Journal of Information Security and Applications 71 (2022) 103348

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

ELAT: Ensemble Learning with Adversarial Training in defending against evaded intrusions

Ying-Dar Lin[a], Jehoshua-Hanky Pratama[a], Didik Sudyana[a], Yuan-Cheng Lai[b], Ren-Hung Hwang[c], Po-Ching Lin[d,*], Hsuan-Yu Lin[e], Wei-Bin Lee[f], Chen-Kuo Chiang[d]

[a] National Yang Ming Chiao Tung University, Hsinchu, 300, Taiwan
[b] National Taiwan University of Science and Technology, Taipei, 106, Taiwan
[c] National Yang Ming Chiao Tung University, Tainan, 711, Taiwan
[d] National Chung Cheng University, Chiayi, 621, Taiwan
[e] Telecom Technology Center, Taiwan
[f] Foxconn Research, Taiwan

ARTICLE INFO

Keywords:
Adversarial attack
Machine learning
Intrusion detection
Ensemble learning

ABSTRACT

Network intrusion detection systems (NIDSs) now adopt machine learning (ML) for detection of wide attack variants. However, ML is also known vulnerable to adversarial attacks, which can degrade the accuracy of ML. A number of defense strategies have been proposed but mostly in image classification areas. In this work, we propose Ensemble Learning with Adversarial Training (ELAT) to combine adversarial training and ensemble learning into a solution. We compare four approaches: single, ensemble, adversarial and ELAT. In the experiments, several models were developed and tested using different approaches to know which method is robust against adversarial attacks for ML-based NIDSs. The average F1 score for the single models was 0.93 within a wide range (0.82-0.99), but dropped to 0.29 when facing adversarial attacks, particularly dropped to 0.07 caused by the strongest attack, Projected Gradient Descent (PGD). With ensemble, adversarial and ELAT, the average scores were recovered to 0.80, 0.88 and 0.91, respectively. In addition, this work involves prediction of the models and approach implemented behind the system using cosine similarity with an accuracy of 99.9%.

1. Introduction

Cybersecurity has become essential because of the continuous growing threat of cybersecurity attacks. As hacking techniques become more advanced, there are no limits for cyber threats even after applying security rules and policies [1]. To defend a network from cyber threats, a Network Intrusion Detection System (NIDS) is introduced to detect malicious network traffic and alert the system administrator if hackers attempt to attack a network. A NIDS conventionally looks for known attack signatures in network traffic or detects outliers from normal network usage. However, such approaches do not seem very promising since they may be unable to detect attack variants. Conventional approaches rely only on human knowledge and analysis, which may be error-prone. Therefore, NIDS research has started to adopt machine learning (ML) to detect variants of malicious network traffic [2]. Not only can ML detect most variants of malicious network traffic, but it can also ensure the accuracy of prediction because it eliminates human errors when analyzing data. However, advanced hackers can see such ML adoption as an opportunity to launch adversarial attacks, which perturb the input features by adding noise to the input and therefore can delude an ML-based NIDS. Adversarial attacks in a network domain can be very powerful because after an attacker succeeds in deluding the ML system, he/she can attack the network behind the NIDS, called a ‘double attack’ [3]. Because of the vulnerability of an ML-based NIDS to adversarial attacks, it becomes necessary to secure both the ML models and the network so as to have a robust NIDS.

Adversarial attacks on AI

There are two types of adversarial attacks, poison and evasion attacks. A poison attack is where a training dataset is attacked and then used to train the model; the trained model will then be inaccurate. An evasion attack is an attack that perturbs the malicious input samples and therefore causes an ML-based NIDS to misclassify them as benign

* Corresponding author.
E-mail addresses: ydlin@cs.nctu.edu.tw (Y.-D. Lin), jehoshua.cs09@nycu.edu.tw (J.-H. Pratama), didik.cs08@nycu.edu.tw (D. Sudyana), laiyc@cs.ntust.edu.tw (Y.-C. Lai), rhhwang@nycu.edu.tw (R.-H. Hwang), pclin@cs.ccu.edu.tw (P.-C. Lin), hylin@ttc.org.tw (H.-Y. Lin), wei-bin.lee@foxconn.com (W.-B. Lee), ckchiang@cs.ccu.edu.tw (C.-K. Chiang).

https://doi.org/10.1016/j.jisa.2022.103348

Available online 1 November 2022

[4]. Both types of attacks have the ability to transfer an attack in which the adversarial examples generated from one model can be used to successfully fool another [5]. This ability is called the transferability property. A stronger defense approach is thus needed to defend against an adversarial attack and its transferability property.

Adversarial defense

A variety of methods to defend against adversarial attacks have been proposed. In essence, there are three types of adversarial defense techniques that have been explored in ML-based NIDSs. The techniques include model configuration, adversarial training, and ensemble learning. In model configuration, a robust model is achieved by adjusting the hyper-parameters of a classifier. In adversarial training, the model is trained with adversarial samples, so that it is smart enough to detect adversarial attacks injected into a dataset [6]. Ensemble learning gathers multiple models into a group and makes the decision from the agreement between the models. Since there are a diversity of models to choose, an ensemble of possible models will be more reliable than any single model [7].

Combinations for adversarial defense

In this work, we consider combining adversarial training with the ensemble method, which we refer to it as the Ensemble Learning with Adversarial Training (ELAT) mechanism. Such kind of combination has already been tested by [8] for images and by [9] for network intrusion detection, and the performance of the defense method has been found to be good for defending against adversarial attacks. As a result, we want to further study the approach to find the most effective combination of detection models with the options of ensemble learning and adversarial training under consideration. Furthermore, the transferability property of an adversarial attack is also evaluated. From two methods of defending adversarial attacks noted above, ensemble learning and adversarial training, there are three possible approaches in the defense combination, being (1) ensemble learning, (2) adversarial training, and (3) ensemble learning with adversarial training. The three approaches work as their names suggest. We also consider single models without defenses as the baseline models for comparison when evaluating the effectiveness of the defense approaches. To evaluate the robustness of these techniques to counter adversarial attacks, we must first create an adversarial dataset in which the data have been attacked. The creation needs both the models and the adversarial attacks to test the proposal.

The research addresses the following three issues. (1) *To identify the most effective defense approach for NIDSs*: The goal is to determine which defense approach is the most effective among our evaluated ones against adversarial attack functions. The determination is important because many ML-based models have not incorporated any adversarial attack prevention techniques. (2) *To create the adversarial dataset for NIDSs*: Most studies evaluated ML models using a public dataset; however, there are a very limited number of public datasets that contain adversarial samples. To provide adversarial samples to evaluate ML models and carry out adversarial training, injecting adversarial attacks into a dataset needs to be carried out. (3) *To predict the model of an ML-based NIDS based on the results of adversary attacks*: Knowing the approach that the model behind a system is both useful and challenging. It is useful because a better picture of attacking and defending an ML-based NIDS can be achieved if knowledge of the ML method behind is available. It is challenging because there are many ways to create an ML-based NIDS with different ML algorithms.

Approaches to evaluation of adversarial defense

We select the following five classifiers to evaluate the ELAT method: Decision Tree (DT), Extreme Gradient Boosting (XGB), Logistic Regression (LR), Support Vector Machine (SVM) and Deep Neural Network (DNN). Note that the classifiers in an ML-based NIDS and for different types of attacks may vary. However, these classifiers in our evaluation are very common and fundamental. Therefore, we believe the observations from them can reflect the cases of ML-based NIDSs that are based on these typical models. To carry out an attack on an ML-based IDS, adversarial datasets need to be generated. We cover six attack techniques, namely, Decision Tree Attack (DTA) [10], Fast Gradient Sign Method (FGSM) [3], Projected Gradient Descent (PGD) [11], Carlini and Wagner (C&W) [12], Zeroth Order Optimization Attack (ZOO Attack) [13] and Jacobian-based Saliency Map Attack (JSMA) [14] to craft the adversarial samples in the evaluation. Readers are referred to good tutorials such as [15] for a quick overview of these attack techniques. We select these classifiers and attack techniques because they are well known and quite common. Each of them also has its own uniqueness. For example, DTA is selected because it can attack tree-based classifier that is specific to DT. FGSM is selected because of its speed in generating adversarial attack with a high attack success rate. PGD is selected because it is an improvement of FGSM for its feature of projection; however, it is slower than FGSM in terms of the adversarial sample generation. JSMA is selected because its uniqueness of using Jacobian maps to perturb. C&W is one of the strongest adversarial attack techniques. ZOO is selected because its ability to attack without knowing any gradient information.

In the threat model, we assume all the attacks are white-box (i.e., knowing the details of the classifiers), and each adversarial attack needs to be crafted for a specific classifier; however, only those classifiers that are supported by an adversarial attack can be used. Moreover, we assume the perturbations in the attacks are applied to the feature space; thus, an attacker needs to know the features used in a NIDS to craft the raw traffic that will be transformed to the features with the desired perturbations. If the attacker does not have such knowledge (e.g., fail to reverse engineer the NIDS), the adversarial attacks may not work exactly as expected. This restriction is assumed because this work is intended to study the effective models against adversarial attacks and how to craft raw traffic to generate the perturb features with the perturbations is beyond the scope of this work.

To create an ensemble model efficiently, we propose the use of double fault and kappa statistics as the measurement score to filter out those classifiers that do not meet the requirements for part of an ensemble model. Kappa statistics gather the models that have good agreement with each other to ensure the inter-classifier reliability, while double fault filters those combinations of classifiers that share the same errors to ensure the diversity among the classifiers. We select the two measurement scores because they can be easily computed and the parameters in the computation are ready from the testing (see Eqs. (1) and (2) in Section 2).

The scores of all the approaches in the experiments will be collected and stored in a database as a reference for model predictions, in which cosine similarity will be used to compare the vectors of scores with two metrics, F1 scores and Area Under Curve (AUC), and determine whether the model behind the system being tested is the same as one of the models referred to in the database [16].

This work aims to answer the following essential questions: (1) Which is the strongest adversarial attack function that can perturb the attacks to evade an ML-based NIDS? (2) Which ML algorithm is robust against an adversarial attack? (3) To what extent does the transferability property of an adversarial attack degrade the performance of an ML-based NIDS? (4) How can an ML-based NIDS be made more robust against any adversarial attack? (5) How can we know the ML model that an ML-based NIDS uses?

The rest of this paper is organized as follows. In Section 2, we review the related works. In Sections 3 and 4, the problem statement and solution design are described. In Section 5, we describe our experimental results, and conclusions and future work are given in Section 6.

2. Related work

Table 1 lists 11 papers that have explored defending adversarial attacks to a NIDS, and each has its own way of defense. The second column lists three adversarial defense techniques: model configuration,

Table 1
Related work (FGSM: Fast Gradient Sign Method, BIM: Basic Iterative Method, PGD: Projected Gradient Descent, C&W: Carlini and Wagner, JSMA: Jacobian-based Saliency Map Attack, NES: Natural Evolution Strategies, ZOO: Zeroth Order Optimization, RF: Random Forest, DT: Decision Tree, MLP: Multi-layer Perceptron, SVM: Support Vector Machine, GAN: Generative Adversarial Network, FNN: Feedforward Neural Networks, SNN: Self-normalizing Neural Network, ANN: Artificial Neural Network, DNN: Deep Neural Network, CNN: Convolutional Neural Network, C-LSTM: CNN with Long Short-Term Memory, LR: Logistic Regression).

Paper	Defense technique	Attack techniques	Classifiers	Diversity area			Diversity measurement	Transferability property
				Training	Model	Decision		
[17]	Model configuration	FGSM, BIM, PGD	FNN and SNN	V	–	–	–	–
[18]	Adversarial training	FGSM, BIM, C&W, PGD	RF and Nearest Neighbor	V	–	–	–	–
[19]	Adversarial training	C&W, FGSM, BIM, PGD, Deepfool	ANN and RF	V	–	–	–	–
[20]	Adversarial training	JSMA	RF and J48	V	–	–	–	–
[21]	Adversarial training	FGSM, BIM, DeepFool and JSMA	DNN	V	–	–	–	–
[22]	Adversarial training	GAN	DNN	V	–	–	–	–
[23]	Ensemble model	Alter some features	RF	–	V	–	–	–
[24]	Ensemble model	FGSM, JSMA, C&W, Deepfool, BIM and PGD	SVM, DT, DNN with voting	–	V	V	–	–
[25]	Ensemble model	Extending flow duration, Adding junk data	RF, MLP, DT, AdaBoost Wide and Deep	–	V	V	–	–
[26]	Ensemble model	Keyword manipulation	SVM	V	V	V	–	–
[9]	Adversarial training & ensemble team	HopSkipJumpAttack Pointwise, NES, Boundary Opt-Attack	MLP, CNN, C-LSTM	V	V	V	–	–
Ours	Adversarial training & ensemble team	DT Attack, JSMA, FGSM, PGD, C&W, ZOO Attack	DT, SVM, XGBoost, LR, DNN	V	V	V	Kappa statistics & double fault	V

adversarial training and ensemble model. These papers also explored multiple adversarial attack techniques and classifiers to evaluate the effectiveness of their adversarial defenses. We consider diversity an important property of defense techniques, and view the property in three respects: training, model, and decision diversity. Training diversity means multiple datasets are used to train the model(s), model diversity means multiple models are used for defense, and decision diversity means multiple decisions are made and then they are weighted for the final decision. Although this work is not the sole one that covers the three areas of diversity, it involves higher diversity inside the areas than related work (e.g., generating the adversarial datasets from more attack techniques for adversarial training). Because adversarial learning is a trending topic in machine learning, there are numerous attack techniques, learning models and defense methods in the literature. Thus, it is difficult to cover the combination of them all all exhaustively in a single study, and the issue of increasing diversity by covering more schemes certainly deserves further study in the future. However, we believe that this work has covered many common and well-known combinations in the evaluation. Moreover, although we agree that diversity is an important property, we do not imply it is the only novelty. Our work also covers study on model selection in a systematic way, the transferability property, and model prediction. These are the key differences of this work from the others.

Ibitoye et al. [17] used model configuration as a defense strategy, which works as finding the best parameter that a model can apply to render deep learning robust against adversarial attacks. To assess their model's configuration, they used three adversarial attack techniques, FGSM, BIM and PGD. However, their results are not entirely promising because their deep learning cannot counter against the adversarial attacks. The key aspect of that paper is that by feature normalization, the performance of models against adversarial attacks could be improved.

Five papers applied adversarial training as a defense technique. Each of them explored different types of adversarial attacks and classifiers. Pawlicki et al. [18] used adversarial training as a defense to evaluate four attack techniques on two ML techniques, namely C&W, FGSM, BIM, and PGD on Random Forest and Nearest Neighbor. They found that their approach was good at defending adversarial attacks, but was not very effective against the transferability property. Adversarial training has been also explored with another attack such as Deepfool, which was explained in [19]. In the exploration, ANN and Random Forest were used as the classifiers to evaluate the performance of each adversarial attack on the adversarial training. The result of that approach was promising, but that work considered only training diversity, but not model diversity because there were no combinations of models to defend the attacks. Adversarial training was also tested on JSMA to assess the capability to defend the attack, which was explored further in [20], in which adversarial training can help to defend a model from JSMA. Wang et al. [21] generated mimicked adversarial samples for multi-class intrusions from a single Generative Adversarial Network (GAN) model to augment the dataset, which was then used by [21] for adversarial training. A similar approach was explored in [22] with an improvement of a built-in Event-Condition-Action (ECA) model, where an event refers to a specific anomaly, a condition describes the rules, and the action is a countermeasure taken against an anomalous event.

Ensemble learning is also a very promising defense technique for making the model more robust against adversarial attacks. Apruzzese et al. [23] used Random Forest as the adversarial defense technique, which is categorized as ensemble classifiers. They tried to increase diversity of their Random Forest algorithm, but did not extend the diversity in training and decision-making, meaning that there were not adversarial training or voting steps. In that way, they showed that the model becomes more robust. Voting on the final decision is a technique to get the final result from an ensemble model, which was explained further in [24], where an ensemble model with voting, including SVM, DT and DNN, was used as the defense. The authors made their final decision based on majority voting, resulting in a diverse set of decisions and a model that could be utilized to develop the defensive technique. However, they did not apply any diversity measurements, which may lead to not obtaining the optimal ensemble model that they could have. Apruzzes et al. [25] designed an ensemble model with a layer of application-specific detectors, followed by another layer of botnet-specific classifiers, whose results were combined to make the final decision. The main purpose of that work was to restrict the range of variations that the adversarial attacks can generate (e.g., by adding junk data). Biggio et al. [26] presented a three-classifier ensemble model, including a two-class classifier and two one-class classifiers. Such a combination can reduce the chances of adversarial attacks without significantly sacrificing accuracy in the absence of attacks. The attack techniques and classifiers in the above two studies were not as diverse as those in this work. They also neither apply any diversity measurements nor study the transferability property.

Another approach is combining adversarial training and ensemble learning into the defense. We believe that the diversity of models can be increased by such a combination. Zhang et al. [9] adopted this approach, and saw a significant improvement in the robustness of the model against adversarial attacks. They augmented the training dataset with adversarial samples and retrained the network intrusion detection models, thereby reinforcing their capabilities against adversarial samples. However, the authors did not explain how they chose the right models to assemble or explain much on the transferability property that an adversarial attack has. Our work presents a systematic approach to select an effective model combination against a given set of adversarial attacks. To ensure effectiveness of the model combination, we use kappa statistics and double fault as the measurement score. This approach is important because we can eliminate those combinations that do not fit well with our objectives and thus obtain an efficient model. To ensure that our defense technique is strong, we evaluate it in depth using the transferability property of adversarial attacks.

An ensemble model consists of multiple models where each combination is evaluated using double fault and kappa statistics as the measurement scores. Double-fault compares the results from two model's predictions and calculates the score as

$$DF = \frac{FF}{TT + FT + TF + FF}. \tag{1}$$

Kappa statistics determines the agreement between each model, and the agreement score between each model is calculated as

$$KS = \frac{TT + FF}{TT + FT + TF + FF}. \tag{2}$$

TT means that both models' predictions are correct when tested with the testing dataset; FT and TF mean that one model's prediction is incorrect, and FF means that both models' predictions are wrong. Lower double-fault scores indicate more a successful combination because each model does not share the same false prediction. For the kappa statistics, a median score will be chosen to avoid having a very diverse combination.

3. Adversarial attack and defense problem formulation

This section covers our discussion of the problems to be addressed in two subsections: the notation table and the description of the issues.

3.1. Notations

Table 2 shows the notations used in this work, which are classified into three categories. The first is a dataset that includes two types of dataset, a clean feature dataset and adversarial attacked dataset. The second, machine learning category, includes machine learning algorithms, machine learning models, and ensemble models. This category shows how we denote models in this paper. The last category is adversarial attack, which includes many adversarial attack techniques. Details of each category are listed as follows:

Dataset: A NIDS with a high F1 score is capable of distinguishing between malicious and benign network traffic. To build such a NIDS, a well prepared dataset is crucial. Datasets consist of data inputs x_i and their corresponding labels y_i, which are then split into 3 smaller ones, training, validation and testing datasets. An adversarial attacked dataset D^+ is derived from a clean dataset that has been attacked by adversarial attack functions. To perform adversarial training, an expanded dataset D^E with adversarial samples is required. To obtain such an expanded dataset, clean dataset D and adversarial attacked dataset D^+ need to be combined, thus

$$D^E = D \cup D^+. \tag{3}$$

Machine Learning: A model can be developed by training the classifier with the prepared dataset. An F1 score is used to measure how good the performance of a model is. Let M^* be the ML model with the highest F1 score and M^{+*} be the adversarial trained ML model with the highest F1 score. Ensemble models are classified into those for clean models E_k, and those for adversarial trained models E_k^+.

Experimental results R are collected from the model prediction system. The results are stored in a database and are compared to the vector of results collected when attacking the model using the adversarial attack functions. Cosine similarity is used to compare the two vectors of test results: a vector from the database and that from the testing phase.

Adversarial Attack: The last section of Table 2 is adversarial attack. There are six adversarial attack techniques f_m in this work to ensure diversity of attack data. The attack data will then be used to evaluate the robustness of clean and adversarial trained models. A set of adversarial attack techniques, F, will be used to generate the adversarial attacked dataset and evaluate the robustness of the models.

3.2. Description of the issues

There are three main issues in this work, (1) the most effective ML-based model for a NIDS, (2) adversarial attacked dataset for a NIDS, and (3) model prediction in a NIDS. These issues are discussed below.

3.2.1. Issue 1. Most effective ML-based model for a NIDS

The first issue is divided into three sub-issues, *single and ensemble*, *adversarial trained ensemble*, and *most effective approach*. The sub-issues are connected in a chain, and therefore the first needs to be addressed first to fulfill the next sub-issues.

1.1. Single and ensemble. We assemble multiple baseline models by training multiple models using the dataset we already have. We then create the ensemble model by combining the models we get from the first step. The objective here is to obtain a model for both baseline configuration and ensemble with the highest F1 score.

Given the training dataset D^R, machine learning algorithm ML_j and testing dataset D^T, we expect the outputs to include a single M^* and an ensemble model E^* that have the highest F1 score when tested using the testing dataset D^T.

1.2. Adversarial trained ensemble. Similar to the previous sub-issue, we want to obtain an adversarial trained model and an ensemble adversarial trained model by including the expanded data which consist of adversarial samples as the training data generated in the next issue. An ensemble adversarial trained model will be obtained by combining several adversarial trained models. The objective with this sub-issue is to measure the performance of adversarial training and the ensemble adversarial trained approach.

The inputs of this issue are the expanded training dataset D^{ER}, the adversarial attacked testing dataset D^{T+}, machine learning algorithm ML_j and single model M_j. We aim to determine the adversarial trained model M^{+*} and the ensemble adversarial trained model E^{+*} that maximize the difference of the sums of the F1 scores of the model without and with adversarial training when tested using adversarial attacked testing dataset D^{T+}.

1.3. Most effective approach. Here we want to know which approach is the most effective among the four models we created with the previous sub-issues. By comparing their F1 scores, we will see which is the most effective.

In this case, adversarial attacked testing dataset D^{T+} and the four models, M^*, E^*, M^{+*}, and E^{+*}, are the inputs for determining the most effective approach A^* that minimizes the degradation of F1 scores when tested using D^{T+}.

Table 2
Notation.

Dataset		
Dataset	D	$D = \{(x_i, y_i), i = 1, 2, \ldots, n\}$; $D = D^R \cup D^T$; $R \cup T = \{1, 2, \ldots, n\}$
Dataset for testing	D^T	$D^T = \{(x_i, y_i), i \in \mathrm{T}\}$
Dataset for training	D^R	$D^R = \{(x_i, y_i), i \in \mathrm{R}\}$
Adversarial attacked dataset	D^+	$D^+ = \{(x_i^+, y_i^+), i = 1, 2, \ldots, n\}$
Adversarial attacked data	x_i^+	$x_i^+ \in R^n$, where n = number of features; $x_i^+ = f_m(x_i)$
Adversarial dataset for testing	D^{T+}	$D^{T+} = \{(x_i^+, y_i^+), i \in \mathrm{T}\}$
Expanded dataset with adversarial samples	D^E	$D^E = D \cup D^+$
Expanded dataset for training	D^{ER}	$D^{ER} = \{(x_i, y_i), i \in \mathrm{R}\} \cup \{(x_i^+, y_i^+), i \in \mathrm{R}\}$
Data input	x_i	$x_i \in \mathbb{R}^n$, where n = number of features
Label	y_i	$y_i \in \{0,1,2\}$, where 0, 1, 2 are normal data, network attack data, adversarial attack data respectively
Machine learning		
Number of ML algorithms	N_{ML}	
ML algorithm	ML_j	$0 \leq j \leq N_{ML}$-1
ML model	M_j	$M_j = ML_j(D^R)$
ML model with the highest F1 score	M^*	
ML model with adversarial training	M_j^+	$M_j^+ = M_j(D^{ER})$
Adversarial trained ML model with the highest F1 score	M^{+*}	
Ensemble model	E_k	$E_k \subset M$, where $\mathrm{M} = \bigsqcup_j M_j$, $k = 0, 1, 2, \ldots, 2^{\mathrm{N_{ML}}}$ - 1
Ensemble model with the highest F1 score	E^*	
Ensemble model with adversarial training	E_k^+	$E_k^+ \subset M^+$, where $\mathrm{M}^+ = \bigsqcup_j M_j^+$, $k = 0, 1, 2, \ldots, 2^{\mathrm{N_{ML}}}$ - 1
Adversarial trained ensemble model with the highest F1 score	E^{+*}	
Experimental results	R	List of experimental results [F1, AUC] from $M_j(D^+) \cup M_j^+(D^+)$
Best approach	A^*	Approach with the lowest F1 score difference
List of models behind the system	M^S	Closest performance result of system's models compared to the experimental models
Adversarial attack		
Number of attack techniques	N_F	
Adversarial attack technique	f_m	$0 \leq \mathrm{m} \leq N_F$-1
Set of adversarial attack techniques	F	$F = \{f_1, f_2, \ldots, f_{N_F}\}$

3.2.2. Issue 2. Adversarial dataset for a NIDS

The dataset in this sub-issue will be perturbed by applying multiple adversarial attack functions into the dataset. We intend to see how the performance of each model degrades by perturbing the dataset using an adversarial attack function. The generated dataset can then be used to test the models and become the input for our adversarial training approach.

The inputs to this issue are the training dataset D^R, adversarial attack functions F and single models M_j, where we want to determine which attacked datasets D^+ can minimize the F1 score of the model when tested using adversarial attacked datasets D^{T+}.

3.2.3. Issue 3. Model prediction in a NIDS

In this last issue, we will use all the scores from the experimental results to predict the model. The objective is to be able to have a system with a high F1 score when predicting the model adopted behind the system.

Given adversarial attacked testing dataset D^{T+}, experimental results R and the system model M^S as the inputs, we want to decide which prediction algorithm to maximize the cosine similarity score when predicting with the model M^S.

4. Design approaches

4.1. Solution overview

In this work, we propose ELAT, which is a defense technique that makes an ML-based NIDS resistant to adversarial attack techniques in order to address the issues outlined in Section 3.2.

Several single learning models are required to assess the most effective combination to be used in the development of an ensemble model. Thus, we first create these models. The clean models are trained with network traffic dataset which consists of normal and attack traffic, and an adversarial trained model is trained with adversarial attacks. To create clean models, we design a solution called *Single and Ensemble Model Creation based on F1 score and Measurement Score*. The ensemble technique we use is stacking ensemble, which uses a meta-learning algorithm to learn how to best combine the predictions from two or more base machine learning algorithms. An ensemble technique can harness the capabilities of a range of well-performing models on a classification or regression task and make predictions that have better performance than any single model. For the final decision, we use soft majority pooling which combines the probability of each prediction [27].

In order to obtain the adversarial attacked dataset, we need an adversarial attack function to craft the dataset which will address the *Exhaustive Generation* solution design (see Section 4.2.2). By generating an adversarial attacked dataset, we are then able to create an adversarial trained model which forms the basis of *Adversarial Trained and Ensemble Adversarial Trained Model Creation based on F1 Score and Measurement Score*. In this solution, we also design the process of creating an ensemble adversarial model by combining adversarial trained models.

After getting all the models ready, we need to evaluate how each of the them performs as an adversarial defense system in an ML-based NIDS. To do so, we design a solution to *Model Selection*, in which we use all the scores obtained from the experiment steps to help us in predicting the model behind the tested system. To be able to predict this model, we use a cosine similarity approach which we term *Cosine Similarity for Predicting Model*, and which is explained in Section 4.2.3.

Fig. 1 illustrates the relationship between the issues on the left and solutions on the right side in the design.

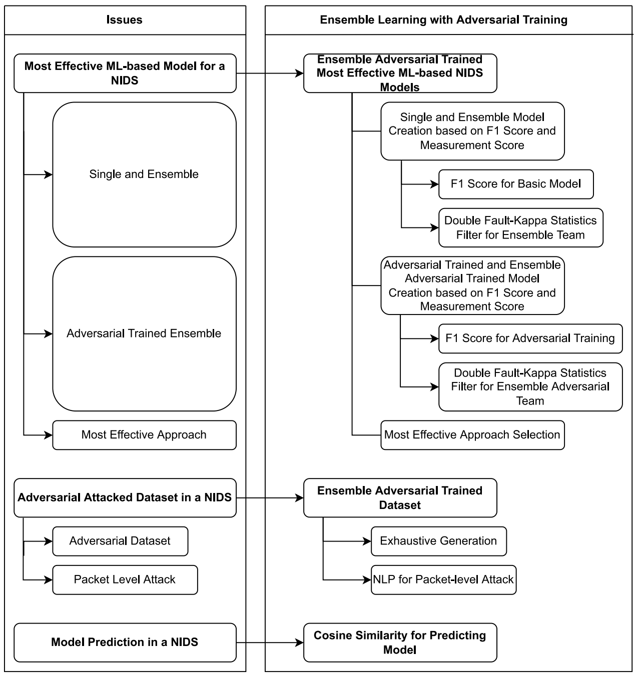
Issues
Ensemble Learning with Adversarial Training
Most Effective ML-based Model for a NIDS
Ensemble Adversarial Trained Most Effective ML-based NIDS Models
Single and Ensemble
Single and Ensemble Model Creation based on F1 Score and Measurement Score
F1 Score for Basic Model
Double Fault-Kappa Statistics Filter for Ensemble Team
Adversarial Trained Ensemble
Adversarial Trained and Ensemble Adversarial Trained Model Creation based on F1 Score and Measurement Score
F1 Score for Adversarial Training
Double Fault-Kappa Statistics Filter for Ensemble Adversarial Team
Most Effective Approach
Most Effective Approach Selection
Adversarial Attacked Dataset in a NIDS
Ensemble Adversarial Trained Dataset
Adversarial Dataset
Exhaustive Generation
Packet Level Attack
NLP for Packet-level Attack
Model Prediction in a NIDS
Cosine Similarity for Predicting Model

Fig. 1. Issues (left) and solutions (right) mapping.

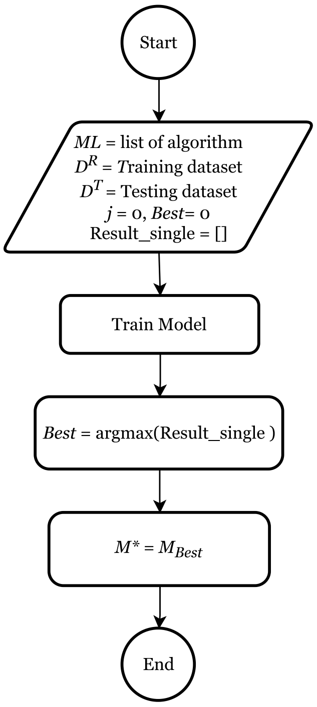

Fig. 2. F1 score for a single model.

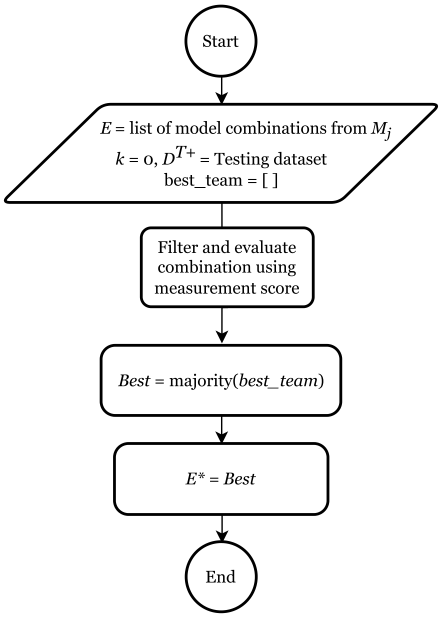
Start
E = list of model combinations from M_j
k = 0, D^{T+} = Testing dataset
best_team = [ ]
Filter and evaluate combination using measurement score
$Best$ = majority($best\_team$)
$E^* = Best$
End

Fig. 3. Double fault and kappa statistics filter for ensemble model.

4.2. Solution details

4.2.1. Solution 1. ML-based NIDS models with ELAT

For an ML-based NIDS, we consider three possible solutions to obtain the most effective defense approach for achieving a model robust against an adversarial attack and its transferability property.

Solution 1.1. Single and ensemble model creation based on F1 score and measurement score. There are two possible solutions here: *F1 Score for Single Model* and *Double Fault-Kappa Statistics Filter for ensemble model*.

F1 Score for single model. Fig. 2 shows the flowchart of the F1 score for single model solution. The iteration of this solution depends on how many ML algorithms are evaluated. Each of the ML algorithms will be trained with a training dataset and evaluated with a testing dataset. The score from the testing steps will allow us to ascertain which algorithm performs the best. We then use the argmax function, which indexes the highest F1 score within a list of possible ensemble models.

Double Fault-Kappa Statistics Filter for Ensemble model (Fig. 3). The ensemble model creation is be determined using double fault and kappa statistics, which will help us to obtain the most effective model combination automatically, so that we do not have to test each combination.

We create the model from the list of possible model combinations gathered from the previous step. Each of the models then has to be evaluated to obtain the most effective model to defend from an adversarial attack. The iteration of this solution depends on the number of combinations. We evaluate the model for each iteration using two measurement scores, double fault and kappa statistics.

From Fig. 3, one can see that each model combination is evaluated one by one, and we set a rule that a combination is accepted only if the double fault score is below 50% and the kappa statistics score is between 40% and 80%. The double fault is set below 50% because the models that have a double fault score above 50% will generate too many false positives and false negatives, which are undesired in a robust NIDS. A score between 40% and 80% for the kappa statistics is used because we want to obtain medium to high diversity for the ensemble model. As stated in Section 2, kappa statistics measures how each of the models agree with each other on the prediction results. Anything below 40% will make the combination too diverse, and therefore there might still be false negatives or false positives, whereas above 80% means that the models are not diverse and hence assembling them into a model might not be needed.

The models that follow the rule will then be listed, and finally we can see how frequent the models appear within the list of conformed models. If an ensemble model appears many times, it means that the model follows the rule in different test conditions. The model that appears the most times will then be considered the most effective one in the ensemble approach. This algorithm helps us to find the

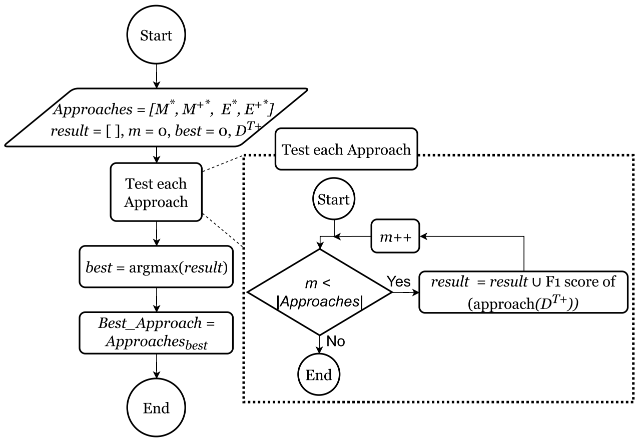
Start
Approaches = [M*, M+*, E*, E+*]
result = [ ], m = 0, best = 0, D^T+
Test each Approach
best = argmax(result)
Best_Approach = Approaches_best
End
Test each Approach
Start
m++
m < |Approaches|
Yes
result = result ∪ F1 score of (approach(D^T+))
No
End

Fig. 4. Model selection.

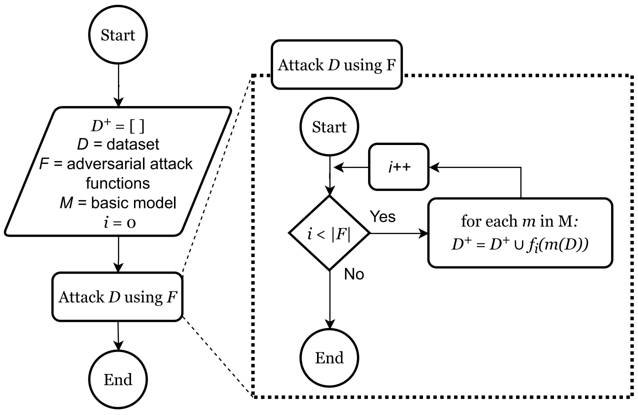

Fig. 5. Exhaustive generation.

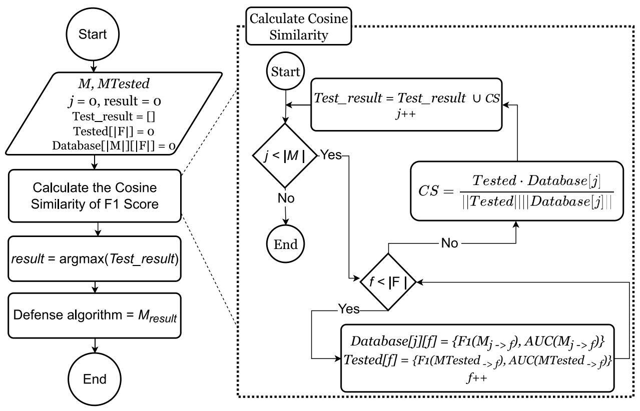
Start
M, MTested
j = 0, result = 0
Test_result = []
Tested[|F|] = 0
Database[|M|][|F|] = 0
Calculate the Cosine Similarity of F1 Score
result = argmax(Test_result)
Defense algorithm = M_result
End
Calculate Cosine Similarity
Start
Test_result = Test_result ∪ CS
j++
j < |M|
Yes
No
End
CS = Tested · Database[j] / ||Tested|| ||Database[j]||
No
f < |F|
Yes
Database[j][f] = {F1(M_j->f), AUC(M_j->f)}
Tested[f] = {F1(MTested->f), AUC(MTested->f)}
f++

Fig. 6. Cosine similarity for predicting model.

most effective combination in an effective way; otherwise, a manual comparison needs to be made for every possible combination.

Solution 1.2. Adversarial trained and ensemble adversarial trained model creation based on F1 score and measurement score. This solution design is similar to Solution 1.1, but its difference is in the input. The input in this case is no longer a clean dataset, but an adversarial attacked dataset produced in Solution 2, *Exhaustive Generation*. This solution is divided into two parts, one for generating the adversarial training models, and the other is for creating the ensemble model for the adversarial trained models.

F1 Score for adversarial training. To generate an adversarial training model, we design a solution similar to that in Fig. 2. The difference is in the input data, where this solution uses the expanded dataset to train the model. For evaluation, we use multiple testing datasets which contain adversarial samples from multiple attack techniques generated by applying adversarial attacks to the datasets. The F1 scores will be collected into a list, and we take the argmax of that list to identify the most effective model. The objective of this solution is to have the most effective adversarial trained model which is robust against adversarial attacks and its transferability property.

Double fault and kappa statistics filter for ensemble adversarial model. After all the adversarial trained models are collected, we can combine them into a model. For this solution, we create an ensemble model with at least two models. The flow of the solution is exactly the same from Fig. 3. The difference is that the ensemble model no longer creates a model from clean single models but from adversarial trained single models.

Solution 1.3. Model selection. To address the issue of selecting the most effective model for an ML-based NIDS to defend against adversarial attacks, we compare the performance of all the models in consideration. To evaluate these models, we use all the adversarial attacked testing datasets and test the approaches towards this dataset. Fig. 4 shows the four models, M^*, M^{+*}, E^* *and* E^{+*}, which are collected as described in Section 3.2. Each of the models will be tested using the adversarial attacked testing dataset, and the score will be stored in a list. We then take the argmax of this list for the index of the most effective model. Finally, we can find the desired model that will be used as the defense in an ML-based NIDS against adversarial attacks. All testing results within the first solution will be stored in a database, and then will be used as the input for the third solution.

4.2.2. Solution 2. Exhaustive generation

This solution aims to produce a dataset of adversarial attacks based on multiple adversarial attack techniques. Fig. 5 shows the flowchart of this proposed solution, which has main inputs, a dataset, adversarial attack functions, and all the models generated by the previous solution. We term this solution the *exhaustive generation* because generating the dataset has been compiled exhaustively depending on how many adversarial attack functions and models we have.

4.2.3. Solution 3. Cosine similarity for predicting model

The design of this model is shown in Fig. 6. This model needs multiple inputs such as all the adversarial attacked datasets generated in the previous model, a database which contains all the scores derived from the various test steps, and the targeted system itself. First, we insert the adversarial attacked dataset into the targeted system to generate a vector of scores. This vector then will be compared to the vectors that are already in our database by using cosine similarity. We choose cosine similarity because Euclidean distance and cosine similarity are two major approaches to compare two vectors, but the latter is more suitable for comparing two high-dimensional vectors. The dimension is the number of attack techniques for different classifiers, which is 17 in our evaluation (see Fig. 8). The dimension will be even higher if more attack techniques are applied in any future evaluation. We use two metrics to compare the two vectors of scores, the F1 score and AUC (Area Under Curve). With these two metrics, detection will be more precise. Finally, we use argmax to see which vector in our database is most similar to the score we obtained from testing the targeted system. A higher cosine similarity score means the model is more likely the one being compared in our database.

5. Experimental results

This section begins with the description of the hyper-parameter configuration for all classifiers that we use. We then give the experimental results of the issues covered in Section 1.

Table 3
Baseline configuration.

Classifiers	Hyper-parameters	Values
DT	Criterion	Entropy
	Splitter	Best
	Max_depth	9
	Min_sample_leaf	1
	Min_sample_split	2
LR	C	202010.513969
XGB	Booster	Gbtree
	Lambda	7.10074
	Alpha	0.00102
	Max_depth	9
	Eta	0.67375
	Gamma	6.10126
	Grow_policy	Lossguide
SVM	C	2.70677
DNN	Input Layer	1 Layer
	Hidden Layer	3 Layers
	Output Layer	1 Layer
	Hidden Activation	ReLU
	Output Activation	Softmax
	Dropout	0.01

5.1. Baseline configuration

We chose CICIDS-2017 as our dataset because it has the most up-to-date common attacks and is therefore relevant to today's network environment. The advantage of using CICIDS-2017 is that it fulfills the ten necessary requirements of a reliable benchmark dataset: complete network configuration, complete traffic, labeled dataset, complete interaction, complete capture, available protocols, attack diversity. heterogeneity, feature set and metadata [28]. Unlike most other works which rely on public datasets, we reproduced attacks and generated datasets for our evaluation to draw the conclusion. The public dataset is only used for comparison purpose. Since the compared target also used the same dataset, we chose to report the comparison results with the same dataset. Nevertheless, we are aware that CICIDS-2017 was pointed out to be flawed in some aspects [29], but we believe that no dataset is perfect. Since CICIDS-2017 is still one of the most commonly used datasets in the literature, and was still used in some recent seminal papers such as [30] for easier comparison with many papers using this dataset, it is still a proper choice for this work. Moreover, several papers [31–33] reported that the F1 scores or detection rates of ML-based intrusion detection were rather similar, if a latter version of CICIDS-2017, CSE-CIC-IDS2018, was used. Although there are not any comparisons of the detection performance between CICIDS-2017 and its improved version by [29] in the literature, to the best of our knowledge, we believe that if we had used that improved version, the numerical results of detection performance in this work might have been slightly different, but the main observations and conclusions would have been similar. Furthermore, although we used only one dataset in the experiments, we have produced exhaustively the dataset with a rich combination of attack techniques and detection models from this dataset (see Section 4.2.2), and have carried out deep study on the produced dataset with the rich combination. Therefore, the experiments still cover the diversity that can reach meaningful conclusions.

Several hyper-parameter configurations have been utilized to optimize the performance of each of the classifiers. Table 3 lists the hyper-parameter settings for this work. These were obtained using Optuna [34], which is a tool to optimize the hyperparameters of a classifier. Hyper-parameter tuning was carried out to ensure the optimal baseline model for the entire work. Our experiment shows that such tuning can boost the performance of a model by 4% for Logistic Regression.

To implement adversarial attacks, an open-source toolkit called *adversarial robustness toolbox* was used. This toolkit consists of many adversarial attack techniques as well as defense, especially in the image domain. Since most of the attacks are implemented in the image domain, a minor adjustment for tackling network flows is needed to use this toolbox in the network domain [35]. The perturbations in the adversarial attacks were applied to the feature space according to our threat model (see Section 1).

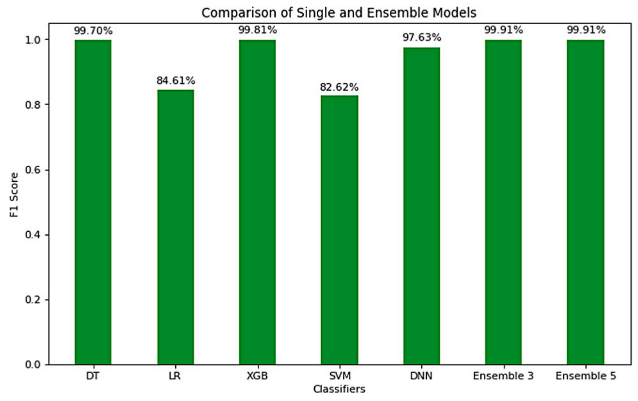

Fig. 7. Comparison of single and ensemble models.

5.2. Solution results

5.2.1. Single vs. Ensemble

This section compares the performance of the single and ensemble models with the testing dataset that we have, the original testing dataset and the adversarial set.

Original testing dataset. Fig. 7 shows the performance of the single models and ensemble models against network attacks without any adversarial attacks injected into the dataset. For the original testing dataset, the single models perform well with an average F1 score of 0.93 for the different models. XGB was the best performer with an F1 score of almost 1 (99.81%[1]) compared to the other single models, and SVM was the lowest with an F1 score of 0.83 (82.62%). From a single model perspective, DT and XGB performed the best out of the five classifiers because the internal nodes in the tree structures of DT and XGB tested an attribute with 'if else' statements made to fit the data to a certain threshold, which was done in the training step. Compared to the others, however, SVM did not perform as well because it separates the data points above and below the classified hyper-plane, and there is thus no probabilistic explanation for the classification. When models are combined into an ensemble model and the decision was taken by pooling, we saw an increase in terms of the F1 scores. Fig. 7 also shows an increase of F1 score by 0.001 (0.1%) from the highest F1 score of the single model because the decision is not only made by one model, but by multiple models that provide their predictions to the testing dataset.

There are two types of ensemble models in this work. The first consists of three models, and the second consists of five models. An odd number of models was selected because we wanted to obtain a fair comparison where the case of 50:50 is less likely to occur with an odd number.

After the experiment, we found that the best combination for a three-model ensemble model is the combination of DT, XGB, and SVM (i.e., 'Ensemble 3' in Fig. 7). This combination was the best because they had a good kappa and double-fault score and therefore performed well when combined. On the other hand, the five-model combination gave similar performance with the former model. From this experiment, the ensemble approach is better than the single approach when tested on the original testing dataset.

[1] To reserve the higher precision for comparisons, we present the F1 scores in percentage and keep two decimal places in Figs. 7 and 10.

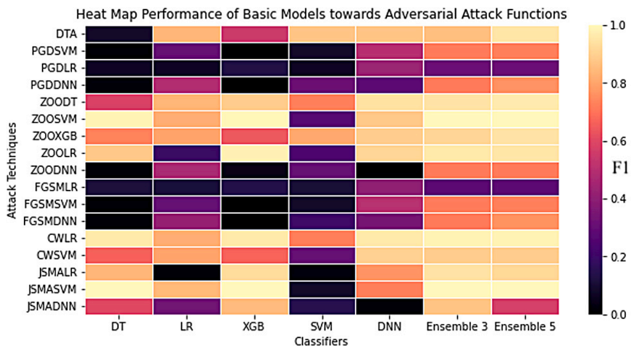

Fig. 8. Heat map performance of single and ensemble models against adversarial attack functions. The item names in the *y*-axis are concatenations of attack names and the classifier. The 'Ensemble 3' model incorporates DT, XGB, and SVM, while the 'Ensemble 5' incorporates all the five classifiers in this work.

Adversarial testing dataset. There are a couple of things that we can learn from the adversarial testing dataset, such as the strength of adversarial attack techniques, the strength of classifiers compared to adversarial attack techniques, the strength of transferability property, and how the ensemble model performs against an adversarial attack. The details are described below.

The Strength of adversarial attack technique

We would first want to know which adversarial attack technique is the strongest: the strength of each adversarial attack can be seen from Fig. 8, cell by cell. The *x*-axis has five classifiers and two ensemble models, and the *y*-axis has all kinds of adversarial attack functions that we want to analyze. The colors represent the performance of each model. The lighter the color, the higher the F1 score is. The figure shows that most of the single models are perturbed by adversarial attacks: most cells are in a dark color, which indicates that it is possible to attack an ML-based NIDS using adversarial attacks. From the results, we can see that the performance of the single models drops because of the darker color. The average score is dropped from 0.93 to 0.07 and from 0.93 to 0.61 using a PGD and a ZOO attack, respectively. A PGD attack is the strongest because it relies on a multi-step attack based on FGSM and therefore multiple perturbations are carried out in PGD. A ZOO attack has the lowest average success rate, which is defined as 1-(F1 score), since the gradient information used in the attack is not taken from an optimization problem, and so the effect is low-level perturbation. Therefore, the strength of an adversarial attack depends on whether the attack mechanism uses single-step or multi-step. Moreover, the strength of an adversarial attack also relies on the gradient of the model, and larger gradient means stronger impact of the adversarial attack.

The Strength of classifiers vs. adversarial attack technique

Fig. 8 also shows which classifier is more robust against adversarial attack functions. LR with an F1 score of 0.52 is the most robust classifier against adversarial attacks because it is a simple classifier with a small gradient, and any perturbation against a small gradient classifier will result in a low-level perturbation. SVM, with an F1 score of 0.31, is not robust against an adversarial attack because it depends only on a support hyper-plane that is very sensitive to any changes to the dataset. The robustness of a classifier depends on its gradient, where a high gradient makes the classifier more vulnerable to an adversarial attack because it will add more noises to the input feature. On the other hand, a low gradient means adding only some noises to the input feature, which results in more robust classifiers. A high gradient is closely related to complex classifiers, whereas a low gradient is related to simple classifiers.

The Strength of transferability property

Fig. 8 also presents the strength of the transferability property on a classifier. This property of an adversarial attack can also be observed in an ML-based NIDS by looking at the figure horizontally, to see whether an attack with the adversarial examples from one model can successfully fool another. Thus, the transferability property can be measured from the F1 scores or success rate (defined as 1- (F1 score)) in such attack cases. If the F1 score of an attack is degraded significantly or the success rate is high, we consider the attack has the transferability property. A specific adversarial attack is possible to attack multiple classifiers because data structures and number of features are similar across models, i.e, this attack is transferable. We can see that most F1 scores decrease when various attack techniques are used against the models. The highest average attack success rate for the transferability property is achieved using DNN, with a success rate of 0.84. As a complex classifier will have a higher gradient, a high gradient with noise will create a stronger perturbation. On the other hand, CWLR has the lowest average attack success rate at 0.38. As noted above, since LR is a simple classifier, the gradient will be small and the perturbation on a low gradient classifier will be also small. From these results, we can learn that, when crafting adversarial samples using a complex classifier, we will achieve more transferability, but when crafting using a simple classifier, the transferability is less.

Ensemble model vs. adversarial attack

The last thing that we can see in Fig. 8 is how an ensemble model performs against adversarial attacks. The last two columns in the figure are the results of the ensemble models. Looking from a different perspective, we know that an ensemble model is robust against adversarial attacks, which is represented by an F1 score that increases from 0.41 to 0.81, compared to the average performance of single models. Robustness of the ensemble model is achieved by combining the decision of multiple models. The F1 scores between the three- and five-model combinations are slightly different. The point of this result is that an ensemble approach can enhance the robustness of a system against adversarial attacks. Selecting the right combination is necessary to create the best ensemble models. Based on kappa and double fault statistics, DT, SVM, and XGB are the best for a three-combination ensemble model, and have been demonstrated to be better than a five-model ensemble, with an average F1 score of 0.81 for ensemble 3 and 0.79 for ensemble 5.

5.2.2. Single vs. Adversarial

In this subsection, we want to determine whether adversarial training actually helps the system to be more robust against adversarial attacks or not. From Fig. 9, we can see the performance of models when trained using adversarial samples. Most of the scores in the heat map are very high. The scores are represented by their colors, and the average score is recovered to 0.95 if we compare Fig. 8 and Fig. 9. The average F1 score of adversarial training is higher than that of ensemble model by 0.08, from 0.80 to 0.88. This improvement is expected because an adversarial approach will enable those models to learn the adversarial techniques, and adversarial training is thus one of the most effective ways to make the system robust against adversarial attacks. An ensemble model performs worse because the models which are assembled into an ensemble model have not learned the patterns of adversarial attacks yet; therefore, combining those models will not make the decisions better than a model that has already learned the patterns of the adversarial samples.

There are however problems when an adversarial approach is implemented in a system. Fig. 10 shows that an adversarial training model will not be able to recover the F1 score when the transferability property is carried out by an adversarial attack. To demonstrate this, we first train all available models with PGDSVM, and then attack all the models using PGDLR. The blue bars show the performance in F1 scores of the single models tested using the original testing dataset, which shows that the performance of these models is good.

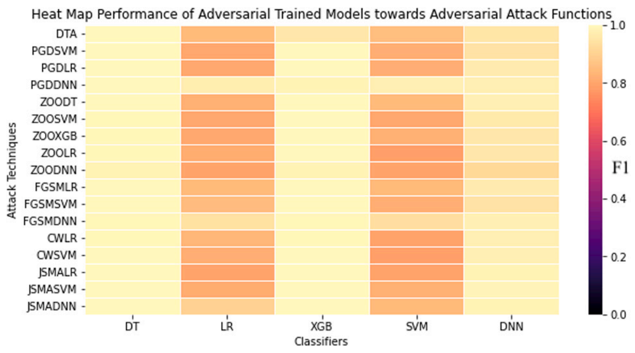

Fig. 9. Heat map of adversarial training model vs. adversarial attack.

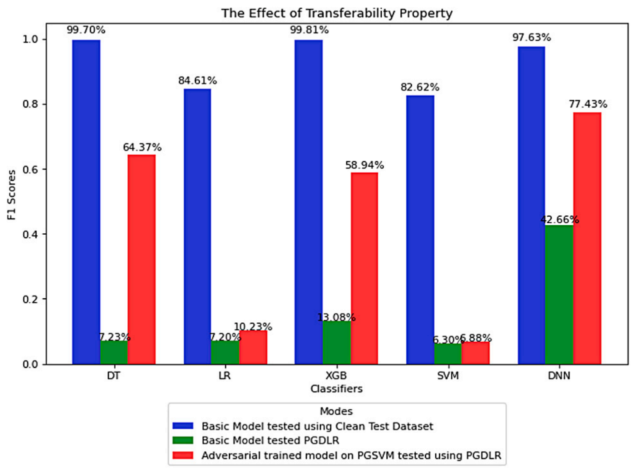

Fig. 10. The effect of transferability property.

When the single models were tested using PGDLR, more than 0.80 of their performance was degraded, as shown by the green bar. The red bar shows the adversarial trained model that has been trained using PGDSVM. It can be seen that even though the model had been trained with adversarial samples, the performance still did not recover to its original levels. Fig. 10 shows how the PGDSVM model is still vulnerable to a PGDLR attack because the models only learned a specific type of attack and failed to be generalized to other adversarial attacks. Thus, when the model was attacked by other attack techniques, it still could not recognize them. From this, we learn that adversarial training does not solve the problem of defending adversarial attack completely, since it still fails the transferability property tests.

5.2.3. *Adversarial vs. Ensemble adversarial*

In this section, we wanted to determine whether an adversarial or ensemble adversarial approach performed better. The difference between these two approaches is that one uses the ensemble approach to create a model and the other does not. The last column of Fig. 11 shows that the ensemble learning with adversarial training (ELAT) approach can increase its average F1 score from 0.69 to 0.91, and this result is as expected. This shows that by combining multiple models (using 'Ensemble 3' in this figure), we can increase the robustness of a model by 0.22 because by combining both ensemble and adversarial training approaches, a single model can cover more attack possibilities. When a certain adversarial attack tries to attack a model, the ensemble model already knows the behavior of the attack because we combined adversarial models. ELAT performs better than adversarial training even when the transferability property occurs with the adversarial attack techniques. We can thus see that an ensemble adversarial approach is the most preferable to be used as a defense against adversarial attacks for an ML-based NIDS.

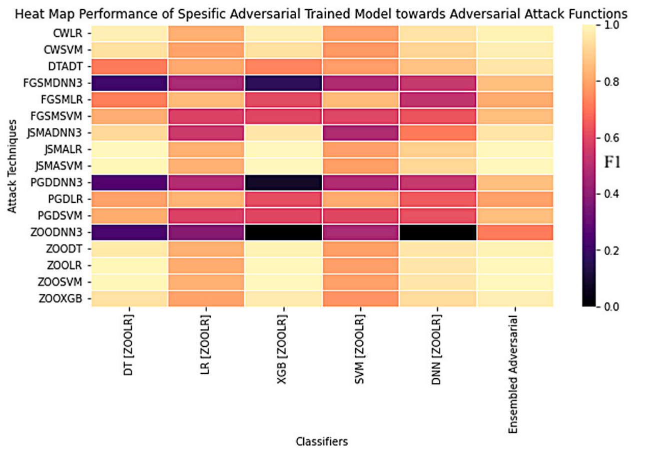

Fig. 11. Heat map of adversarial vs. ensemble adversarial approaches.

5.2.4. *Predicting model adopted by system under test*

This last section shows the result of predicting a model that is used behind an ML-based NIDS. The average accuracy of the method with cosine similarity described in Section 4.2.3 is very high, where 99.9% of the predictions are correct. This method is capable of predicting approach and models including the single, adversarial, trained and ensemble model behind the system by applying adversarial attacks into the target system and comparing the results with the one that uses the cosine similarity algorithm.

6. Conclusion and future work

The average attack success rate for all adversarial attack techniques is 0.71. PGD is the strongest adversarial attack technique, whose success rate is 0.92. PGD is a multi-step algorithm, meaning that perturbations are carried out many times and in addition this attack technique uses projection that makes the effect of the attack stronger.

All the classifiers are affected by adversarial attacks with an average of 0.45 in terms of their performance, which means that adversarial attacks become a real threat towards an ML-based NIDS. The most robust classifier for ML-based NIDSs against adversarial attacks is LR, with an average F1 score of 0.52. Because of its simplicity, LR will have a smaller gradient and is therefore not sensitive to any small changes to the dataset. The simpler the classifier is, the more robust it is against adversarial attacks.

There is on average a possibility of 0.61 (i.e., the average attack success rate for DNN and CWLR) that an adversarial attack is transferable. The strongest transferability property results from crafting the adversarial samples using DNN. Our results show that 84% of adversarial attack techniques are transferable using DNN because DNN is a complex model which makes the perturbation even higher and stronger, resulting in the perturbation being sent to other classifiers, where the effect of the perturbation can be strongly felt.

ELAT is the most effective approach to defend an ML-based NIDS against adversarial attacks and their transferability property, where it combines two approaches, adversarial training and ensemble model. This approach learns from the 'bad guys' and combines multiple models, making this approach robust against adversarial attacks. ELAT can also improve the F1 score from 0.07 to 0.91 when attacked by adversarial attacks.

Adversarial learning is a trending topic in machine learning. It can be seen that numerous attack techniques, learning models and defense methods have been proposed and studied in the literature. Thus, it is difficult to cover the combination of them all exhaustively in a single study. The issue of increasing diversity by covering more schemes certainly deserves further study in the future.

CRediT authorship contribution statement

Ying-Dar Lin: Problem definition, Paper revision. **Jehoshua-Hanky Pratama:** Algorithm design, Software, Experimental study, Writing. **Didik Sudyana:** Software, Experimental study. **Yuan-Cheng Lai:** Algorithm design, Supervision. **Ren-Hung Hwang:** Conceptualization, Supervision. **Po-Ching Lin:** Writing – review & editing, Paper revision. **Hsuan-Yu Lin:** Problem definition. **Wei-Bin Lee:** Paper revision. **Chen-Kuo Chiang:** Paper revision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential.

Acknowledgment

This work was supported in part by Telecom Technology Center.

References

[1] Reddy GN, Reddy GJU. A study of cyber security challenges and its emerging trends on latest technologies. 2014, ArXiv, arXiv:1402.1842.
[2] Ahmad Z, Shahid Khan A, Wai Shiang C, Abdullah J, Ahmad F. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. Trans Emerg Telecommun Technol 2021;32(1). http://dx.doi.org/10.1002/ett.4150.
[3] Goodfellow IJ, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. In: Bengio Y, LeCun Y, editors. 3rd international conference on learning representations. 2015, URL http://arxiv.org/abs/1412.6572.
[4] Li D, Li Q. Adversarial deep ensemble: Evasion attacks and defenses for malware detection. IEEE Trans Inf Forensics Secur 2020;15:3886–900. http://dx.doi.org/10.1109/tifs.2020.3003571.
[5] Wu L, Zhu Z. Towards understanding and improving the transferability of adversarial examples in deep neural networks. In: Pan SJ, Sugiyama M, editors. Proceedings of the 12th asian conference on machine learning. Proceedings of machine learning research, vol.129, PMLR; 2020, p. 837–50, URL https://proceedings.mlr.press/v129/wu20a.html.
[6] Bai T, Luo J, Zhao J, Wen B, Wang Q. Recent advances in adversarial training for adversarial robustness. In: International joint conference on artificial intelligence. 2021, arXiv:2102.01356.
[7] Ganaie MA, Hu M, Tanveer M, Suganthan PN. Ensemble deep learning: A review. 2021, URL https://arxiv.org/abs/2104.02395.
[8] Tramèr F, Kurakin A, Papernot N, Goodfellow I, Boneh D, McDaniel P. Ensemble adversarial training: Attacks and defenses. 2020.
[9] Zhang C, Costa-Pérez X, Patras P. Adversarial attacks against deep learning-based network intrusion detection systems and defense mechanisms. IEEE/ACM Trans Netw 2022;1–18. http://dx.doi.org/10.1109/TNET.2021.3137084.
[10] Papernot N, Mcdaniel P, Goodfellow IJ. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. 2016, ArXiv, arXiv:1605.07277.
[11] Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A. Towards deep learning models resistant to adversarial attacks. 2017, arXiv URL https://arxiv.org/abs/1706.06083.
[12] Carlini N, Wagner D. Towards evaluating the robustness of neural networks. In: IEEE symposium on security and privacy. Los Alamitos, CA, USA: IEEE Computer Society; 2017, p. 39–57, URL https://doi.ieeecomputersociety.org/10.1109/SP.2017.49.
[13] Chen P-Y, Zhang H, Sharma Y, Yi J, Hsieh C-J. ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In: Proceedings of the 10th ACM workshop on artificial intelligence and security. 2017, p. 15–26. http://dx.doi.org/10.1145/3128572.3140448.
[14] Papernot N, Mcdaniel P, Jha S, Fredrikson M, Celik ZB, Swami A. The limitations of deep learning in adversarial settings. In: IEEE european symposium on security and privacy. 2016, p. 372–87.
[15] Martins N, Cruz JM, Cruz T, Henriques Abreu P. Adversarial machine learning applied to intrusion and malware scenarios: A systematic review. IEEE Access 2020;8:35403–19.
[16] Sitikhu P, Pahi K, Thapa P, Shakya S. A comparison of semantic similarity methods for maximum human interpretability. 2019, arXiv URL http://arxiv.org/abs/1910.09129.
[17] Ibitoye O, Shafiq MO, Matrawy A. Analyzing adversarial attacks against deep learning for intrusion detection in IoT networks. In: IEEE global communications conference. 2019, p. 1–6.
[18] Pawlicki M, Choraś M, Kozik R. Defending network intrusion detection systems against adversarial evasion attacks. Future Gener Comput Syst 2020;110:148–54, URL https://www.sciencedirect.com/science/article/pii/S0167739X20303368.
[19] Khamis RA, Matrawy A. Evaluation of adversarial training on different types of neural networks in deep learning-based IDSs. 2020.
[20] Anthi E, Williams L, Rhode M, Burnap P, Wedgbury A. Adversarial attacks on machine learning cybersecurity defences in industrial control systems. 2020, URL https://arxiv.org/abs/2004.05005.
[21] Wang J, Pan J, AlQerm I, Liu Y. Def-IDS: An ensemble defense mechanism against adversarial attacks for deep learning-based network intrusion detection. In: International conference on computer communications and networks. 2021, p. 1–9.
[22] Novaes MP, Carvalho LF, Lloret J, Proença ML. Adversarial deep learning approach detection and defense against DDoS attacks in SDN environments. Future Gener Comput Syst 2021;125(C):156–67. http://dx.doi.org/10.1016/j.future.2021.06.047.
[23] Apruzzese G, Andreolini M, Colajanni M, Marchetti M. Hardening random forest cyber detectors against adversarial attacks. IEEE Trans Emerg Top Comput Intell 2020;4:427–39.
[24] Asadi M, Jamali MAJ, Parsa S, Majidnezhad V. Detecting botnet by using particle swarm optimization algorithm based on voting system. Future Gener Comput Syst 2020;107:95–111, URL https://www.sciencedirect.com/science/article/pii/S0167739X19323350.
[25] Apruzzese G, Andreolini M, Marchetti M, Colacino VG, Russo G. AppCon: Mitigating evasion attacks to ML cyber detectors. Symmetry 2020;12.
[26] Biggio B, Corona I, He Z-M, Chan PPK, Giacinto G, Yeung DS, et al. One-and-a-half-class multiple classifier systems for secure learning against evasion attacks at test time. In: International workshop on multiple classifier systems. 2015.
[27] Wang H, Yang Y, Wang H, Chen D. Soft-voting clustering ensemble. In: Zhou Z-H, Roli F, Kittler J, editors. Multiple classifier systems. Berlin, Heidelberg: Springer Berlin Heidelberg; 2013, p. 307–18.
[28] Sharafaldin I, Lashkari AH, Ghorbani AA. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Intl conf. on information systems security and privacy. 2018, p. 108–16.
[29] Engelen G, Rimmer V, Joosen W. Troubleshooting an intrusion detection dataset: the CICIDS2017 case study. In: IEEE security and privacy workshops. 2021, p. 7–12.
[30] Apruzzese G, Laskov P, Tastemirova A. SoK: The impact of unlabelled data in cyberthreat detection. In: IEEE 7th european symposium on security and privacy. 2022, p. 20–42.
[31] Catillo M, Rak M, Villano U. 2L-ZED-IDS: A two-level anomaly detector for multiple attack classes. In: Workshops of the international conference on advanced information networking and applications. 2020, p. 687–96.
[32] Lima Filho FSd, Silveira FAF, Medeiros Brito Junior Ad, Vargas-Solar G, F. Silveira L. Smart detection: An online approach for DoS/DDoS attack detection using machine learning. In: Security and communication networks. 2019.
[33] Gamage S, Samarabandu J. Deep learning methods in network intrusion detection: A survey and an objective comparison. J Netw Comput Appl 2020;169.
[34] Akiba T, Sano S, Yanase T, Ohta T, Koyama M. Optuna: A next-generation hyperparameter optimization framework. 2019.
[35] Nicolae M-I, Sinn M, Tran MN, Buesser B, Rawat A, Wistuba M, et al. Adversarial robustness toolbox v1.0.0. 2019.

Ying-Dar Lin is a Chair Professor of computer science at National Yang Ming Chiao Tung University (NYCU), Taiwan. He received his Ph.D. in computer science from the University of California at Los Angeles (UCLA) in 1993. He was a visiting scholar at Cisco Systems in San Jose during 2007–2008, CEO at Telecom Technology Center, Taiwan, during 2010–2011, and Vice President of National Applied Research Labs (NARLabs), Taiwan, during 2017–2018. He was the founder and director of Network Benchmarking Lab (NBL) in 2002–2018, which reviewed network products with real traffic and automated tools, and has been an approved test lab of the Open Networking Foundation (ONF). He also cofounded L7 Networks Inc. in 2002, later acquired by D-Link Corp, and O'Prueba Inc. a spin-off from NBL, in 2018. His research interests include network security, wireless communications, network softwarization, and machine learning for communications. His work on multi-hop cellular was the first along this line, and has been cited over 1000 times and standardized into IEEE 802.11s, IEEE

802.15.5, IEEE 802.16j, and 3GPP LTE-Advanced. He is an IEEE Fellow (class of 2013), IEEE Distinguished Lecturer (2014–2017), ONF Research Associate (2014–2018), and received in 2017 Research Excellence Award and K. T. Li Breakthrough Award. He has served or is serving on the editorial boards of several IEEE journals and magazines, including Editor-in-Chief of IEEE Communications Surveys and Tutorials (COMST, 1/2017–12/2020). He published a textbook, Computer Networks: An Open Source Approach, with Ren-Hung Hwang and Fred Baker (McGraw-Hill, 2011).

Jehoshua Hanky Pratama received his M.S. degree in Electrical Engineering and Computer Science (EECS) International Graduate Program from National Yang Ming Chiao Tung (NYCU), Hsinchu, Taiwan, in 2022. He was an associate researcher at High Speed Network Lab, NYCU, in 2020–2022 and currently works as a full-time Software Engineer at Trend Power Technology Co., LTD in Hukou, Taiwan. His research interests include network security, machine learning, intrusion detection system, adversarial attack, and adversarial defense.

Didik Sudyana is a Ph.D. candidate at Electrical Engineering and Computer Science (EECS) International Graduate Program of National Yang Ming Chiao Tung University (NYCU). He received his M.S. degree in Informatics from Universitas Islam Indonesia (UII), Indonesia, in 2016. He is a lecturer and a researcher in Informatics at STMIK Amik Riau, Indonesia. His research interests include cybersecurity, machine learning, and network design and optimization.

Yuan-Cheng Lai received his Ph.D. degree in the Department of Computer and Information Science from National Chiao Tung University in 1997. He joined the faculty of the Department of Information Management at National Taiwan University of Science and Technology in August 2001 and has been a distinguished professor since June 2012. His research interests include performance analysis, software-defined networking, wireless networks, and IoT security.

Ren-Hung Hwang (Senior Member, IEEE) received his Ph.D. degree in computer science from the University of Massachusetts, Amherst, Massachusetts, USA, in 1993. He is the Dean of the College of Artificial Intelligence, National Yang Ming Chiao Tung University (NYCU), Taiwan. Before joining NYCU, he was with National Chung Cheng University, Taiwan, from 1993 to 2022. He is currently on the editorial boards of IEEE Communications Surveys and Tutorials and IEICE Transactions on Communications. He received the Best Paper Award from

The 6th International Conference on Internet of Vehicles 2019, IEEE Ubi-Media 2018, IEEE SC2 2017, IEEE IUCC 2014, and the IEEE Outstanding Paper Award from IEEE IC/ATC/ICA3PP 2012. He served as the general chair of the International Computer Symposium (ICS), 2016, and International Symposium on Pervasive Systems, Algorithms, and Networks (ISPAN) 2018, International Symposium on Computer, Consumer and Control (IS3C) 2018, IEEE DataCom 2019 (The 5th IEEE International Conference on Big Data Intelligence and Computing). His current research interests include in Deep Learning, Wireless Communications, Network Security, AIoT, and Cloud/Edge/Fog Computing.

Po-Ching Lin received the Ph.D. degree in computer science from the National Chiao Tung University, Hsinchu, Taiwan, in 2008. He joined the Faculty of the Department of Computer Science and Information Engineering, National Chung Cheng University, in August 2009. He is currently a Professor. His research interests include network security, network traffic analysis, and performance evaluation of network systems.

Hsuan-Yu Lin received his Ph.D. in electrical engineering from National Chung Cheng University (CCU), Chia-Yi, Taiwan, in 2006. He has worked for Telecom Technology Center (TTC) since 2006 and currently serves as the Deputy CEO leading the research divisions of wireless communications and cybersecurity. His research interests include adaptive signal processing, radio resource management, and the cybersecurity of mobile broadband systems. In addition, he was a leading engineer at Taiwan Mobile Co. (TWM) during 2000–2006, where he worked in mobile network design and performance optimization.

Wei-Bin Lee received his Ph.D. degree from National Chung Cheng University in 1997. Dr. Lee served as a professor in the Department of Information Engineering & Computer Science at Feng Chia University. He was also a visiting professor at both Carnegie Mellon University in the USA and University of British Columbia in Canada. Since 2021, he has served as the CEO of HonHai Research Institute as well as the director of the information security research center. Before joining the Foxconn group, the CEO Wei-Bin Lee held important positions in Taipei City Government, Taipei Fubon Bank, Fubon Financial Holdings, and Feng Chia University. His research experiences fall in network security, cryptography, digital rights management, and privacy/security management and governance.

Chen-Kuo Chiang is an associate professor at the Department of Computer Science and Information Engineering, National Chung Cheng University, Taiwan. He received his Ph.D degree in computer science department at National Tsing Hua University, Hsinchu, Taiwan, in 2011. He received his B.S. degree in the department of computer information science at National Chiao Tung University, Taiwan, in 1998, and M.S. degree in the department of computer science information engineering at National Taiwan University, Taiwan, in 2000. He joined Institute of Information Industry as a software engineer from 2001 to 2005. In 2009, he was a visiting scholar with Columbia University in New York. His research interests include computer vision, machine learning and pattern recognition.