A Deep Learning Based Semi-Supervised Network Intrusion Detection System Robust to Adversarial Attacks

SYED. MD. MUKIT RASHID*, Pennsylvania State University, USA
MD. TOUFIKUZZAMAN*, Bangladesh University of Engineering and Technology, Bangladesh
MD. SHOHRAB HOSSAIN, Bangladesh University of Engineering and Technology, Bangladesh

Network intrusion detection systems (NIDS) are used to detect abnormal behavior in network traffic, which is vital for secure communication. Recently, deep learning based solutions have been adopted for NIDS which suffer from two main problems. Most of them are based on supervised learning and cannot utilize the information that can be obtained from unlabeled data. Also, deep learning based methods are shown to be vulnerable to adversarial attacks. In this paper, we propose a novel semi-supervised and adversarially robust deep learning based approach which can utilize both labeled and unlabeled training samples. Our IDS first performs K-Means clustering to soft label part of the unlabeled data and then obtain a decision tree based on labeled and soft labeled samples. It then pretrains an autoencoder based multi-layer perceptron and later learns separate multi-layer perceptrons on each individual leaf of the decision tree. Our results show that the performance of our system is comparable to state-of-the art supervised learning approaches and outperforms existing state-of-the-art semi-supervised NIDS. Furthermore, we have extensively tested the adversarial robustness of our method using the popular blackbox Fast Gradient Sign Method (FGSM) and Generative Adversarial Network based IDSGAN approaches. Comparisons with other state-of-the-art NIDS baselines show that our proposed mechanism provides significantly higher adversarial detection rates, proving the robustness of our system to adversarial attacks.

Additional Key Words and Phrases: Deep Learning, Intrusion Detection, Adversarial Testing, Semi-supervised

1 INTRODUCTION

With the advancement of software technology and the Internet, the number of security issues and threats is increasing by the day. As a result, there is an increasing need to establish systems that can identify and prevent such threats. These systems must identify irregular behavior that potentially identifies an invasion and either passively inform the administrator or actively try to prevent any intrusion. A network intrusion detection system (NIDS) is a system that is used to identify deviations from normal network traffic behavior, thus indicating an invasion or intrusion. It is usually approached as a classification problem. It may be either a binary classification of whether the network behavior is normal or anomalous or a multi-class classification problem that also tries to identify the attack type.

Developing a network intrusion detection system is quite challenging because of different network protocols and variations in attacks. Early NIDS-related works were based on matching attack signatures with traffic packets [1, 8, 13]. The signature-based methods were preceded by traditional machine learning models like Support Vector Machine [4] and different boosting algorithms [11, 22, 23, 26]. Recently, deep learning based models provided high classification accuracies as they can capture complex patterns [17, 19]. Many deep learning based systems have been proposed for intrusion detection based systems, based on convolutional neural networks(CNN)[27, 28], recurrent neural networks[30], autoencoders[12, 24] etc., which provided very high accuracy along with lower false alarm rates.

*Both authors contributed equally to this research.

Authors' addresses: Syed. Md. Mukit Rashid, Department of Computer Science and Engineering, Pennsylvania State University, 201 Old Main, University Park, Pennsylvania, USA, 16802, szr5848@psu.edu; Md. Toufikuzzaman, Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, 1205, toufik@cse.buet.ac.bd; Md. Shohrab Hossain, Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, 1205, mshohrabhossain@cse.buet.ac.bd.

However, these deep learning based systems were based on *supervised* learning. These models suffer from several drawbacks. They are typically trained on labeled datasets. Developing a large dataset where all the samples are labeled is quite difficult. It is a laborious task; experts are required for such annotation, errors can easily occur, and it is also time-consuming. Often a huge amount of data remains unlabeled, and the learning model cannot leverage the information that could be gained from this data. A *semi-supervised* learning method in this scenario would make the most practical sense, as it can extract information from both labeled and unlabeled data. Unfortunately, there are only a few network intrusion detection works based on a semi-supervised learning approach [2, 9, 14] that can utilize both labeled and unlabeled information. Most notably, Ashfaq et al. [2] proposed a self learning approach coupled with fuzziness measure to first learn soft labeling of unlabeled data and further train a single-layer neural network trained with high and low fuzziness data. But the results are somewhat lagging behind compared to more recent state-of-the-art supervised learning approaches. Hara et al. [9] proposed an adversarial autoencoder based semi-supervised method. Another serious issue with deep learning based models is that they are shown to be vulnerable to adversarial attacks [29]. Adversarial examples can be generated to fool a simple deep learning based solution to misclassify an intrusion traffic to be benign. An attacker can thus simulate an equivalent scenario to fool a specific IDS model built based on such a deep learning based solution. There are a few works in the literature to build adversarially robust network intrusion detection systems [10]. However, they are only limited to *supervised* learning based solutions.

In light of the aforementioned issues, we thus propose a *semi-supervised* learning based solution which is robust to adversarial examples. Our approach first performs K-Means clustering [18] on the dataset for soft labeling and also keeps track of clusters whose labeled members are predominantly of a particular attack class. After that, we soft label some of the unlabeled data based on the clustering. Based on the labeled and soft labeled data, we build a decision tree with a fixed number of minimum samples per leaf. Next, we adopt an autoencoder trained on all available training samples to obtain a compact representation of the data. We then pretrain a 3-layer perceptron [5] based on this representation, based on both the original labeled data and newly soft labeled data. Finally, we train separate 3-layer perceptrons on each leaf of the decision tree. While giving output, we determine the decision tree leaf assigned to any payload data and provide class prediction based on the trained neural networks.

This approach provides superior accuracy compared to the state of the art approaches, achieving 80% accuracy in fully supervised learning and 77% accuracy in semi-supervised learning with 50% of the total data labeled on the popular NSL-KDD dataset [6]. We also extensively test the adversarial robustness of our model and compare it with other baselines. We test adversarial robustness using the blackbox FGSM [20] and also the more recent IDSGAN [15] based adversarial attack. These tests prove that using soft-labeled data based on K-Means clustering, along with the decision tree-based mechanism, greatly improves the adversarial robustness of our model over state-of-the-art approaches.

The main contributions of our paper are as follows:

- We propose a *semi-supervised* deep learning based approach that utilizes both labeled and unlabeled data.
- We utilize the inherent clustering information in the payload space to soft label the portion of the unlabeled data to assist in downstream model learning and also increase adversarial robustness.
- We use a decision tree to split up the imbalanced dataset and learn specialized neural networks in the leaves that significantly improves accuracy and adversarial robustness.
- We extensively test the adversarial robustness of our model by generating adversarial samples using the popular blackbox FSGM method and recently developed IDSGAN method and showing the superior adversarial robustness of our model over state-of-the-art intrusion detection systems in semi-supervised learning conditions.

2 METHODOLOGY

The overall pipeline of designing and training and deep learning models is shown in Figure 1. We discuss the different steps of the pipeline in the following sections.

2.1 Dataset

In this study, we have used the popular NSL-KDD dataset [26] for performance comparison and adversarial testing. The NSL-KDD is a widely used benchmark dataset for network intrusion detection systems. This dataset is a rectified version of the KDD Cup '99 dataset [25]. The NSL-KDD dataset has samples of 4 broad attack types ('DoS,' 'Probe,' 'U2R,' and 'R2L') and 'Normal' traffic. The training set of NSLKDD has 125,973 samples, and the test set has 22,544 samples. The 'U2R' and 'R2L' attacks can be considered minority classes, as they only contain 995 and 52 samples in the training set and 2,754 and 200 samples, respectively, in the test set. Many previous works present only 10-fold cross-validation results in which 99%+ accuracy is easily achieved, and the challenges of the test set are not addressed. We thus evaluate the performance of our IDS based on the given separate training and test dataset on a multi-class prediction environment.

To test our method and different baselines in a semi-supervised learning environment, we perform training of our model and different baselines by making a portion of the training dataset's labels available to the model while making the rest of the sample labels unavailable to the model. For fully supervised methods that cannot take advantage of unlabeled samples, we only use the labeled portion.

2.2 Performance Metrics

We report the class-wise F1 score (harmonic mean of precision and recall) and overall accuracy of the model in both supervised and semi-supervised learning scenarios. However, the accuracy measurement is flawed, as a model can achieve high accuracies by only attending to the majority classes; hence we also report the macro average of the F1 score, where each class gets the same share. This amplifies the importance of correctly classifying the minority classes.

The performance metrics are described below.

- **Accuracy:** Accuracy is denoted by the number of correctly classified samples in proportion to the total number of samples. In classification tasks like network intrusion detection number of attack traffic is very low compared to number of normal traffic. Hence a NIDS system which classifies all the samples as most frequently occurring class might achieve good accuracy. Accuracy is defined as the following equation.
$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$
- **Precision:** It is defined as the ratio of the samples that were correctly classified and the total no. of samples that were classified as belonging to this class.Mathematically,
$$Precision = \frac{TP}{TP + FP}$$
- **Recall:** Recall is defined as the ratio of samples that were correctly classified as belonging to one class and the total no. of actual samples belonging to this class.Mathematically,
$$Recall = \frac{TP}{TP + FN}$$

- **F1 Score:** It is the harmonic mean of the precision and recall values of a class.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

2.3 K-Means Clustering

We perform the K-Means clustering algorithm with a fixed number of centroids. We then assign labels to a portion of the unlabeled samples based on the cluster they are part of. We call this process of labelling unlabeled samples using the cluster membership "soft labelling". We assign soft labels to clusters first. A cluster can be assigned to an attack class if more than 50% of its labeled members are of the same attack class, and the class does not contain more than 10% labeled samples of any other class. We observe that if a cluster belongs to an attack class, almost all of its members belong to the same class in most cases, so in the case of many unlabeled samples, this soft labeling is justified. In some cases, the number of labeled samples in a very minor class is assigned to a cluster containing the majority of samples belonging to another attack class. For this case, we restrain from assigning soft labels to a cluster if it contains a sufficient portion of all available labeled samples of a different class. Here we restrain if a minority class has more than 10% samples of all available labeled samples of that class. We assign a cluster as belonging to the normal class if all of its labeled members belong to the normal class. If a cluster can be assigned a soft label, we soft label all the unlabeled members of this cluster as belonging to that particular class. This soft labeling mechanism helps us achieve a high soft labeling frequency along with a high percentage of correctly soft labeled data.

To select the number of centroids, we tested out with different numbers of centroids and finally select the total number of cluster centroids to be $\frac{1}{25}^{th}$ of the training dataset, a total of 5,038 clusters. The algorithm is run for 300 iterations with a tolerance level of $1e-04$.

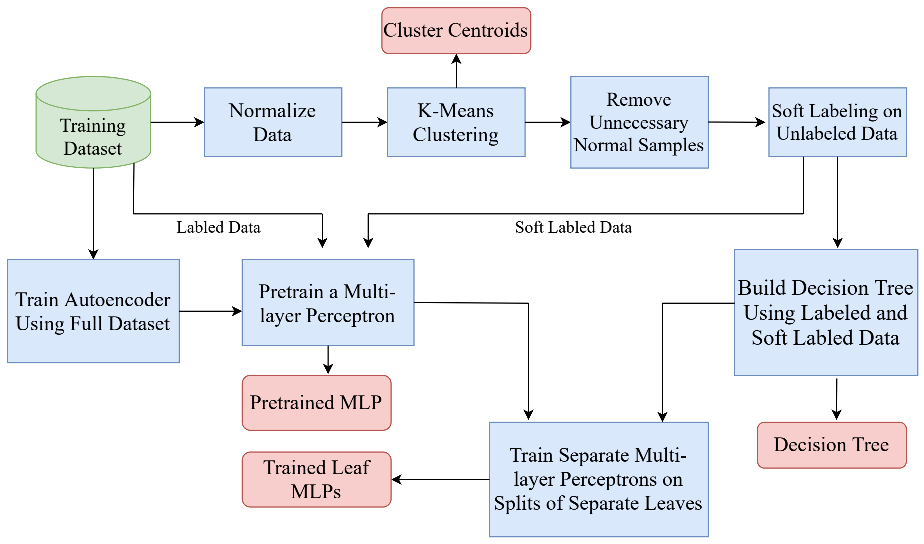

Cluster Centroids
Training Dataset
Normalize Data
K-Means Clustering
Remove Unnecessary Normal Samples
Soft Labeling on Unlabeled Data
Labled Data
Soft Labled Data
Train Autoencoder Using Full Dataset
Pretrain a Multi-layer Perceptron
Build Decision Tree Using Labeled and Soft Labled Data
Pretrained MLP
Decision Tree
Trained Leaf MLPs
Train Separate Multi-layer Perceptrons on Splits of Separate Leaves

Fig. 1. Block Diagram for Training of Our Proposed Methodology

2.4 Building Decision Trees

Based on the labeled and soft labeled data, we build a decision tree using the CART [16] algorithm. We limit the minimum number of samples that must be present in a leaf node of the decision tree. This helps prevent overfitting while training Multi-Layer Perceptron (MLP) models for a particular leaf and also to prevents creating too many MLPs. Using a decision tree in our model has two main benefits: (i) It helps split up the dataset in the payload space so that each individual split could be separately used for training a separate MLP, which, combined with a globally pre-trained MLP helps us achieve much higher accuracy than a single pre-trained MLP itself, and (ii) This mechanism also boosts the adversarial robustness of our dataset, as each MLP corresponding to the leaf node of the tree is only concerned with providing correct classifications for samples only belonging to that particular leaf. This significantly reduces adversarial vulnerability, as we will see in Section 3. We build the decision tree with the minimum number of samples per leaf set to $\frac{1}{50}^{th}$ of the total number of labeled and soft labeled samples.

2.5 Training the Autoencoder

After building the decision tree separately, train an under-complete autoencoder using all the data available, as this learning does not require any labels. Using an autoencoder helps reduce the dimensionality of the input samples, as high dimensional data often causes overfitting and also increases adversarial vulnerability. Thus using an autoencoder reduces overfitting and provides more adversarial robustness to our model. Furthermore, in this way, we can also leverage the structuring information of unlabeled data to obtain better representations of input samples. A high-level block diagram of the autoencoder is shown in Figure 2.

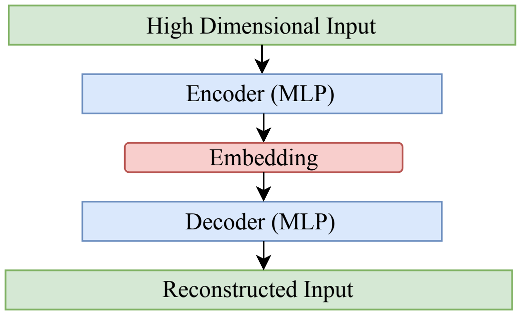


Fig. 2. Block Diagram of an Undercomplete Autoencoder

2.6 Training the Leaf MLPs

We finally train separate 3-layer perceptrons for each split found in the leaves of the decision tree generated. These are used to infer the class of a sample belonging to a particular leaf of the decision tree. We first pre-train a 3-layer perceptron with all labeled and soft labeled samples. These MLPs take the lower dimensional representation given as output by the autoencoder as input and give the final class predictions as output. Pre-training the model helps achieve greater accuracies quickly and is needed as the split sizes in which downstream MLPs corresponding to particular leaves would be trained would be smaller than the full dataset; thus, this pre-training helps avoid overfitting. After pre-training, we then train separate 3-layer perceptrons for each leaf, taking the split of the labeled dataset corresponding to this leaf as the training dataset. We use the pre-trained model parameters for initialization during the training of these MLPs. These MLPs focus on inferring classes for samples that are assigned to a single leaf.

Information about the autoencoder and MLP hyperparameters of our methodology is given in Table 1. We store the model parameters with the minimum training loss. We store the cluster centroids, soft label assignment to clusters, attribute mean and standard deviation information, the decision tree, the trained autoencoder, the pre-trained MLP model, and MLP models per leaf for class inference.

Hyperparameter	**Value**
Encoding Layer Dimensions	96, 64
Decoding Layer Dimensions	64, 96
Autoencoder Embedding Dimension	32
Autoencoder Training Epochs	80
Autoencoder Learning Rate	0.001
MLP Hidden Layer Dimensions	16, 8
MLP Pretraining Learning Rate	0.001
Leaf MLP training Learning rate	0.0005
MLP pre-training epochs	200
Leaf MLP training epochs	200

Table 1. Values of Different Hyperparameters for Implementation of Our Approach

2.7 Class Inference

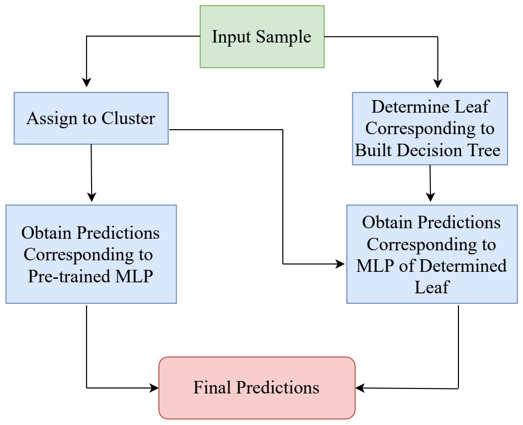

Input Sample
Assign to Cluster
Determine Leaf Corresponding to Built Decision Tree
Obtain Predictions Corresponding to Pre-trained MLP
Obtain Predictions Corresponding to MLP of Determined Leaf
Final Predictions

Fig. 3. Class Inference Using Our Proposed Method

The class inference method is depicted in Figure 3. To infer outputs, we first find out the cluster the sample is assigned to and also the leaf of the generated decision tree it is assigned to. We then use the MLP corresponding to the leaf of the sample and the globally pre-trained MLP to generate the output. If the cluster the sample is assigned to corresponds to a known attack class, we recognize the sample as attack data and zero out the probability of the sample being benign in both the pre-trained and leaf MLP-predicted probabilities. We take the class with the maximum probability score assigned to the class prediction of an MLP. We take into consideration both the prediction of the pre-trained MLP and the leaf MLP. We prioritize the minority classes having less than 10% of training data. For example, the NSL-KDD dataset has two such classes, namely *U2R* and *R2L*, with *U2R* having the lowest no. of samples in the training data. We infer the sample belonging to class *U2R* if any one of the two aforementioned MLPs predicts class *U2R*. If not, we predict class *R2L* if any one of the MLPs predicts *R2L*. We serially go in such way in order of the portion of samples a minority class had in the training dataset. If none of the above cases match, we output the prediction of the leaf MLP. This is done so as we observe that for classes that had sufficient samples in the training dataset, the leaf MLP provides better accuracy than the globally pre-trained MLP, but leaf models do not get sufficient samples to accurately detect minority class samples, where the pre-trained MLPs excel compared to the leaf MLPs. Our inference mechanism provides the best of both worlds.

2.8 Code, Environment, and Availability

All the experiments were designed using the PyTorch deep learning library [21]. The experiments were performed on Nvidia Titan X GPU with 12 GB of VRAM and Google Colab. The dataset and source codes are available at https://github.com/tzpranto/NIDS_NSYSS_23.

3 EXPERIMENTS AND RESULTS

In this section, we discuss our experimental analysis and findings in detail. We implement various baseline models from previous studies to compare with our work. Besides, to justify the choice of different components of our method, we train and test the stand-alone components of our combined model and their various combinations as separate models and compare with our our proposed approach (i.e. the combined model). For a representative of a recent state-of-the-art semi-supervised learning based technique, we consider the work of Asfaq et al. [2], where they proposed a fuzziness-based semi-supervised learning(Fuzziness-SSL). They reported the performance on binary classification. We reproduce their model for multiclass classification. We adopt the NN with 80, 40, and 20 hidden nodes as this configuration performed best on their methodology. As a representative of the state-of-the-art deep learning technique, we reproduce the RNN-based supervised methodology proposed by Yin et al. [30]. The best-performing hyperparameters (80 nodes in the hidden layer and 0.5 learning rate) reported in their paper have been used to design the model.

Besides these baselines, we also consider simple learning based techniques such as MLPs, K-Means, and decision trees, which are independent components of our proposed approach. We first consider a deep neural network with three hidden layers as a simple and intuitive baseline. The number of neurons on the hidden layers is 60, 30, and 15 accordingly. Next, in the *K-Means* model, all the training samples are taken, and we execute the K-Means algorithm. After that, for all clusters that contain at least one labeled sample, we consider a cluster's label to be the one majority of its labeled members have. In the decision tree models, we take all the labeled training samples and form a decision tree. A sample is predicted as the same label as the leaf it gets assigned to. We consider two variants of decision trees; in one, we only use the labeled training samples and completely discard the unlabeled ones, and in another, we also add soft labeled samples labeled through the K-Means procedure. The soft labeling mechanism is the same as in our model. We denote them as with/without soft labeling.

3.1 Performance Comparisons

We show the F1 scores of all classes, macro-F1 score, and accuracy for 100% labeled data in Table 2 and for 50% labeled data in Table 3. A graphical representation of the results is shown in Figure 4. We evaluate the models on the test set (KDDTest+ split) of the NSL-KDD dataset.

Table 2. Performance comparison of different models on 100% labeled data

Model	**F1 Score**						**Accuracy**
	Normal	**DoS**	**Probe**	**R2L**	**U2R**	**Macro Avg.**	
DNN	0.76	0.81	0.74	0.00	0.00	0.46	0.72
RNN	0.75	0.80	0.71	0.00	0.00	0.45	0.71
Fuzziness Based IDS	0.81	0.89	0.72	0.06	0.16	0.53	0.77
K-Means	0.80	0.88	0.71	0.06	0.17	0.53	0.77
DT w/o soft labelling	0.79	0.87	0.70	0.16	0.13	0.53	0.76
KM + AE + DNN (Full IDS - (DT and leaf procedure))	0.80	0.88	0.79	0.27	0.09	0.57	0.78
KM + DT + DNN (Full IDS - AE)	0.83	0.84	0.66	0.36	0.00	0.54	0.77
Full IDS	0.83	0.89	0.81	0.27	0.07	0.57	0.80

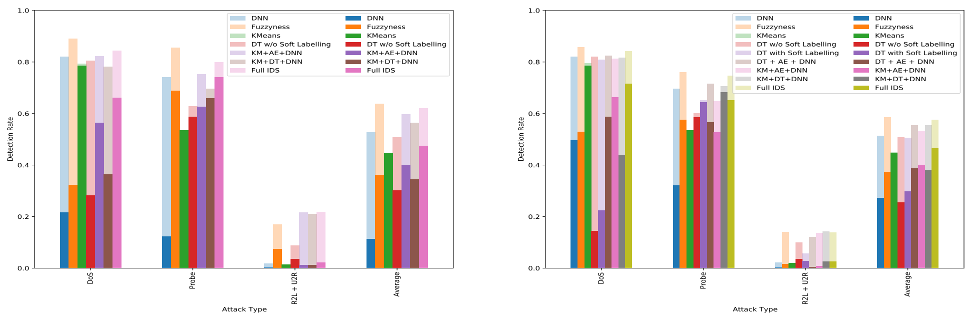
Table 3. Performance comparison of different models on 50% labeled data

Model	**F1 Score**						**Accuracy**
	Normal	**DoS**	**Probe**	**R2L**	**U2R**	**Macro Avg.**	
DNN	0.75	0.80	0.71	0.00	0.00	0.45	0.71
RNN	0.75	0.81	0.78	0.00	0.00	0.47	0.72
Fuzziness Based IDS	0.78	0.84	0.71	0.14	0.06	0.51	0.75
K-Means	0.79	0.88	0.73	0.06	0.18	0.53	0.76
DT w/o soft labelling	0.79	0.88	0.70	0.18	0.07	0.53	0.77
DT with soft labelling	0.80	0.88	0.70	0.11	0.08	0.51	0.76
DT + AE + DNN (Full IDS - KM Soft Labeling)	0.81	0.88	0.77	0.22	0.03	0.54	0.78
KM + AE + DNN (Full IDS - (DT and leaf procedure))	0.80	0.86	0.71	0.23	0.14	0.55	0.77
KM + DT + DNN (Full IDS - AE)	0.80	0.88	0.66	0.23	0.14	0.54	0.76
Full IDS	0.80	0.89	0.72	0.23	0.15	0.56	0.77

We observe that our NIDS model outperforms all other baselines and variants both in terms of accuracy and macro-F1 score. Our model achieves 80% accuracy in a supervised learning scenario and 77% accuracy in a semi-supervised learning scenario with 50% labeled data. We also observe that compared to the semi-supervised learning method by [2], our model specially performs well for minority classes U2R and R2L, achieving F1 scores of 0.23 and 0.16 compared to 0.14 and 0.06 of that of Fuzziness based IDS by [2]. We also observe that for all classes, our NIDS achieves respectable accuracies, outperforming all except the KM+DT+DNN variant in U2R class in the supervised learning scenario and for the benign class in the semi-supervised learning scenario for the DT+AE+DNN variant.

3.2 Justification of Different Components of Our Model

We now analyze our model by comparing different variants and justify the use of different components of our model. We generate results for different combinations of sub-modules of our model. The hyperparameters used for the variations of our NIDS model are the same used in our model. In the *DT+AE+DNN* variant, we do not use the soft labeling through K-Means, but all the other architecture and hyperparameters of this model are the same as ours. Also, during inference, cluster information is not used, and only the predictions of the pretrained global MLP and the leaf MLP are used. Comparison with this variant shows the effectiveness of soft labeling more samples through the K-Means algorithm. Note that for the fully supervised scenario, it is the same as our model, as no soft labeling is required, so we do not show this variant for the experiments with 100% label. In the *KM+AE+DNN* variant, we exclude the decision tree mechanism



(a) 100% Label

(b) 50% Label

Fig. 4. Comparison of adversarial robustness using adversarial attacks generated by IDSGAN model. Light bars indicate original detection rates, dark bars indicate adversarial detection rate.

Table 4. Adversarial Performance comparison of different models using Blackbox FGSM

Model	**100% Labeled Data**			**50% Labeled Data**		
	Original	**Substitute**	**Adversarial**	**Original**	**Substitute**	**Adversarial**
DNN	0.758	0.728	0.339	0.746	0.735	0.324
RNN	0.742	0.719	0.636	0.739	0.736	0.49
Fuzziness Based IDS	0.811	0.764	0.48	0.784	0.761	0.51
KMeans	0.801	0.797	0.608	0.78	0.714	0.611
DT w/o soft labelling	0.779	0.779	0.036	0.785	0.761	0.391
DT with soft labelling	–	–	–	0.792	0.753	0.101
DT + AE + DNN (Full IDS - KM Soft Labeling)	–	–	–	0.806	0.784	0.507
KM + AE + DNN (Full IDS - (DT and leaf procedure))	0.804	0.793	0.725	0.804	0.793	0.673
KM + DT + DNN (Full IDS - AE)	0.84	0.795	0.728	0.807	0.806	0.641
Full IDS	0.831	0.801	0.763	0.799	0.754	0.683

of our model. The output is predicted through the globally pre-trained model in our NIDS. Comparison with this variant in terms of performance and adversarial robustness justifies the decision tree mechanism used in our NIDS. Finally, we consider the *KM+DT+DNN* variant, where we omit the use of autoencoders. In this model, the MLPs directly take the full high-dimensional sample as input. Here we use 4-layer perceptrons, with the hidden layers containing 64, 32, and 8 neurons, respectively. Comparison with this variant justifies the use of autoencoders. The results are shown in Table 2 and 3.

Compared to K-Means and decision trees, our model performs better for Probe, U2R, and R2L classes, especially the Probe class. Stand-alone K-Means and decision trees fail to perform well for minority classes, and extra mechanisms are needed to improve the detection rates of these classes. Compared to *KM+AE+DNN*, our model achieves superior F1 scores in the Normal class and also for DoS and Probe attacks, i.e., the majority classes, while being equal for U2R class and marginally behind for R2L class, resulting in a slight lead in terms of accuracy. But our model, in particular, excels compared to this variant in terms of adversarial robustness, as we will see in Section 4. We also observe that the usage of the autoencoder significantly improves accuracy, especially the detection rate of the Probe class. All the variants involving autoencoders achieve an F1 score of over 0.75 in this class in supervised learning conditions, while the variants not using autoencoders: KM+DT+DNN, Decision trees, K-Means, DNN, RNN, and Fuzziness based NIDS do not. The reduction to lower dimensions assists in achieving this accuracy. Also, if we compare the performance of DT+AE+DNN to that of our model, we observe that our model is marginally ahead in terms of U2R and R2L classes, which proves soft labeling marginally improves accuracy. But it greatly improves the adversarial robustness of models, as we will see in the next Section. Overall, soft labeling and the decision tree mechanism help improve both accuracy and adversarial robustness, while using an autoencoder largely improves accuracy.

4 ADVERSARIAL ROBUSTNESS TESTING

Adversarial attacks on NIDS models is carried out by clever manipulation the of network traffic. This subtle perturbation in input features of the network traffic can fool the machine learning model into misclassification [3, 7]. Hence, adversarial testing is necessary for intrusion detection systems to assess their robustness and effectiveness by simulating real-world attack scenarios and identifying potential vulnerabilities. Due to the importance of adversarial testing, we have dedicated a separate section to discuss the adversarial robustness of our model and compare it with the baselines. We test with two different adversarial attacks: the popular Blackbox FGSM attack [20] and the more recent IDSGAN-based attack [15].

4.1 Testing with Blackbox FGSM Attack

We first perform the blackbox FGSM attack [20] on our proposed model and also the baselines. Here, for the substitute DNN model, we use a 4-layer deep neural network, with the hidden layers containing 128, 64, and 32 neurons, respectively. We initially used 150 samples obtained from the test set, with σ set to 10 and ρ set to 6. The perturbation constant λ is set to 0.3, and λ is set to 0.1. We report the accuracy of the model in terms of binary prediction on the rest of the test dataset. We report the accuracy of the target NIDS model, the accuracy of the substitute model, and the adversarial accuracy of the target NIDS model as obtained through the blackbox FGSM attack. The results are shown in Table 4.

We observe that for both supervised and semi-supervised learning scenarios, our model outperforms other baselines and variants in terms of adversarial accuracy, proving the adversarial robustness of our model. Interestingly, we observed that K-Means, and the variants that adopted soft labeling to increase the size of the training dataset, achieved greater adversarial robustness than other models. Our model achieves 61.7% and 36% higher adversarial accuracy than Fuzziness-based NIDS in supervised and semi-supervised learning scenarios, respectively. In the adversarial attack where there are no constraints on features, we see that the adversarial accuracy of decision trees itself is quite low, but incorporated into our model as a component, it helps achieve greater adversarial accuracy, as it achieves 5.24% greater accuracy than the KM+AE+DNN variant in supervised learning and 1.49% greater accuracy in 100% and 50% labeled data respectively. This observation is in line with the explanation mentioned in [7], where it was found that the linearity of bare neural network models leaves them adversarially vulnerable, but as our inference mechanism also takes into account the cluster assignment of the test sample, this introduces some non-linearity which assists in increasing adversarial robustness. Also, for the DNN-based model, we see a lesser adversarial accuracy for the same reason.

4.2 Testing with IDSGAN Attack

In the work by Papernot et al. [20], although it works for a blackbox model where the model parameters and training dataset need not be available for the attacker, one potential problem it has that it does not consider the constraints of the attack input features w.r.t the attack class it belongs to. Recently Lin et al. [15] proposed the IDSGAN, where the generator of the model produced an adversarial sample of the input attack through perturbation but only perturbing the non-functional features of the input so that the nature of the attack remains unchanged. However, the IDSGAN suffers from some instability, as reported in [29]. For this reason, we obtain the adversarial accuracy of the target NIDS model after each epoch and report the minimum adversarial accuracy obtained after any epoch. This better test the adversarial robustness of the system. As mentioned in [15], we train and test for separate attack traffic and report for each. We show the results for both supervised and semi-supervised learning scenarios. For each case, half of the benign training data points and the considered attack data points were used to train the IDSGAN. The noise size was set to 27, the generator was trained for 20 iterations, and the discriminator to 1 iteration per epoch. The IDSGAN was trained for five epochs for all tests. The results are shown in Figure 4.

We see that our NIDS achieves higher adversarial accuracy on average for both 100% and 50% labeled data. Again as in blackbox FGSM, we observe that models incorporating soft labeling achieve higher adversarial accuracy, which proves our claim that the cluster assignment during inference helps achieve adversarial robustness. We observe that the K-Means algorithm is fairly robust to adversarial samples, especially in DoS-based attacks, but somewhat suffers in other minority classes. On the other hand, decision tree-based methods are adversarially robust to Probe attacks when the functional features are kept unmodified but hugely suffer in DoS, U2R, and R2L-based attacks. Also, we see the use of autoencoders help achieve higher adversarial detection rates, as our NIDS outperforms the KM+DT+DNN variant by 21% in the supervised learning setting and by 6.35% in the semi-supervised learning setting. This proves that the use of autoencoders also somewhat helps in increasing adversarial robustness alongside the cluster assignment during inference and the use of decision trees.

5 CONCLUSION

In this paper, we propose a novel semi-supervised learning algorithm that performs K-Means clustering, builds a decision tree and autoencoder, and obtains MLPs at each individual leaf of the decision tree to achieve higher accuracy in a semi-supervised scenario and achieve superior adversarial robustness compared to state-of-the-art semi-supervised learning algorithms. We observed that the soft labeling based on K-Means clustering and also using that information to infer classes significantly improves adversarial accuracy testing on the popular blackbox FGSM and IDSGAN models. Versatile datasets, adversarial training approach and scalability in practical scenario could be explored in future. Besides, we intend to incorporate recent deep learning techniques, e.g., few-shot learning, and observe if this improves accuracy in the minority classes while maintaining adversarial robustness, as there is still room for improvement in the accuracy of these classes.

REFERENCES

[1] Farooq Anjum, Dhanant Subhadrabandhu, and Saswati Sarkar. 2003. Signature based intrusion detection for wireless ad-hoc networks: A comparative study of various routing protocols. In *IEEE 58th Vehicular Technology Conference (VTC-2003-Fall*, Vol. 3. IEEE, Orlando, Florida, USA, 2152–2156.

[2] Rana Aamir Raza Ashfaq, Xi-Zhao Wang, Joshua Zhexue Huang, Haider Abbas, and Yu-Lin He. 2017. Fuzziness based semi-supervised learning approach for intrusion detection system. *Information Sciences* 378 (2017), 484–497.

[3] Nicholas Carlini and David Wagner. 2017. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*. ACM, Dallas, TX, USA, 3–14.

[4] Nello Cristianini and John Shawe-Taylor. 2000. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, Cambridge.

[5] George Cybenko. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems* 2, 4 (1989), 303–314.

[6] L Dhanabal and SP Shantharajah. 2015. A study on NSL-KDD dataset for intrusion detection system based on classification algorithms. *International journal of advanced research in computer and communication engineering* 4, 6 (2015), 446–452.

[7] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *3rd International Conference on Learning Representations, ICLR 2015*, Yoshua Bengio and Yann LeCun (Eds.). IEEE, San Diego, CA, USA. http://arxiv.org/abs/1412.6572

[8] Bart Haagdorens, Tim Vermeiren, and Marnix Goossens. 2004. Improving the performance of signature-based network intrusion detection sensors by multi-threading. In *International Workshop on Information Security Applications*. Springer, Jeju Island, Korea, 188–203.

[9] Kazuki Hara and Kohei Shiomoto. 2020. Intrusion Detection System using Semi-Supervised Learning with Adversarial Auto-encoder. In *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, Budapest, Hungary, 1–8.

[10] Mohammad J Hashemi and Eric Keller. 2020. Enhancing Robustness Against Adversarial Examples in Network Intrusion Detection Systems. In *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, Leganes - Madrid, Spain, 37–43.

[11] Hui Jiang, Zheng He, Gang Ye, and Huyin Zhang. 2020. Network Intrusion Detection Based on PSO-Xgboost Model. *IEEE Access* 8 (2020), 58392–58401.

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

[13] Vinod Kumar and Om Prakash Sangwan. 2012. Signature based intrusion detection system using SNORT. *International Journal of Computer Applications & Information Technology* 1, 3 (2012), 35–41.

[14] V Valli Kumari and P Ravi Kiran Varma. 2017. A semi-supervised intrusion detection system using active learning SVM and fuzzy c-means clustering. In *International Conference on IoT in Social, Mobile, Analytics and Cloud (I-SMAC)*. IEEE, Palladam, Tamil Nadu, India, 481–485.

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

[16] Wei-Yin Loh. 2011. Classification and regression trees. *Wiley interdisciplinary reviews: data mining and knowledge discovery* 1, 1 (2011), 14–23.

[17] Simone A Ludwig. 2017. Intrusion detection of multiple attack classes using a deep neural net ensemble. In *IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, Honolulu, HI, USA, 1–7.

[18] James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. University of California Press, Oakland, CA, USA, 281–297.

[19] Nour Moustafa, Jiankun Hu, and Jill Slay. 2019. A holistic review of network anomaly detection systems: A comprehensive survey. *Journal of Network and Computer Applications* 128 (2019), 33–55.

[20] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*. ACM, Abu Dhabi, United Arab Emirates, 506–519.

[21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[22] Muhammad Shakil Pervez and Dewan Md Farid. 2014. Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs. In *8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*. IEEE, United States, 1–6.

[23] Ngoc Tu Pham, Ernest Foo, Suriadi Suriadi, Helen Jeffrey, and Hassan Fareed M Lahza. 2018. Improving performance of intrusion detection system using ensemble methods and feature selection. In *Proceedings of the Australasian Computer Science Week Multiconference*. ACM, Brisbane, QLD, Australia, 1–6.

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

[25] Salvatore Stolfo, Wei Fan, Wenke Lee, Andreas Prodromidis, and Philip Chan. 1999. KDD Cup 1999 Data. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C51C7N.

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

[27] R Vinayakumar, Mamoun Alazab, KP Soman, Prabaharan Poornachandran, Ameer Al-Nemrat, and Sitalakshmi Venkatraman. 2019. Deep learning approach for intelligent intrusion detection system. *IEEE Access* 7 (2019), 41525–41550.

[28] R Vinayakumar, KP Soman, and Prabaharan Poornachandran. 2017. Applying convolutional neural network for network intrusion detection. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, Udupi (Near Mangalore), India, 1222–1228.

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

[30] Chuanlong Yin, Yuefei Zhu, Jinlong Fei, and Xinzheng He. 2017. A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access* 5 (2017), 21954–21961.