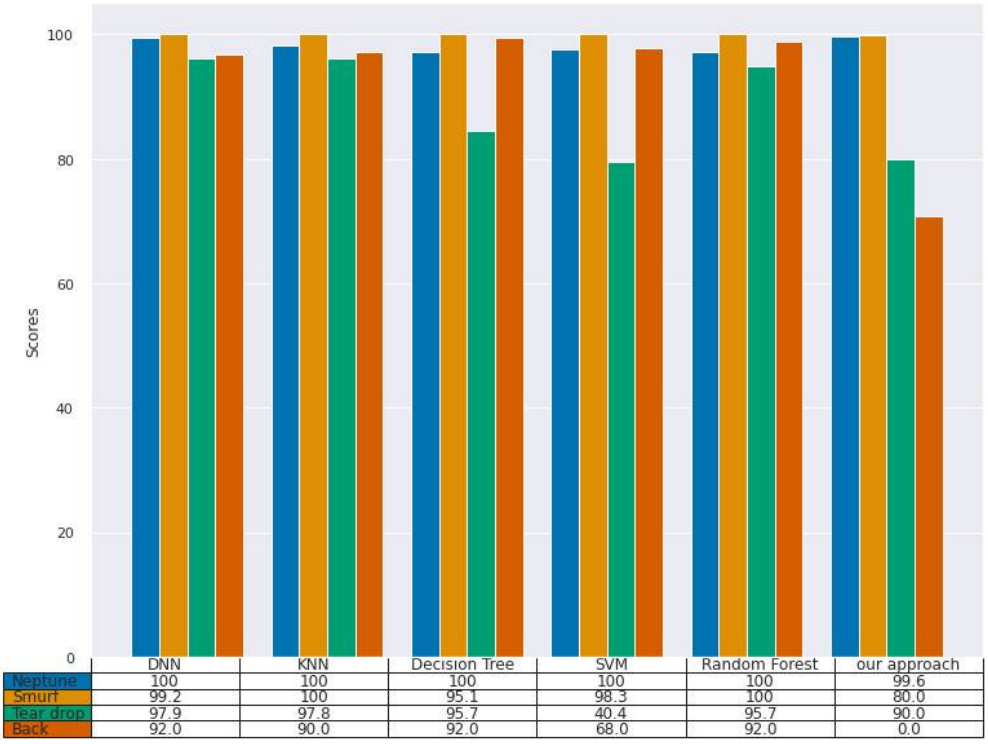

**Figure 3.** Result of few-shot detection on DoS attack mutant for CIC-IDS2017 dataset.

For the probe attack mutants (Figure 2), our model also performs well, achieving the highest DR score of 99.8% on the port sweep attack, and a DR score of 99.6%, 80.0%, and 70.7% for the port sweep, Nmap and Satan, respectively. All the attack mutants achieved a good DR, which indicated that the attack mutants had many attributes in common, which our feature extractor model was able to discover.

Similarly, with the CIC-IDS2017 dataset, our model's performance was good. We achieved the highest DR score of 99.1% on DoS http test attack (Figure 3), and DR scores of 93.5%, 86.7%, and 80.0% on DoS golden eye, DoS slow loris and DoS hulk attacks, respectively. The results follow a similar trend as in the NSL-KDD dataset.

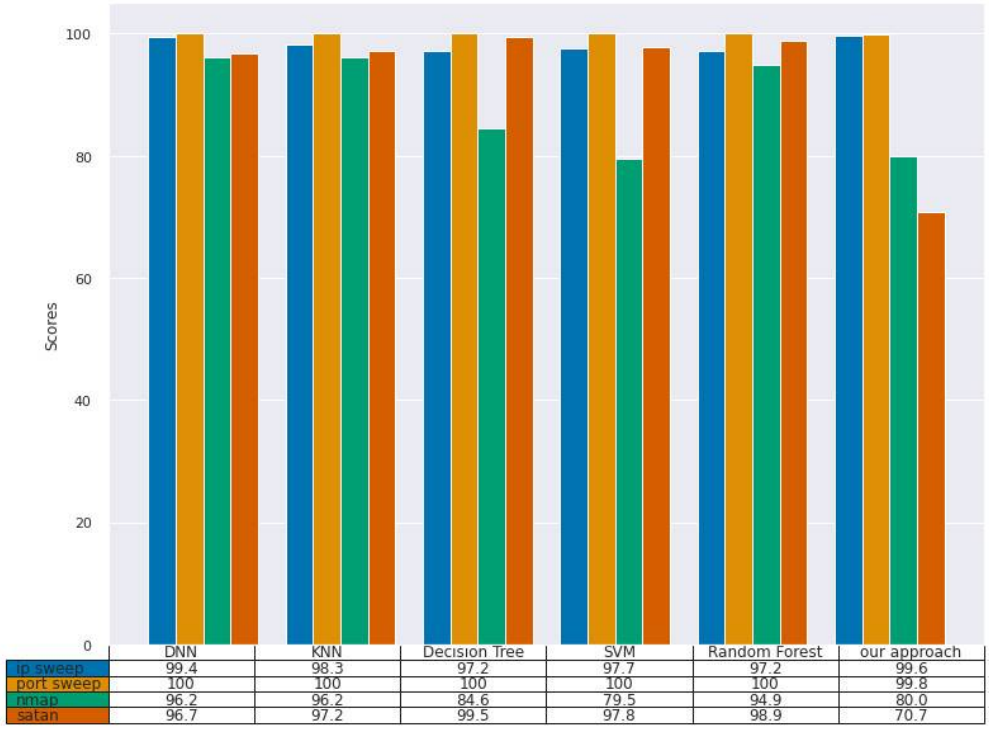
We also compared our model's DR with baseline models, and we employed a combination of classical and deep learning algorithms as our baselines. For deep learning, we employed a multi-layer perceptron (MLP), while for the classical ML models we employed K-nearest neighbor (KNN), decision tree, support vector machine (SVM) and random forest. All the baseline models were trained using the full dataset.

Figures 4–6 depict the results of such comparisons. As can be seen from results, there was not much of a difference in terms of performance between the DL models, ML models, and our model. It is well known that classical ML models perform well on small and medium datasets. The highest DR score was 100% on the Neptune DoS mutant (Figure 4), which was achieved by all baselines, while our model scored 99.6% (despite having been trained with only 10 examples), which is quite on par with the baselines results. The lowest DR score was for the back attack mutant, where our model's DR score as 0.0%. However, the lowest DR score from the baselines was achieved by SVM, which scored 40.4% on the tear drop attack.



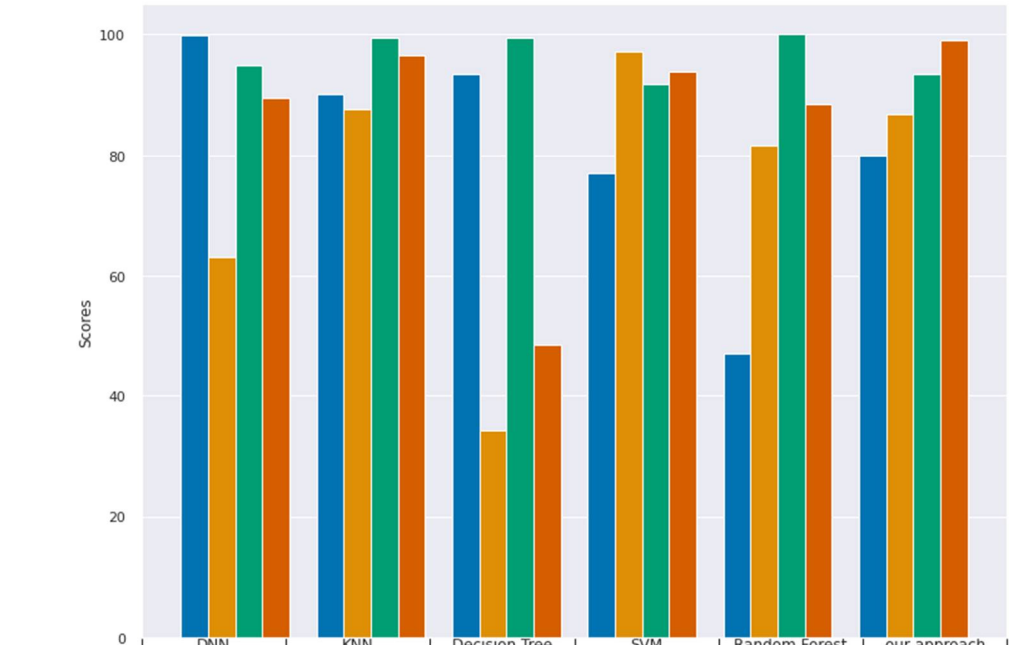

| | DNN | KNN | Decision Tree | SVM | Random Forest | our approach |
|---|---|---|---|---|---|---|
| Neptune | 100 | 100 | 100 | 100 | 100 | 99.6 |
| Smurf | 99.2 | 100 | 95.1 | 98.3 | 100 | 80.0 |
| Tear drop | 97.9 | 97.8 | 95.7 | 40.4 | 95.7 | 90.0 |
| Back | 92.0 | 90.0 | 92.0 | 68.0 | 92.0 | 0.0 |

**Figure 4.** Detection rate (DR) comparison of various classification methods on DoS subclasses for NSL-KDD datasets.




| | DNN | KNN | Decision Tree | SVM | Random Forest | our approach |
|---|---|---|---|---|---|---|
| ip sweep | 99.4 | 98.3 | 97.2 | 97.7 | 97.2 | 99.6 |
| port sweep | 100 | 100 | 100 | 100 | 100 | 99.8 |
| nmap | 96.2 | 96.2 | 84.6 | 79.5 | 94.9 | 80.0 |
| satan | 96.7 | 97.2 | 99.5 | 97.8 | 98.9 | 70.7 |

**Figure 5.** Detection rate (DR) comparison of various classification methods on probe attack subclasses for NSL-KDD datasets.




| | DNN | KNN | Decision Tree | SVM | Random Forest | our approach |
|---|---|---|---|---|---|---|
| DoS Hulk | 99.8 | 90.2 | 93.5 | 77.0 | 47.1 | 80.0 |
| DoS Slowloris | 63.2 | 87.6 | 34.2 | 97.1 | 81.5 | 86.7 |
| DoS GoldenEye | 94.9 | 99.5 | 99.4 | 91.7 | 100 | 93.5 |
| DoS Http test | 89.4 | 96.5 | 48.5 | 93.8 | 88.5 | 99.1 |

**Figure 6.** Detection rate (DR) comparison of various classification methods on DoS subclasses for CIC-IDS2017 datasets.

In addition, for the probe attack class, the result followed similar trend, the highest DR score achieved was 100%, which was achieved by the baseline models on the port sweep attack mutant (Figure 5), while our model was slightly worse with a DR score of 99.8% on the same attack class.

Our model also recorded the lowest DR score, 70.7%, on the Satan attack mutant, while the lowest DR score achieved by the baselines was 79.5%, which was scored by SVM on the Nmap attack mutant. Overall, there was no significant difference in performance with our model; our model was either slightly worse or on par with the baselines.

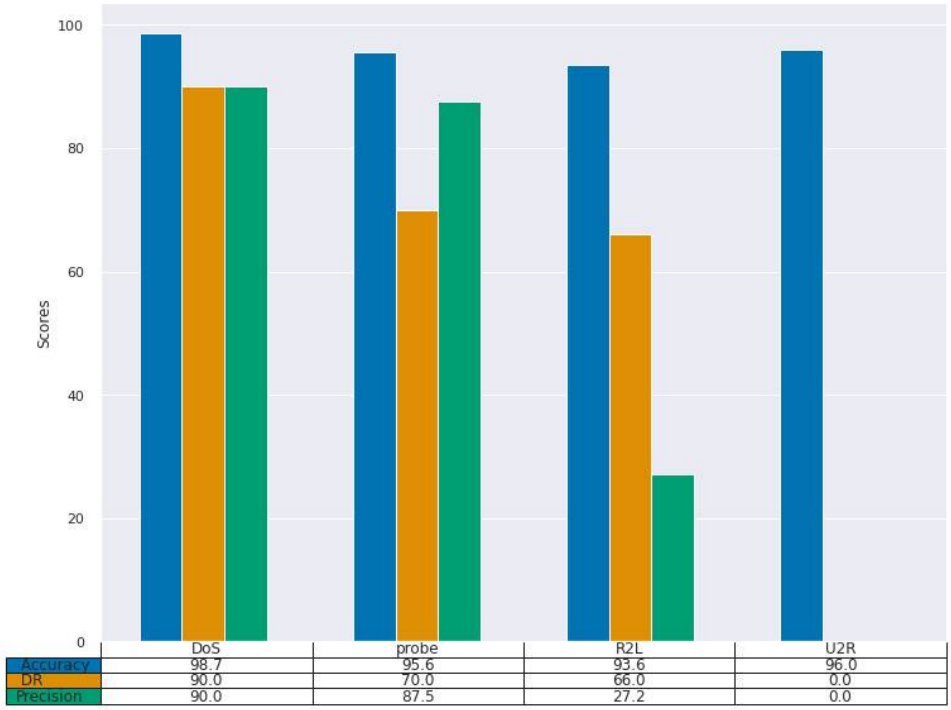
Similarly, Figure 6 presents the results of the comparison with baselines models for the CIC-IDS2017 dataset. The highest performing model was random forest, which achieved a DR of 100% for the DoS golden eye attack mutant. The lowest DR score was 48.5% which was achieved by the decision tree model for the DoS http test attack mutant. Similar to the NSL-KDD dataset, our model performed competitively with the baseline models, where it achieved DRs of 99.6%, 99.8%, 80.0%, and 70.7% on IP sweep, port sweep, Nmap and Satan, respectively.

#### 4.4.2. Experiment: General Anomaly Detection

The objective here was to evaluate our approach in detecting broader classes of attacks that were not seen before. For example, the NSL-KDD dataset can be broadly categorized into the following classes: DoS, Probe, R2L and U2R. Similarly, the CIC-IDS2017 dataset, when classified broadly, is made up of the following classes: DoS, port scan, heart bleed, Botnet, SSH-Patator, FTP-Patator, and web-based attacks

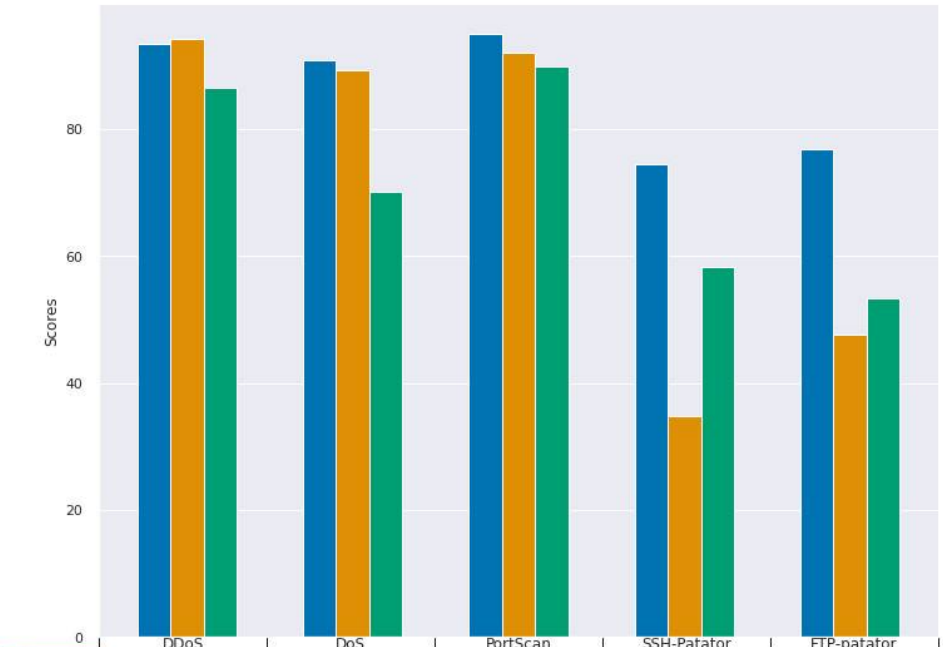
To perform the experiment, we applied a similar logic as in experiment 1. For instance, to detect the probe class of attack in the NSL-KDD dataset, our meta-training set consisted of all other attack classes (DoS, R2L, and U2R). While the meta-testing set comprised the attack we wanted to classify, which was the probe attack in this case. We applied the same procedure as for the CIC-IDS2017 dataset.

Figures 7 and 8 present the results of our experiments when the number of samples selected to train our classifier was limited 10 for both NSL-KDD and CIC-IDS2017, respectively.




| | DoS | probe | R2L | U2R |
|---|---|---|---|---|
| Accuracy | 98.7 | 95.6 | 93.6 | 96.0 |
| DR | 90.0 | 70.0 | 66.0 | 0.0 |
| Precision | 90.0 | 87.5 | 27.2 | 0.0 |

**Figure 7.** Result of experiment 2: general anomaly detection for NSL-KDD dataset.




| | DDoS | DoS | PortScan | SSH-Patator | FTP-patator |
|---|---|---|---|---|---|
| Accuracy | 93.3 | 90.9 | 95.0 | 74.5 | 76.8 |
| DR | 94.1 | 89.3 | 92.1 | 34.8 | 47.6 |
| Precision | 86.4 | 70.1 | 89.9 | 58.2 | 53.3 |

**Figure 8.** Result of experiment 2: general anomaly detection for CIC-IDS2017 dataset.

For the NSL-KDD dataset, (Figure 7), our approach performed reasonably well, especially for DoS and probe classes. Our model achieved the highest DR score of 90% with the DoS class and 70% on probe class. However, overall, our model's performance was low compared to the results of experiment 1. Some attacks were poorly detected. For example, our model's DR scores was 66% for the R2L class, and it completely failed to detect the U2R attack class, scoring a DR of 0.0%.

Similarly, the result of our experiment on CIC-IDS2017 (Figure 8) follows similar trend. There is drop in performance, our model performance was not as good as that

of experiment 1. The highest DR score was 94.1%, which was achieved for the DDoS attack class. Our model poorly detects the Patator attack classes, scoring 34.8% and 47.6%, respectively.

### 4.5. Discussion

Section 4.4 presents a performance evaluation of our approach. We designed two different experiments to evaluate our approach. The first experiment was designed to evaluate detecting mutants of existing attacks (subclasses of attacks). This was designed to addresses situations where attackers introduce some variations of a known existing attack to evade detection. We hypothesized that finding good representations using existing known similar attacks will enable the model to detect a novel mutant of that attack by using a few examples since attack mutants share similar attributes. Thus, our feature extractor would be able to discover the manifold of a broader class of such attacks.

Figures 1–3 in Section 4.4 present the results of our experiment, which prove our hypothesis. As we were able to detect several mutants with an excellent detection rate using only a few shot examples. We achieved up to 99.9% DR on the Neptune DoS mutant (Figure 1), and, overall, we achieved an excellent detection rate. This shows that our feature extractor was able to discover the manifold of optimal representations of such a class.

To further validate the efficacy of our feature extractor model, we increased the number of samples required to train our classifier in the few-shot detection stage from 10 to 20 samples. However, as can be seen in Tables 4–6, respectively, the DR varies slightly, despite doubling the number of samples. This is contrary to conventional supervised learning approaches, where the DR increases significantly with an increase in the number of training examples.

**Table 4.** Effect of increasing the number of samples on DoS attacks in the NSL-KDD dataset.

| Attack Type | No. of Samples = 10 DR (%) | No. Samples = 20 DR (%) |
|---|---|---|
| Neptune | 99.9 | 100 |
| Smurf | 80.0 | 81.9 |
| Tear drop | 90.0 | 93.5 |
| Back | 0.0 | 0.0 |

**Table 5.** Effect of increasing the number of samples on probe attacks in the NSL-KDD dataset.

| Attack Type | No. of Samples = 10 DR (%) | No. Samples = 20 DR (%) |
|---|---|---|
| IP sweep | 99.6 | 99.9 |
| Port sweep | 99.8 | 100 |
| Nmap | 80.0 | 82.0 |
| Satan | 70.7 | 71.3 |

**Table 6.** Effect of increasing the number of samples on DoS attacks in the CIC-IDS2017 dataset.

| Attack Type | No. of Samples = 10 DR (%) | No. Samples = 20 DR (%) |
|---|---|---|
| DoS hulk | 80.0 | 81.1 |
| DoS golden eye | 93.5 | 95.0 |
| DoS http test | 99.1 | 99.8 |
| DoS slow loris | 86.7 | 88.0 |

The more powerful the representations, the smaller the number of training samples required. This concludes that our feature extractor model is able to discover the manifold of optimal representations, which causes the DR to slightly depends on the number of examples.

We also compared our approach with the baseline models, which comprised the DL model and the classical ML, as shown in Figures 4–6. The baseline models were trained on the full dataset.

As can be observed from the results, our approach performed competitively with the baseline models. On all the datasets, our model was on par with, or slightly worse than, the baseline models, despite being trained on few-shot examples.

The second experiment was designed to discover whether it was possible to learn representations that would be useful in detecting any class of attack, when re-trained with few-shot examples of that attack. The results of the experiment are presented in Figures 6 and 7. Our model performed reasonably well in detecting certain classes of attacks. For instance, our model achieved detection rates of 90% and 89% for DoS classes in both the NSL-KDD and CIC-IDS2017 datasets. This shows that our feature extractor still learns some useful representations to detect these classes of attacks. However, it performs poorly in detecting other classes, such as R2L and U2R, from NSL-KDD (Figure 7), where the detection rate was 66% and 0.0%, respectively. Similarly, it failed to achieve good detection rates for both FTP-Patator and SSH-Patator attack classes for the CIC-IDS2017 dataset (Figure 8), scoring 34.8% and 47.6%, respectively. Therefore, as expected, there was a drop in overall performance for our model compared to the results of experiment 1. This was due to the fact that our feature extractor tries to discover a singular representation for identification, based on the different classes of attacks observed in the meta-training stage. However, it is difficult to discover such representations due to the diverse nature of the attacks in the meta-training set. Since attacks differ in purpose and implementation, for instance, the DoS attack class tries to shut down traffic flow to and from a target system, a U2R attack tries to gain access to the system or network, an R2L attack tries to gain access to the remote machine, while attacks such as probe try to get information from a network, we assumed that these attack types are too diverse to allow our feature extractor to learn singular representations for identification that can enable excellent detection when trained with a few examples.

## 5. Conclusions

Network intrusion detection using machine learning methods has been studied for a long time, with many commercial intrusion detection systems (IDSs) using machine learning algorithms as part of their detection engines. However, machine learning-based IDSs are susceptible to false alarm rates, which makes the field an active area of research.

Recently, DL methods have been widely applied in network-based IDSs due to their success in fields such as natural language processing (NLP) and computer vision. However, to achieve a better detection rate, DL methods require sizeable volumes of datasets. Collecting large-scale datasets is non-trivial, especially in the cybersecurity domain where the landscape is constantly changing. Hence, few-shot network intrusion detection is emerging as an alternative to conventional supervised DL methods. The concept is popularly addressed based on a meta-learning paradigm, whereby transferable knowledge is learned in some related tasks using complex optimization techniques, which enables generalization at test time with limited examples.

However, in this paper, we propose a simple framework for few-shot network intrusion detection. Our approach relies on learning powerful representations, and is implemented in two stages. We first train a feature extractor model using discriminative representation learning with a supervised autoencoder, and we then train a classifier on top of the feature extractor, which is able to generalize with a few examples.

To validate our approach, we evaluated our model using two publicly available intrusion detection datasets. Our proposed method achieved excellent detection rates

in detecting mutants of existing attacks. However, though our approach achieves good detection rates for certain classes of attacks in the general anomaly detection scenario it performs poorly for others. This is due to the diverse nature of attacks, and it is difficult to learn singular representations that can enable generalization with only a few examples.

Therefore, based on the results of the experiments conducted, our approach is more suited for detecting specific classes of attack or mutants of an existing attack. In addition, it is safe to say that our model can be used in situations like zero-day attacks, since, even in such a scenario, a few samples of attacks can be obtained, which will be sufficient enough to train our model to detect similar occurrences of the same attack or its variants in the future.

**Author Contributions:** Conceptualization, A.S.I. and U.A.A.; methodology, A.S.I. and U.A.A.; software, A.S.I.; validation, U.A.A. and L.Z.; formal analysis, L.Z.; investigation, L.Z.; resources, L.Z.; data curation, U.A.A.; writing-original draft preparation, A.S.I.; writing-review and editing, U.A.A.; visualization, A.S.I.; supervision, L.Z.; project administration, L.Z., funding acquisition, L.Z. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was supported by the China Scholarship Council (CSC) 2018GXZ021733.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The datasets used in this work were collected by the Canadian Institute of Cybersecurity (CIC) and are publicly available at https://www.unb.ca/cic/datasets, accessed on 24 October 2021.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Scarfone, K.A.; Mell, P.M. *Guide to Intrusion Detection and Prevention Systems (IDPS)*; NIST Special Publication 800–94; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2007. [CrossRef]
2. Kruegel, C.; Toth, T. Using Decision Trees to Improve Signature-Based Intrusion Detection. In *Recent Advances in Intrusion Detection*; Vigna, G., Kruegel, C., Jonsson, E., Eds.; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2820, pp. 173–191. [CrossRef]
3. Mell, P.M.; Hu, V.; Lippmann, R.; Haines, J.; Zissman, M. *An Overview of Issues in Testing Intrusion Detection Systems*; NIST Interagency/Internal Report (NISTIR)—7007; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2003. [CrossRef]
4. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
5. Gamage, S.; Samarabandu, J. Deep learning methods in network intrusion detection: A survey and an objective comparison. *J. Netw. Comput. Appl.* **2020**, *169*, 102767. [CrossRef]
6. Ziai, A. Active Learning for Network Intrusion Detection. In *Data Science. Theory, Algorithms, and Applications*; Verma, G.K., Badal, S., Bourennane, S., Ramos, A.C.B., Eds.; Springer: Cham, Switzerland, 2021; p. 11.
7. Wang, Y.; Yao, Q.; Kwok, J.; Ni, L.M. Generalizing from a Few Examples: A Survey on Few-Shot Learning. *arXiv* **2020**, arXiv:1904.05046. Available online: http://arxiv.org/abs/1904.05046 (accessed on 4 December 2021). [CrossRef]
8. Finn, C.; Abbeel, P.; Levine, S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *arXiv* **2017**, arXiv:170303400. Available online: http://arxiv.org/abs/1703.03400 (accessed on 4 December 2021).
9. Bertinetto, L.; Henriques, J.F.; Torr, P.H.S.; Vedaldi, A. Meta-Learning with Differentiable Closed-Form Solvers. *arXiv* **2019**, arXiv:1805.08136. Available online: http://arxiv.org/abs/1805.08136 (accessed on 4 December 2021).
10. Wang, Y.-X.; Hebert, M. Learning to Learn: Model Regression Networks for Easy Small Sample Learning. In *Computer Vision—ECCV 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer International Publishing: Cham, Switzerland, 2016; Volume 9910, pp. 616–634. [CrossRef]
11. Wang, Y.-X.; Hebert, M. Learning from Small Sample Sets by Combining Unsupervised Meta-Training with CNNs. In Proceedings of the 30th Conference on Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; 9p.
12. Li, Z.; Zhou, F.; Chen, F.; Li, H. Meta-SGD: Learning to Learn Quickly for Few-Shot Learning. *arXiv* **2017**, arXiv:170709835. Available online: http://arxiv.org/abs/1707.09835 (accessed on 6 December 2021).
13. Dhillon, G.S.; Chaudhari, P.; Ravichandran, A.; Soatto, S. A Baseline for Few-Shot Image Classification. *arXiv* **2020**, arXiv:190902729. Available online: http://arxiv.org/abs/1909.02729 (accessed on 6 December 2021).
14. Tian, Y.; Wang, Y.; Krishnan, D.; Tenenbaum, J.B.; Isola, P. Rethinking Few-Shot Image Classification: A Good Embedding Is All You Need? *arXiv* **2020**, arXiv:200311539. Available online: http://arxiv.org/abs/2003.11539 (accessed on 6 December 2021).

15. Ouali, Y.; Hudelot, C.; Tami, M. Spatial Contrastive Learning for Few-Shot Classification. *arXiv* **2021**, arXiv:201213831. Available online: http://arxiv.org/abs/2012.13831 (accessed on 4 December 2021).
16. Gogna, A.; Majumdar, A. Discriminative Autoencoder for Feature Extraction: Application to Character Recognition. *Neural Process. Lett.* **2019**, *49*, 1723–1735. [CrossRef]
17. Du, F.; Zhang, J.; Ji, N.; Hu, J.; Zhang, C. Discriminative Representation Learning with Supervised Auto-encoder. *Neural Process. Lett.* **2019**, *49*, 507–520. [CrossRef]
18. Tsai, C.-F.; Hsu, Y.-F.; Lin, C.-Y.; Lin, W.-Y. Intrusion detection by machine learning: A review. *Expert Syst. Appl.* **2009**, *36*, 11994–12000. [CrossRef]
19. Liao, Y.; Vemuri, V.R. Use of K-Nearest Neighbor classifier for intrusion detection. *Comput. Secur.* **2002**, *21*, 439–448. [CrossRef]
20. Li, W.; Liu, Z. A method of SVM with Normalization in Intrusion Detection. *Procedia Environ. Sci.* **2011**, *11*, 256–262. [CrossRef]
21. Kumar, M.; Hanumanthappa, M.; Kumar, T.V.S. Intrusion Detection System using decision tree algorithm. In Proceedings of the IEEE 14th International Conference on Communication Technology, Chengdu, China, 9–11 November 2012; pp. 629–634. [CrossRef]
22. Farnaaz, N.; Jabbar, M.A. Random Forest Modeling for Network Intrusion Detection System. *Procedia Comput. Sci.* **2016**, *89*, 213–217. [CrossRef]
23. Kuttranont, P.; Boonprakob, K.; Phaudphut, C.; Permpol, S.; Aimtongkhamand, P.; KoKaew, U.; Waikham, B.; So-In, C. Parallel KNN and Neighborhood Classification Implementations on GPU for Network Intrusion Detection. *J. Telecommun. Electron. Comput. Eng.* **2017**, *9*, 29–33.
24. Goeschel, K. Reducing false positives in intrusion detection systems using data-mining techniques utilizing support vector machines, decision trees, and naive Bayes for off-line analysis. In Proceedings of the IEEE SoutheastCon 2016, Norfolk, VA, USA, 30 March–3 April 2016; pp. 1–6. [CrossRef]
25. Malik, A.J.; Shahzad, W.; Khan, F.A. Network intrusion detection using hybrid binary PSO and random forests algorithm: Network intrusion detection using hybrid binary PSO. *Secur. Commun. Netw.* **2015**, *8*, 2646–2660. [CrossRef]
26. Liu, H.; Lang, B. Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey. *Appl. Sci.* **2019**, *9*, 4396. [CrossRef]
27. Alom, M.Z.; Bontupalli, V.; Taha, T.M. Intrusion detection using deep belief networks. In Proceedings of the 2015 National Aerospace and Electronics Conference (NAECON), Dayton, OH, USA, 15–19 June 2015; pp. 339–344. [CrossRef]
28. Chen, H.; Jiang, L. Efficient GAN-based method for cyber-intrusion detection. *ArXiv* **2019**, ArXiv:190402426. Available online: http://arxiv.org/abs/1904.02426 (accessed on 31 May 2021).
29. Min, E.; Long, J.; Liu, Q.; Cui, J.; Chen, W. TR-IDS: Anomaly-Based Intrusion Detection through Text-Convolutional Neural Network and Random Forest. *Secur. Commun. Netw.* **2018**, *2018*, 4943509. [CrossRef]
30. Yin, C.; Zhu, Y.; Fei, J.; He, X. A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *IEEE Access* **2017**, *5*, 21954–21961. [CrossRef]
31. Wang, W.; Sheng, Y.; Wang, J.; Zeng, X.; Ye, X.; Huang, Y.; Zhu, M. HAST-IDS: Learning Hierarchical Spatial-Temporal Features Using Deep Neural Networks to Improve Intrusion Detection. *IEEE Access* **2018**, *6*, 1792–1806. [CrossRef]
32. Al-Qatf, M.; Lasheng, Y.; Al-Habib, M.; Al-Sabahi, K. Deep Learning Approach Combining Sparse Autoencoder With SVM for Network Intrusion Detection. *IEEE Access* **2018**, *6*, 52843–52856. [CrossRef]
33. Rao, K.N.; Rao, K.V.; Prasad Reddyand, P.V.G.D. A hybrid Intrusion Detection System based on Sparse autoencoder and Deep Neural Network. *Comput. Commun.* **2021**, *180*, 77–88. [CrossRef]
34. Le, T.-T.-H.; Kim, Y.; Kim, H. Network Intrusion Detection Based on Novel Feature Selection Model and Various Recurrent Neural Networks. *Appl. Sci.* **2019**, *9*, 1392. [CrossRef]
35. Bertinetto, L.; Henriques, J.F.; Valmadre, J.; Torr, P.; Vedaldi, A. Learning feed-forward one-shot learners. In Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, 5–10 December 2016; 9p.
36. Hindy, H.; Tachtatzis, C.; Atkinson, R.; Brosset, D.; Bures, M.; Andonovic, I.; Michie, C.; Bellekens, X. Leveraging Siamese Networks for One-Shot Intrusion Detection Model. *arXiv* **2021**, arXiv:200615343. Available online: http://arxiv.org/abs/2006.15343 (accessed on 4 December 2021).
37. Xu, C.; Shen, J.; Du, X. A Method of Few-Shot Network Intrusion Detection Based on Meta-Learning Framework. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3540–3552. [CrossRef]
38. Razakarivony, S.; Jurie, F. Discriminative Autoencoders for Small Targets Detection. In Proceedings of the 22nd International Conference on Pattern Recognition (ICPR), Stockholm, Sweden, 24–28 August 2014.
39. Sharafaldin, I.; Lashkari, A.H.; 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 (ICISSP 2018), Funchal, Portugal, 22–24 January 2018; pp. 108–116.
40. 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, Ottawa, ON, Canada, 8–10 July 2009; pp. 1–6. [CrossRef]