ENAD: An Ensemble Framework for Unsupervised Network Anomaly Detection

Jingyi Liao, Sin G. Teo, Partha Pratim Kundu, Tram Truong-Huu
Institute for Infocomm Research (I^{2}R), Agency for Science, Technology and Research (A*STAR), Singapore
Email: liaoj@i2r.a-star.edu.sg, teosg@i2r.a-star.edu.sg, kundupp@i2r.a-star.edu.sg, truonght@i2r.a-star.edu.sg

2021 IEEE International Conference on Cyber Security and Resilience (CSR) | 978-1-6654-0285-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/CSR51186.2021.9527982

***Abstract*—Network anomaly detection is paramount to early detect traffic anomalies and protect networks against cyber attacks such as (distributed) denial of service attacks and phishing attacks. As deep learning has succeeded in various domains, it has been adopted for network anomaly detection using a supervised learning approach. Due to the high velocity and dynamics of network traffic, labeling such voluminous network data with specific domain knowledge is difficult, and yet impossible. It makes supervised learning techniques become impractical. Several existing works have proposed unsupervised learning techniques to train detection models with unlabeled data. However, a single model cannot detect all types of attacking traffic due to the variety of their behavior. In this work, we develop an ensemble framework that uses different AutoEncoders (AEs) and generative adversarial networks (GANs) for network anomaly detection. We develop a weighting scheme that allows us to quantify the importance (goodness) of each model to each attacking traffic and then determine the final prediction score during the inference (detection) phase. We carry out extensive experiments on two recent datasets including UNSW-NB15 and CICIDS2017 to demonstrate the effectiveness of the proposed framework. The experimental results have shown that our framework significantly outperforms many state-of-the-art methods with an increase of up to 14.70% in various performance metrics such as precision, recall, F1-measure, AUROC and AUPRC.**

***Index Terms*—Ensemble Method, Unsupervised Deep Learning, Network Anomaly Detection, Network Security**

I. Introduction

Computer networks have been rapidly increasing in size due to the development of network applications whose endpoints are connected through various communication protocols. Such complex networks are vulnerable to many cyber attacks, e.g., distributed denial of service attacks (DDoS), eavesdropping attacks, malware attacks, etc. These attacks have caused significant damages to the targeted victims, including governments and many business organizations, in the form of revenue, reputation and data loss. Depending on the objectives, attackers employ different tools to exploit network vulnerabilities and launch the attacks from outside or inside the victim networks. Therefore, securing the networks against such attacks is very challenging as there is no single solution for all types of attacks due to their different characteristics and traffic behavior.

To detect attacks early, we usually apply network anomaly detection at the first place and then perform further analysis to identify the type of attacks or attribution. The abnormal traffic behavior generally deviates from the expected pattern of normal network activities with a certain margin. But it is worth to notice that in some cases, a normal user could create anomalies such as typing wrong password multiple times in logged-in screen. However, most of the abnormal traffic behavior is caused by the network traffic created by attacking agents, e.g., a bot sends multiple port scanning packets to look for vulnerable ports on a host. Network anomalies can be detected using a host-based or network-centric approach. In the first approach, a detector is deployed in each host while, in the latter, network traffic is captured at first and then useful features are extracted and analyzed to detect anomalies. Due to the increase in network size and the corresponding increment in network traffic, the host-based detection approach is no longer scalable. Hence, a practical solution is to use the network-centric approach to capture network data for anomaly detection. With a proper deployment location, e.g., at the border of the networks, the detector will have global visibility of the networks and accommodate the heterogeneity of connected devices and their network traffic behavior.

As deep learning has great success in various domains such as computer vision, natural language processing, and robotics, it has gained attention from academia and industry for its application in network security. It has been used to analyze high-dimensional traffic features. The anomaly detection models are trained in 3 different ways: supervised, semi-supervised and unsupervised learning [1]–[7]. In [3], the authors demonstrated that deep learning methods are superior to conventional machine learning approaches such as Decision Tree (DT), Support-vector Machine (SVM) and k-Nearest Neighbour (k-NN) [8]. However, supervised and semi-supervised techniques require labeled data, including a similar number of benign and anomalous traffic instances, which is hard to obtain in practice since labeling high-speed network traffic is challenging and anomalous traffic is rare in practice. Unsupervised learning techniques using generative models, e.g., autoencoders (AE), variational autoencoders (VAE) and generative adversarial networks (GAN), overcome this limitation by learning from unlabeled data. Still, their performance is affected by the variety of attacking traffic behaviors caused by sophisticated tools used by the attackers to manipulate attacking traffic. It leads to the fact that there is no single model that can detect all types of anomalous traffic. This motivates us to develop a novel framework using an ensemble of models to improve network anomaly detection performance.

The recent success of ensemble learning [9] has demonstrated the generalization ability by aggregating multiple mod-

978-1-6654-0285-9/21/$31.00 ©2021 IEEE

els to serve a common task. In this work, we propose an ensemble framework that leverages the advantage of unsupervised learning for network anomaly detection. We develop multiple deep learning models using AE, VAE and GAN, which are built using either fully-connected neural networks or 1-dimensional convolution neural networks (1D-CNNs). The main contributions of our work are as follows.

1) We develop an Ensemble Framework for Unsupervised Network Anomaly Detection (ENAD) using different AEs and GAN models. Given the prediction of individual models, we develop a weighting scheme that allows the proposed framework to quantify the importance of each individual model when determining the final prediction of a sample during the inference phase.
2) We employ the Mahalanobis distance which has been shown to be superior to conventional Euclidean distance in anomaly detection into AEs and GAN to improve the individual model learning capability to differentiate benign samples from anomalous samples in a better way.
3) We carry out extensive experiments and provide a detailed performance analysis on two recent datasets, namely UNSW-NB15 [10] and CICIDS2017 [11].

The rest of this paper is organized as follows. We discuss the related work in Section II. Section III provides the background of unsupervised deep learning techniques for network anomaly detection. Section IV presents the proposed ensemble framework. Section V presents the experiments and performance analysis before we conclude the paper in Section VI.

II. Related Work

There exist several works using supervised or semi-supervised learning for network anomaly detection. In [12], the authors proposed a flow-based approach using Self Organizing Maps to detect network anomalies. In [8] and [13], the authors used different supervised learning techniques to identify malicious traffics inside the network. While [8] used k-NN, Extreme Learning Machine, and Hierarchical Extreme Learning Machine algorithms, [13] used a meta-heuristic Bayesian network approach to classify traffic packets into either normal or anomalous traffic. In [1], the authors combined convolutional neural networks (CNN) and long short-term memory (LSTM) into a model called C-LSTM to detect anomalies in web traffic. In [14], the authors combined unsupervised and supervised machine learning approaches for intrusion detection. In [3], the authors compared the performance of conventional supervised machine learning (e.g., DT, Random Forest (RF) and SVM) and deep learning (e.g., CNN and AE) to classify network anomalies. Similarly, in [7], the authors used the NSLKDD dataset to compare the performance of the conventional machine learning algorithms (e.g., RF and SVM). While those supervised and semi-supervised learning techniques achieved high performance, they require labeled datasets that have a balancing number of samples in each traffic family. In high-speed networks, labeling such voluminous traffic is a time-consuming task and anomalous traffic is usually rare. Thus, obtaining a desired dataset is difficult and impractical.

Unsupervised learning techniques have been developed to solve the issues of supervised learning techniques [15]–[17]. In [18], the authors presented a kernel-based learning algorithm for online DDoS attack detection. Recently, GAN has been adopted for this problem [6]. However, network attacks keep changing their nature with new sophisticated techniques, making it harder to detect network anomalies by a single network detection model. Hence, the authors of [19] proposed an ensemble method, in which they used multiple AEs with the same architecture to analyze different traffic segments and detect network anomalies. Our work differs from [19] in the sense that we combine different models of AEs and GAN, which also use different types of neural networks to detect network anomalies. This adds diversification in order to build models that can generalize better for network anomalies.

III. Background

A. Generative Models

1) AutoEncoder and Variational AutoEncoder: In AE [20], an encoder ($E_{\boldsymbol{\phi}}$) compresses a real sample $\boldsymbol{x} \in \mathcal{X}$ into a latent variable $\boldsymbol{z} \in \mathcal{Z}$ and then a decoder ($D_{\boldsymbol{\eta}}$) restores a sample $\hat{\boldsymbol{x}} \in \mathcal{X}$ from the latent variable $\boldsymbol{z}$ where $E_{\boldsymbol{\phi}}$ and $D_{\boldsymbol{\eta}}$ are two neural networks with trainable parameters $\boldsymbol{\phi}$ and $\boldsymbol{\eta}$, respectively. The objective of AE is to optimize parameters $\boldsymbol{\phi}$ and $\boldsymbol{\eta}$ so that given a sample $\boldsymbol{x}$, it is able to reconstruct sample $\hat{\boldsymbol{x}}$ that is identical or the most similar to $\boldsymbol{x}$. The drawback of AE is that the latent space is usually not guaranteed to have useful properties that enable other advanced generative processes. To address this issue, the encoder of VAE [21] uses a distribution by generating two probability distribution parameters, mean and standard deviation (μ and σ), instead of generating a single latent variable $\boldsymbol{z}$. The decoder of VAE performs sampling from the distribution to reconstruct the original sample. The VAE model is fully differentiable by the "reparameterization trick", and thus being optimized in an end-to-end manner. Specifically, sampling a latent variable $\boldsymbol{z}$ from distribution $N(\mu, \sigma)$ is equivalent to sampling a parameter ε from a distribution $N(0, 1)$ that makes $\boldsymbol{z} = \mu + \varepsilon * \sigma$.

2) Generative Adversarial Networks: GAN [22] has emerged as useful generative models that can implicitly learn data distributions of arbitrarily complex dimensions. A standard GAN composes of a generator and a discriminator, which are neural networks and their parameters are trainable. We define $G_{\boldsymbol{\theta}}$ and $D_{\boldsymbol{\psi}}$ to be the generator and discriminator networks. During the training, the generator and discriminator play a minimax game where the generator $G_{\boldsymbol{\theta}}$ aims to generate a new synthetic sample from a latent variable $\boldsymbol{z} \in \mathcal{Z}$, whereas the discriminator aims to differentiate between real sample $\boldsymbol{x}$ and synthetic sample $G_{\boldsymbol{\theta}}(\boldsymbol{z})$. The objective of the GAN training is solving saddle point problem defined as follows:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\psi}} \mathcal{L}_{\text{GAN}}(\boldsymbol{\theta}, \boldsymbol{\psi}) = \mathbb{E}_{\boldsymbol{x} \sim p_{\mathcal{X}}(\boldsymbol{x})}[\log D_{\boldsymbol{\psi}}(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\mathcal{Z}}(\boldsymbol{z})}[\log(1 - D_{\boldsymbol{\psi}}(G_{\boldsymbol{\theta}}(\boldsymbol{z})))] \quad (1)$$

where $\boldsymbol{x} \sim p_{\mathcal{X}}(\boldsymbol{x})$ is the distribution of the real samples while $\boldsymbol{z} \sim p_{\mathcal{Z}}(\boldsymbol{z})$ is the distribution of the latent variable.

Various GAN models have been developed to improve GAN performance and address several issues such as highly unstable and sensitive training in which the loss functions of both the discriminator and generator with respect to their parameters tend to oscillate wildly during training [23]. In this work, we adopt adversarially-learned anomaly detection (ALAD) [6], a bidirectional GAN, which can simultaneously learn mutually coherent and two-way bijective mappings in order to generate high-quality samples in both the latent and data spaces. Specifically, apart from the generator and discriminator, bidirectional GAN is additionally equipped an encoder that is also a neural network. The encoder is trained to infer the best random value in the latent space that best matches a given data sample.

B. Unsupervised Network Anomaly Detection

Generative models can be applied to unsupervised network anomaly detection using a reconstruction-based approach. We argue that if the generative models have successfully learned the data distribution (i.e., normal network traffic behavior), it is reasonable to assume that, the models would be able to generate the samples that are similar to those in the training set, resulting in small reconstruction errors between original samples and generated ones. In contrast, given an anomalous sample, which the models have not seen during the training, the sample might be poorly reconstructed, resulting in a large reconstruction error. By comparing the reconstruction error between the real sample and the generated sample against a predefined threshold (θ), we then can decide whether the given data sample is anomalous or not. In other words, as the models are trained with normal (benign) traffic samples, they are able to detect novelty also known as "zero-day" attacks.

1) Computation of Reconstruction Error using Mahalanobis Distance: Mahalanobis distance is a measure of the distance between a data point and a distribution, which is usually a multivariate Gaussian distribution. Given a data sample $\boldsymbol{x}$, its Mahalanobis distance is computed as follows:

$$D(\boldsymbol{x}) = \sqrt{(\boldsymbol{x} - \mu)^T \Sigma^{-1} (\boldsymbol{x} - \mu)} \tag{2}$$

where μ and Σ are the mean vector and covariance matrix of the distribution, respectively. It is to be noted that Mahalanobis distance is scale-invariant and does not assume a common scale of the data distribution. It accounts for correlations between dimensions of the data used to estimate the distribution, therefore being more effective than Euclidean distance in multivariate anomaly detection problems [24].

In this work, we adopt Mahalanobis distance to compute the reconstruction error of a sample for all generative models used in the ensemble framework. For AEs and VAEs, we use Eq. (2) to compute the reconstruction error of a sample. Specifically,

$$D(\boldsymbol{x}) = \sqrt{(E(\boldsymbol{x}) - \mu_{\text{train}})^T \Sigma_{\text{train}}^{-1} (E(\boldsymbol{x}) - \mu_{\text{train}})} \tag{3}$$

where $E(\boldsymbol{x})$ is the latent representation of sample $\boldsymbol{x}$ (i.e., the output of the encoder). μ_{train} and Σ_{train} are the mean vector and covariance matrix of the encoded training data.

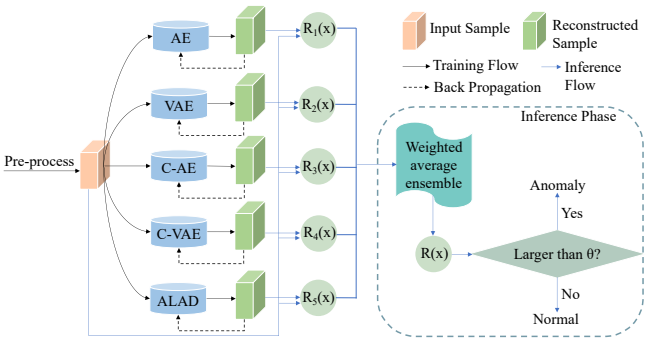


Fig. 1. Ensemble framework for unsupervised network anomaly detection.

Since we use a multivariate standard normal distribution to generate random variable $\boldsymbol{z}$ during the training of GAN models, we will have a zero vector as the mean and an identity matrix as the covariance matrix of the distribution. The reconstruction error of sample $\boldsymbol{x}$ computed by the Mahalanobis distance is defined as $D(\boldsymbol{x}) = \sqrt{(E(\boldsymbol{x})^T E(\boldsymbol{x})}$.

2) Determination of Anomaly Threshold: There are several ways to choose the threshold of the anomaly score to determine whether a sample is anomalous or not. If the training set contains only normal data samples, one may use the highest reconstruction error among the samples in the training set as the threshold. In practice, the training set may contain a small percentage (e.g., 5%) of the data as anomalous samples. One can also use this percentage as the threshold of the anomaly score. The smaller the threshold, the higher the false positive rate. If the labeled data is available, we can make use of it to determine the threshold so as to achieve a high detection rate while having a small false positive rate. When deploying the models to realistic systems, the anomaly threshold is usually determined in a heuristic manner. The anomaly threshold is adjusted based on the system performance and the desired requirement of the network administrator such as achieving a high accuracy or high true positive rate.

IV. Ensemble Framework for Unsupervised Network Anomaly Detection (ENAD)

In this section, we first present the design of the framework (ENAD) and then we present the weighted average ensemble approach used to calculate the final prediction of a sample given the prediction of the individual models in the framework.

A. Framework Design

The design of the proposed framework is depicted in Fig. 1. The first important component of the framework is the data pre-processing. In practice, the traffic stream must be processed before being feed to a machine learning or deep learning model for analysis. In this pre-processing procedure, traffic features will be extracted and statistical features will be computed. If the categorical features are used to train the model, those features must be encoded to numerical values so that the model can process. Furthermore, due to the heterogeneity of traffic features, their values must also be normalized so as to eliminate the dimensional influence

between the features. Extracted traffic features (training data set or testing data set) are learned and analyzed by five unsupervised learning models in the proposed framework. The details of the individual models are given below:

- **AE** uses fully-connected neural networks for its encoder and decoder.
- **VAE** uses fully-connected neural networks for its encoder and decoder. The output of the encoder is passed through two dense (fully-connected) layers of the same size, generating two output vectors for the mean (μ) and standard deviation (σ) of the distribution of the data.
- **C-AE** is an AE whose encoder and decoder are 1D-CNN. To reduce the dimension of the feature vector in the encoder, we use convolution layers with $\texttt{stride} = 1$ instead of using pooling layers. We note that a strided convolution can also be seen as learning the pooling operation, which increases the expressiveness ability of the model. Furthermore, our experiment shows that models with strided convolution layers outperform the models with pooling layers.
- **C-VAE** is similar to VAE but its encoder and decoder are CNN as in C-AE.
- **ALAD** is a bidirectional GAN model as we discussed in Section III. All the components of ALAD use a fully-connected neural network.

Training these models can be done independently without any correlation among them. Given sufficient computing resources, they can also be trained in parallel. While the training time of the models is a one-time cost, it could take hours to train a model to converge to its best performance. Since the models are heterogeneous in their architecture and hyper-parameters, the training time of each model varies depending on its complexity, e.g., ALAD has the most complex architecture, thus requiring the longest training time.

In the inference phase, given a traffic sample to decide whether it is anomalous or not, the sample will be passed through all models to obtain its reconstructed samples, each corresponding to the output of each model. Using Mahalanobis distance, we compute the anomaly score of the sample based on its reconstructions generated by the individual models. The anomaly scores are then gathered by a weighted average ensemble operation to compute the final anomaly score before being compared to a pre-defined threshold in order to decide the nature of the sample. We note that the anomaly score of a sample may be different in various models. To avoid the model dominance that results in a higher anomaly score than others, we normalize the anomaly score using the $\min$ and $\max$ values obtained from the samples in the training set.

It is to be noted that the detection speed depends on the duration of traffic feature collection determined by the administrator (i.e., 3 mins for UNSW-NB15) to form a traffic record for analysis. Given a traffic record, the inference phase is simply a linear operation of applying the traffic feature values to the model. Thus, the ensemble framework does not incur significant overhead.

B. Weighted Average Ensemble for Final Prediction

Weighted average ensemble is an approach of model averaging in ensemble learning where the final prediction of the ensemble learning is weighted by the performance of each individual model in the ensemble framework. In a multi-class classification problem using the supervised ensemble learning with M classes $(\ell_1, \ldots, \ell_i, \ldots, \ell_M)$ and N individual models $(j = 1, \ldots, N)$, the final prediction of a sample is computed using a closed-form expression as follows:

$$\mathbf{P}(\ell_i) = \frac{\sum_{j=1}^{N} w_{i,j} P_j(\ell_i)}{\sum_{i=1}^{M} \sum_{j=1}^{N} w_{i,j} P_j(\ell_i)} \tag{4}$$

where $w_{i,j}$ is the weight of model j to class ℓ_i and $P_j(\ell_i)$ is the prediction probability of model j that the examined sample belongs to class ℓ_i. The final prediction of the examined sample will be class ℓ_i that has the highest value of $\mathbf{P}(\ell_i)$. A challenging question is how to determine the weight $w_{j,i}$ of model j to class ℓ_i in the ensemble framework. A naive solution is to assign equal weight to each model, i.e., every model has the same importance to decide the final prediction ($w_{j,i} = 1$). However, this approach does not optimize the model performance as a model would be good at predicting one class but not another class. In a supervised ensemble learning problem, this can be solved using the training set to determine the weight of each model during the training.

In an unsupervised anomaly detection problem, we cannot directly adopt the above approach as a reconstruction-based approach does not provide the probability of a sample being benign or malicious. Furthermore, as training set contains only benign samples, we cannot determine the weight of the individual models during the training phase of the ensemble framework. To overcome this issue, we use grid search to find the optimal weight of each model. As grid search is an exhaustive method, we reduce the requirement of computational resources by limiting the search space with additional conditions such as all the values have to be in the range of $[0, 1]$ and sum of the values must be 1. We use a validation set that includes both benign and anomalous samples for just optimizing the weight of the individual models. It is to be noted that this validation set is different from the conventional validation set used to improve the performance of the models during the training. We use this validation set only for determining the model weights only after all the individual models have been trained. Given sample $\boldsymbol{x}$ and the normalized anomaly score (reconstruction error) generated by N models $R_j(\boldsymbol{x})$ where $j = 1, \ldots, N$, the final anomaly score of sample $\boldsymbol{x}$ is computed as $R(\boldsymbol{x}) = \sum_{j=1}^{N} w_j R_j(\boldsymbol{x})$ where w_j is optimized using the grid search approach as presented above. $R(\boldsymbol{x})$ is then compared to a pre-defined threshold to determine the nature of the sample.

V. Experiment

A. Datasets and Experiment Setup

1) UNSW-NB15: To collect this dataset, Moustafa *et al.* [10] used a traffic generation appliance called IXIA Per-

TABLE I
DISTRIBUTION OF THE TEST SET OF UNSW-NB15

Network Type	Number of Data Points	Percentage
Benign	399247	87.320
Generic	38855	8.500
Exploits	8106	1.773
Fuzzers	4307	0.942
DoS	2940	0.643
Reconnaissance	2555	0.559
Analysis	493	0.109
Backdoor	423	0.093
Shell-code	264	0.058
Worms	32	0.007

fectStorm, which has implemented various attacking traffic families including Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shell-code and Worms. They collected the raw traffic using `tcpdump` traces for a total of 31 hours at the beginning of 2015, resulting in around 2 million flows. Using Argus[1], a tool for traffic feature extraction and aggregation, the authors extracted 49 features including both basic features such as flow duration, flow size, and statistical features such as mean packet size, the standard deviation of packet size, etc. The four features of source IP address, destination IP address, source port, destination port are used as identification of the flow and are not used for training detection models. The authors also provided the label for each flow whether it is benign or anomalous along with its attack family, which are used during the test phase for measuring the performance of the detection models and the proposed ensemble framework. Table I presents the distribution of the test set of UNSW-NB15. The percentage of anomalous traffic is 13%, which helps us to determine threshold θ for anomaly scores for this dataset.

2) CICIDS2017: To collect this dataset, Sharafaldin *et al.* [11] designed and developed two networks, namely the attacking network and the victim network. The victim network is a highly-secured infrastructure with firewalls, routers, switches and operating systems. The attacking network is a completely separated infrastructure consisting of a router, a switch and a set of computers with public IP addresses and different operating systems to run attacking scenarios. To generate attacking traffic, the authors implemented 6 attack profiles and execute them on the attacking network by using related publicly available tools and codes. The implemented attacks include Inflitration, Brute Force FTP, DoS, Web Attack, Botnet, Brute Force SSH, DDoS, and Heartbleed. The authors run the traffic generation for 5 days from Monday to Friday (July 3rd – 7th), 2017. Attacks were subsequently executed during this period. Similar to UNSW-NB15, the authors captured raw traffic and saved it into files before using CICFlowMeter[2] tool to extract traffic features. Table II represents the distribution of records of the test set of CICIDS2017. The percentage of anomalous traffic is 19.6%. As shown in Table I and Table II, both

[1]Argus: https://openargus.org/ [Accessed on: 20 Feb. 2021]

[2]CICFlowMeter: https://www.unb.ca/cic/research/applications.html\#CICFlowMeter [Accessed on: 25 Feb. 2021]

TABLE II
DISTRIBUTION OF THE TEST SET OF CICIDS2017

Network Type	Number of Data Points	Percentage
Benign	407309	80.500
DoS Hulk	40020	7.910
PortScan	28672	5.667
DDoS	22937	4.533
DoS GoldenEye	1806	0.360
FTP-Patator	1386	0.274
SSH-Patator	1099	0.217
DoS slowloris	996	0.197
DoS Slowhttptest	983	0.194
Bot	375	0.074
Web Attack Brute Force	270	0.053
Web Attack XSS	118	0.023
Infiltration	8	0.002
Heartbleed	4	0.001
Web Attack Sql Injection	3	0.001

datasets are significantly imbalanced in which the minority class (i.e., anomalous traffic) is more important to be detected. We randomly split the samples into the training set and test set with a ratio 80 : 20. We discarded anomalous samples from the training set, thus setting up a novelty detection task, also known as "zero-positive" anomaly detection. From the test set, we also randomly selected 10% of the samples to use as the validation set to optimize the weight of the models for the weighted average ensemble.

3) Experiment Setup: The experiments were carried out on a customized desktop with Intel®Core i9-10900X CPU @ 3.70GHz × 20, 64 GB of RAM and 2 NVIDIA GeForce RTX 2080Ti GPUs, each having 11 GB of memory. To ensure reproducibility, we conducted each experiment over multiple random-seeded runs. All the results presented below were obtained on the test sets of the two datasets.

B. Evaluation Metrics and Comparison

Although accuracy is the most commonly used evaluation metric, it cannot meet the needs of all classification problems, especially for those dealing with imbalanced datasets. In an anomaly detection problem, we are more concerned about how many anomalous samples are identified while reducing as many as possible the number of false positives. Therefore, precision and recall are used to evaluate the performance of the proposed framework: precision indicates the actual positive samples among the samples predicted by the model as positive (i.e., $Precision = TP/(TP+FP)$) where TP is the number of actual positive samples and FP is the number of false positive samples. However, a high-precision model (i.e., with very low FP) does not mean it is effective. Thus, we need the recall metric to describes the sensitivity of the model by computing the fraction of the positive samples predicted by the models among the actual positive samples available in the test data set (i.e., $Recall = TP/(TP + FN)$) where FN is the number of false negatives. Precision and Recall both are in a contradictory relationship with each other. If Precision increases, Recall gets decreased and vice versa. The higher the Precision, the lower the Recall. Thus, in order to characterize the comprehensive performance of the model, we also use

TABLE III
PERFORMANCE COMPARISON WITH THREE EVALUATION METRICS

Method	UNSW-NB15			CICIDS2017		
	Prec.	Recall	F1-Score	Prec.	Recall	F1-Score
DAGMM	0.535	0.527	0.531	0.764	0.765	0.765
AnoGAN	0.434	0.439	0.437	0.144	0.144	0.144
AE	0.928	0.878	0.901	0.797	0.803	0.800
VAE	0.961	0.907	0.932	0.798	0.804	0.801
C-AE	0.975	0.920	0.946	0.716	0.720	0.718
C-VAE	0.977	0.922	0.947	0.769	0.774	0.772
ALAD	0.970	0.916	0.941	0.705	0.709	0.707
ENAD	**0.979**	**0.924**	**0.949**	**0.838**	**0.840**	**0.835**

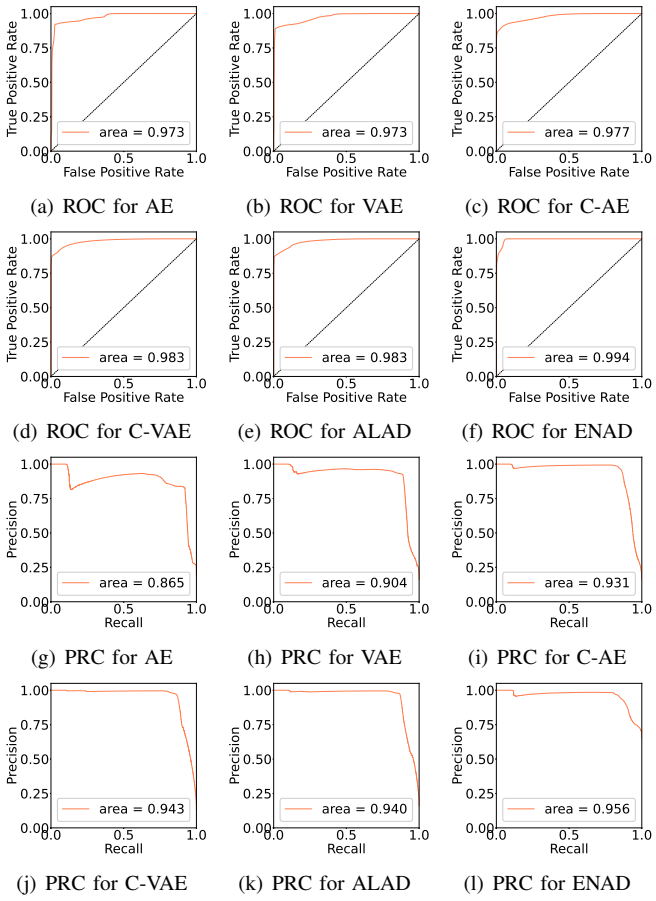
F1 Score, Area Under the Receiver Operating Characteristic (AUROC) and Area Under Precision-Recall Curve (AUPRC). We compare the performance of ENAD with that of the individual models incorporated in the framework. We also compare ENAD with two other state-of-the-art models:

- Deep Autoencoding Gaussian Mixture Model (DAGMM): It is an autoencoder-based approach for anomaly detection. DAGMM consists of an autoencoder and an estimator. While the autoencoder is trained to generate a sensible latent space and reconstruction features, the estimator is trained to output parameters of a Gaussian mixture model (GMM) that models the lower-dimensional latent space. At inference time, the learned GMM computes the likelihood of a sample based on its latent and reconstruction features. The computed likelihood is used as the anomaly score to determine whether the sample is an anomaly or not.
- AnoGAN: It is a standard GAN with only a single generator and a discriminator adopted for the anomaly detection problem [25]. As AnoGAN does not have the encoder, given a sample for inference, AnoGAN randomly selects the latent variables to feed the generator to reconstruct the sample. As this is a greedy approach, AnoGAN may not achieve the best reconstruction due to the limit of the search space and computational time.

We refer the reader to [26] for the detailed description of the architecture of deep neural networks used in our framework.

C. Performance Analysis

1) Overall Performance: Table III presents the overall performance of all the individual models and the ensemble model on the testing datasets. We observe that the ensemble framework (ENAD) achieves the highest performance in all the metrics for all the datasets. On the UNSW-NB15 dataset, C-AE and C-VAE are the second best followed by ALAD. The improvement of the result is more significant on the CICIDS2017 dataset. ENAD improves performance by up to 4.0% for all the metrics. Interestingly, C-AE and C-VAE are not the second best performance on the CICIDS2017 dataset. Instead, AE and VAE with fully-connected neural networks have a better performance compared to C-AE and C-VAE. This further demonstrates the effectiveness of the ensemble framework as there is no single model that is good for different

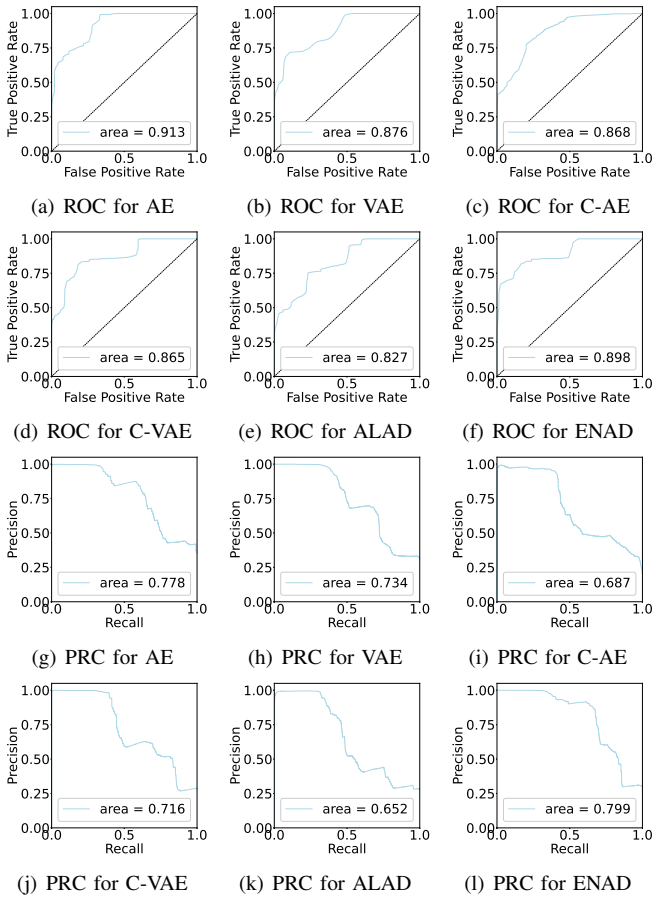


(a) ROC for AE (b) ROC for VAE (c) ROC for C-AE
(d) ROC for C-VAE (e) ROC for ALAD (f) ROC for ENAD
(g) PRC for AE (h) PRC for VAE (i) PRC for C-AE
(j) PRC for C-VAE (k) PRC for ALAD (l) PRC for ENAD

Fig. 2. ROC and PRC curves of the models on the UNSW-NB15 dataset.

datasets. We also observe that there is a performance drop on the CICIDS2017 dataset compared to the UNSW-NB15 dataset. This can be explained by the fact that CICIDS2017 has twice the number of features compared to UNSW-NB15. Generally, more features will increase the difficulty for the models to learn and correlate the features, thus being more difficult to achieve high performance. We note that since DAGMM and AnoGAN perform the worst for both datasets, we do not include them to the ensemble framework.

2) AUROC and AUPRC of the Models: ROC curve represents the trade-off between the true positive rate (TPR) and false positive rate (FPR). Each point on the ROC curve corresponds to an anomaly threshold (θ) that results in a TPR and FPR. Ideally, TPR should be close to 1 and FPR should be close to 0. Therefore the curve should stay away above from the 45-degree line and the Area Under the Curve (AUC) should be maximized. As shown in Fig. 2, ENAD has the best performance compared to the individual models. Compared to the second best model, ENAD increases AUROC by 1.1%, and to the worst model, ENAD increases AUROC by 2.2%. However, ROC curve will not change with the change in the category distribution. Anomaly detection has to deal with highly imbalanced data, in which the large number of negative samples makes the growth of FPR not obvious as shown in Fig. 3 for the CICIDS2017 dataset. In this case, the



(a) ROC for AE (b) ROC for VAE (c) ROC for C-AE

(d) ROC for C-VAE (e) ROC for ALAD (f) ROC for ENAD

(g) PRC for AE (h) PRC for VAE (i) PRC for C-AE

(j) PRC for C-VAE (k) PRC for ALAD (l) PRC for ENAD

Fig. 3. ROC and PRC curve of the models on the CICIDS2017 dataset.

Precision-Recall curve can evaluate the model performance in a more accurate way. As shown in Fig. 3, ENAD also outperforms the individual models. Compared to the best and worst individual models on AUPRC, ENAD improves performance by 2.1% and 14.7%, respectively. This shows that ENAD can better generalize the detection of anomalies based on statistics learned from the normal traffic.

3) Performance Analysis on Individual Traffic Categories: To further demonstrate the efficacy of the proposed framework, we also calculated the individual detection accuracy for every traffic category. As shown in Table IV, several categories are difficult to detect for all the models as they do not have sufficient characteristics to differentiate from normal traffic. For the UNSW-NB15 dataset, all the models perform the worst on Shell-code due to the fact that Shell-code behaves similarly to normal traffic and its abnormal behavior mainly reflects on its payload, which needs deep packet inspection to distinguish. Fuzzers attacks are designed to feed a string of random codes to find out the flaws of the target network. Therefore, the traffic generated by Fuzzers attacks is similar to normal traffic. Reconnaissance also shares similar characteristics.

We observed that each model is good at detecting a different traffic family. In the UNSW-NB15 dataset, Generic takes up the largest proportion of anomalous traffic, followed by Exploits. Among the individual models, C-AE and C-VAE perform the best on Generic traffic by achieving an accuracy of

TABLE IV
ACCURACY FOR EACH CATEGORY IN THE UNSW-NB15 DATASET

Category	AE	VAE	C-AE	C-VAE	ALAD	ENAD
Benign	0.993	0.986	0.997	0.997	0.995	**0.998**
Generic	0.938	0.864	0.990	0.990	0.988	**0.990**
Exploits	0.721	0.554	0.707	0.710	0.687	**0.732**
Fuzzers	0.264	**0.555**	0.183	0.183	0.179	0.194
Dos	**0.875**	0.852	0.861	0.867	0.839	0.859
Reconn.	0.353	**0.384**	0.281	0.325	0.217	0.324
Analysis	0.819	**0.856**	0.804	0.808	0.790	0.797
Backdoor	0.890	0.876	0.877	0.886	0.884	**0.890**
Shell-code	0.004	**0.024**	0.012	0.008	0.028	0.016
Worms	0.714	**0.886**	0.357	0.743	0.200	0.686

TABLE V
ACCURACY FOR EACH CATEGORY IN THE CICIDS2017 DATASET

Category	AE	VAE	C-AE	C-VAE	ALAD	ENAD
Benign	0.918	0.918	0.886	0.908	0.881	**0.930**
DoS Hulk	0.815	0.752	**0.919**	0.918	0.793	0.870
PortScan	0.512	**0.515**	0.015	0.391	0.033	0.513
DDoS	0.674	**0.802**	0.533	0.466	0.722	0.786
DoS GoldenEye	0.914	**0.977**	0.525	0.500	0.971	0.932
FTP-Patator	0.899	0.522	0.996	**1.000**	0.014	0.999
SSH-Patator	0.508	0.502	0.893	**0.993**	0.237	0.508
DoS Slowloris	0.614	**0.706**	0.664	0.669	0.629	0.704
DoS Slowhttptest	0.946	0.946	0.950	0.412	**0.976**	0.946
Bot	0.257	0.048	**0.738**	0.416	0.058	0.340
Web Att. B. Force	0.050	0.084	0.080	0.080	0.050	**0.104**
Web Att. XSS	0.023	**0.046**	0.038	0.038	0.031	0.023
Infiltration	1.000	1.000	1.000	1.000	1.000	**1.000**
Heartbleed	1.000	1.000	1.000	1.000	1.000	**1.000**
Web Att. Sql Inj.	0.000	0.000	1.000	**1.000**	0.000	0.667

0.99, while the other three models also have good performance between 0.86 and 0.98. Combining those individual models into the ensemble framework results in an improved performance with an accuracy of 0.998. ENAD achieves similar performance with Backdoor traffic. For Exploits attacks, the models have different performance. The highest performance is achieved by AE with an accuracy of 0.721 while VAE can correctly detect half of this type of traffic samples. In this case, our ensemble framework can learn from the good models and ignore the bad ones based on the weighted average ensemble, which results in a better performance (an accuracy of 0.732) compared to the individual models. For the remaining traffic families, ENAD performs second best compared to the individual models in terms of detection accuracy. Nevertheless, it has the best overall performance as shown in Table III.

For CICIDS2017, the performance of each model on different categories is shown in Table V. The traffic categories that have the most number of samples are DoS Holk, DDoS and PortScan. C-AE and C-VAE can detect DoS Hulk attacks with an accuracy of 0.918, but they do not perform well on PortScan and DDoS (i.e., 0.015, 0.391, respectively). In contrast, VAE has a superior performance of detecting DDoS and PortScan attacks over other models, it can reach an accuracy of 0.802 on DDoS and 0.515 on PortScan, but its performance on DoS Hulk is relatively lower than C-AE and C-VAE. For these three categories, ENAD can precisely filter the information so that the decision from the model that performs poorly on a traffic type could be discarded or given lower conviction. Similar

TABLE VI
PERFORMANCE IMPACT OF LOSS FUNCTIONS

Datasets	Training	Testing	Precision	Recall	F1-Score
UNSW-NB15	MSE	MSE	0.954	0.902	0.926
	MSE	MAH	0.963	0.910	0.935
	MAH	MSE	**0.979**	**0.924**	**0.949**
	MAH	MAH	**0.979**	0.923	**0.949**
CICIDS2017	MSE	MSE	0.797	0.803	0.800
	MSE	MAH	0.781	0.798	0.796
	MAH	MSE	**0.838**	**0.840**	**0.835**
	MAH	MAH	0.811	0.812	0.814

performance also applies to other traffic families such as DoS GoldenEye, FTP-Patator, DoS Slowhttptest, Infiltration, Heartbleed and Web Attack SQL Injection. Therefore, our ensemble model can performs close to the best performing model on each particular traffic family. We note that PortScan traffic behaves similar to the normal TCP connections, using a large amount of TCP semi-connection to search for opened ports, resulting in low performance. Last but not least, we note that the weight of the models is determined by the optimization of overall performance, which is dominated by the traffic families that have the most number of samples in the datasets. While ENAD does not perform the best for a particular attacking traffic family, in general, it can be seen that ENAD can leverage the advantages of each individual model and largely avoids their shortcomings so as to yield the best overall performance.

4) Performance Impact of Loss Functions: In this analysis, we measure the performance impact of the loss functions used in the proposed framework (ENAD). For every individual model, we use Mahalanobis distance (MAH) or Mean Squared Error (MSE) for the training phase or testing phase. In Table VI, we present the performance of ENAD for different combinations of loss functions for training and testing phases (Column 2 and Column 3 indicate the loss function for all the individual models). The results show that using MAH in the training phase and MSE in the testing phase results in the best performance on both datasets. This shows that MAH is a good choice for training generative models to detect the novelties, which are not present in the training set.

VI. CONCLUSION

In this paper, we developed ENAD, an ensemble framework for unsupervised network anomaly detection. We developed five unsupervised learning models that can effectively detect network anomalies and integrated them into the ensemble framework. To determine the final prediction of a traffic sample, we developed a weighting scheme that allows the framework to quantify each model goodness and then predicts the label of the sample based on the output of the individual models. The great advantage of the proposed ensemble framework is to utilize unsupervised learning techniques, thus not requiring labeled data for model training. This makes the proposed ensemble framework more practical compared to the existing techniques. We carried out extensive experiments on recently-collected datasets: UNSW-NB15 and CICIDS2017. The experimental results have shown that the proposed framework significantly outperforms the state-of-the-art methods by 14.70% in evaluating metrics.

REFERENCES

[1] T.-Y. Kim and S.-B. Cho, "Web traffic anomaly detection using C-LSTM neural networks," *Expert Systems with Applications*, pp. 66–76, 2018.

[2] D. Kwon *et al.*, "An empirical study on network anomaly detection using convolutional neural networks," in *Proc. IEEE ICDCS 2018*, 2018.

[3] S. Naseer *et al.*, "Enhanced network anomaly detection based on deep neural networks," *IEEE Access*, pp. 48 231–48 246, 2018.

[4] E. Anthi *et al.*, "A Supervised Intrusion Detection System for Smart Home IoT Devices," *IEEE Internet Things J.*, vol. 6, no. 5, 2019.

[5] N. Shone *et al.*, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, pp. 41–50, 2018.

[6] T. Truong-Huu *et al.*, "An Empirical Study on Unsupervised Network Anomaly Detection using Generative Adversarial Networks," in *SPAI Workshop @ ACM AsiaCCS 2020*, 2020, pp. 20–29.

[7] M. Zhu, K. Ye, and C.-Z. Xu, "Network anomaly detection and identification based on deep learning methods," in *International Conference on Cloud Computing*, 2018, pp. 219–234.

[8] M. Latah and L. Toker, "An efficient flow-based multi-level hybrid intrusion detection system for software-defined networks," *CCF Transactions on Networking*, pp. 1–11, 2020.

[9] Y. Ren, L. Zhang, and P. N. Suganthan, "Ensemble classification and regression-recent developments, applications and future directions," *IEEE Comput. Intell. Mag.*, vol. 11, no. 1, pp. 41–53, 2016.

[10] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Military Communications and Information Systems Conference*, 2015.

[11] I. Sharafaldin *et al.*, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. ICISSP*, 2018.

[12] T. A. Tang *et al.*, "Deep learning approach for network intrusion detection in software defined networking," in *International Conference on Wireless Networks and Mobile Communications*, 2016, pp. 258–263.

[13] M. K. Prasath and B. Perumal, "A meta-heuristic bayesian network classification for intrusion detection," *International Journal of Network Management*, p. e2047, 2019.

[14] F. Salo *et al.*, "Clustering Enabled Classification using Ensemble Feature Selection for Intrusion Detection," in *IEEE ICNC 2019*, Honolulu, HI, USA, 2019, pp. 276–281.

[15] M. S. Elsayed *et al.*, "Detecting abnormal traffic in large-scale networks," *arXiv preprint arXiv:2008.05791*, 2020.

[16] Q. P. Nguyen *et al.*, "GEE: A gradient-based explainable variational autoencoder for network anomaly detection," in *IEEE CNS 2019*, 2019.

[17] T. Tang *et al.*, "A Deep Learning Approach Combining Auto-encoder with One-class SVM for DDoS Attack Detection in SDNs," in *International Conference on Communications and Networking*, 2019.

[18] S. Daneshgadeh Çakmakçi *et al.*, "Online DDoS attack detection using Mahalanobis distance and Kernel-based learning algorithm," *Journal of Network and Computer Applications*, vol. 168, p. 102756, 2020.

[19] Y. Mirsky *et al.*, "Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection," in *Proc. NDSS 2018*, 2018.

[20] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[21] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, pp. 1–18, 2015.

[22] I. Goodfellow *et al.*, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[23] M. Arjovsky and L. Bottou, "Towards Principled Methods for Training Generative Adversarial Networks," in *ICLR 2017*, France, Apr. 2017.

[24] T. Denouden *et al.*, "Improving Reconstruction Autoencoder Out-of-distribution Detection with Mahalanobis Distance," *CoRR*, vol. abs/1812.02765, 2018.

[25] T. Schlegl *et al.*, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," in *International Conference on Information Processing in Medical Imaging*, 2017.

[26] J. Liao *et al.*, "Ensemble Framework for Unsupervised Network Anomaly Detection: Appendix," 2021. [Online]. Available: https://gitlab.com/ttruonghuu/ieeecsr2021/blob/master/ieeecsr2021_appendix.pdf