An SDN controller-based framework for anomaly detection using a GAN ensemble algorithm

Dubem Ezeh, *Student Member, IEEE*, and J. de Oliveira, *Member, IEEE*

***Abstract*—Of recent, a handful of machine learning techniques have been proposed to handle the task of intrusion detection with algorithms taking charge; these algorithms learn, from traffic flow examples, to distinguish between benign and anomalous network events. In this paper, we explore the use of a Generative Adversarial Network (GAN) ensemble to detect anomalies in a Software-Defined Networking (SDN) environment using the Global Environment for Network Innovations (GENI) testbed over geographically separated instances. A controller-based framework is proposed, comprising several components across the detection chain. A bespoke dataset is generated, addressing three of the most popular contemporary network attacks and using an SDN perspective. Evaluation results show great potential for detecting a wide array of anomalies.**

***Index Terms*—Software-Defined Networking, network anomaly detection, GAN ensemble, machine learning, DDoS.**

I. INTRODUCTION

IN this paper, we propose the use of a GAN ensemble algorithm to detect network anomalies by understanding the distribution of normal network behaviour. in the context of Software Defined Networks (SDNs), where a controller executes the algorithm and can take action upon detecting an anomaly. Network anomaly detection has been extensive researched by academia and industry, and many methods have been proposed; from middleboxes (e.g., Intrusion Detection Systems) to other traditional rule-based tools (e.g., Snort). Machine learning has become the natural choice for anomaly detection, specially given the dynamic nature of the network attacks that are prominent today.

Ensemble learning is a machine learning technique that combines several base models in order to produce one optimal predictive model, usually by taking a (weighted) vote of the predictions of the base learners. Such models have been shown to perform better than each of their base learners [1], [2]. Although ensemble models have been proposed recently in the context of anomaly detection in traditional networks [3], our work uses a GAN ensemble approach in the context of an SDN environment to detect current and future threats. We envision a controller-based framework that comprises various co-operating systems representing each aspect of the detection chain. For validation, we used both existing popular datasets (CAIDA, UNSW-NB) and also created a new dataset by running experiments on the GENI research testbed [4], with instances deployed over geographically-separated zones. Our dataset contains three of the most popular contemporary network attacks. Evaluation results show an edge over other one-class as well as multi-class classification algorithms and a potential for detecting a wide array of new anomalies.

The authors are with the Department of Electrical and Computer Engineering, Drexel University, Philadelphia, PA, USA. (e-mail: jco24@drexel.edu)

DOI: 10.36244/ICJ.2023.2.5

The contributions of this paper include:

- Proposing an SDN Controller framework to deploy a GAN ensemble approach for network anomaly detection;
- Generation of a new dataset that addresses three of the most popular contemporary network attacks;
- Evaluation of the proposed framework using publicly available datasets as well as the newly created dataset;
- Evaluating the proposed framework's performance using a real testbed with geographically-separated nodes.

Throughout this paper, we will refer to an *abnormal* network flow as one that involves any of the intrusion exploits included in the datasets used. The rest of this paper is organized as follows. In Section II, we highlight previous contributions aimed at solving the anomaly detection problem and their shortcomings. We present in Section III, background information on the machine learning model utilized in this work. The major components of our proposed framework, together with the datasets used to validate proposal are detailed in Section IV. We present our experimental results in Section V, while Section VI contains our conclusions and thoughts on future work.

II. RELATED WORK

Anomaly detection has become a mainstay feature in computer networks as not all users utilize network resources for benign purposes. There are many types of network anomaly detection systems and this section will look at related work in these broad areas: signature-based and anomaly-based Network Intrusion Detection Systems (NIDS), machine-learning-based approaches, semi-supervised learning approaches, Generative Adversarial Networks (GANs) and ensembles.

In [5], NIDS have been classified, based on their detection approach as either signature-based (SIDS) or anomaly-based (AIDS) detection systems, and have been used extensively in many network deployments across the globe. The SIDS [6], [7], [8] are popular for efficiently detecting known attacks, but need to be updated constantly to handle new signatures, and thus fair poorly against unknown attacks. The AIDS approach [9], [10], [11] covers statistics-based, knowledge-based and machine learning-based solutions; these techniques require initial training but tend to fair better against new attacks. Of the three, machine learning-based solutions have witnessed immense proliferation and attention due to an improvement in computational capabilities, particularly in the area of Graphics Processing Units (GPUs), which speed up the training process of very complex machine learning models.

Machine learning algorithms can be grouped into four main categories, depending on how much information is known to the model a priori, during training: supervised, semi-supervised, unsupervised and reinforcement learning algorithms, with the supervised and unsupervised learning models being the two most common machine learning techniques used for anomaly detection in SDNs [12].

Supervised learning algorithms [13], [14] use labelled inputs and outputs for training, after which new inputs are used for testing. In [15], a CART-based Decision Tree algorithm was utilized for detecting anomalous traffic in the CICIDS2017 dataset; it was set up as a simulation for an SDN-oriented IoT network with detection being centralized at the controller.

Semi-supervised learning algorithms [16], [17], [18] use both labeled data (usually expensive and difficult to obtain) and unlabeled data (usually cheaper to obtain). These one-class classifiers are advantageous because they can, through different techniques, build boundaries around normal data so that anomalous events are easy to detect. The One-Class Support Vector Machine (OCSVM) algorithm was used in [19], [20] to detect novelty (another term for anomalies) in Internet of Things (IoT) devices; the memory and computation requirements of the OCSVM, which increase with the size of the training dataset, make it a less desired option for anomaly detection [21]. Isolation Forest [22] and Local Outlier Factor [23] are some other one-class classifiers that have also been used in the past for anomaly detection.

Unsupervised learning algorithms [24], [25], [26] are supplied inputs without labels, with the aim of grouping observations based on some metric of similarity between them. Clustering and data aggregation are the most popular unsupervised techniques in use today.

GANs are, originally, unsupervised, and deep-learning-based; they are used for generative modelling and involve two components: a generator and a discriminator, that work together to learn the patterns of a training dataset so that new examples can be generated which look very similar to examples in the training set. Because of their generative and discriminative properties, GANs have been employed to learn patterns of normal data so that they can easily detect anomalous data, thus adapting to anomaly detection solutions. MADGAN, a GAN variant, was used in [27] to detect early onset of brain anomalies, like Alzheimer's disease, mild cognitive impairment and brain metastases. Like MADGAN, several other proposals [28], [29] have been put forward to perform anomaly detection in the computer vision space.

Ensemble models are machine learning models that are built on the backs of two or more base models which may be of the same type, or may be heterogeneous, like in [30]. Ensembles offer the benefits of improved robustness and performance over their base models, and have been used to win different machine learning competitions [31]. In [32] an ensemble model built from multiple Efficient GAN-Based Anomaly Detection (EGBAD) models [33] was proposed; each generator in a base model was configured to learn from all the discriminators in the ensemble and vice-versa. The focus here was mostly on image datasets and computer vision applications.

In this paper, we draw inspiration from [32], [34] to propose the use of an ensemble algorithm, built upon homogeneous base EGBAD models, to properly detect today's SDN-oriented network anomalies, especially those of the highly pervasive DDoS type. We utilize two unique datasets for our experiments: a bespoke dataset, curated from an SDN topology built on the GENI research testbed [4] with instances deployed over geographically-separated zones (instagenis), and data from the UNSWB and CAIDA, all rolled into one. While other classifiers and network anomaly detectors require all classes of attack to be present during training, our GAN-based one-class classifier learns to represent only the normal space so as to detect when any attack is encountered since said attack would fall outside the boundary of normality; it also leverages the benefits of ensemble learning to yield a robust anomaly detector.

III. BACKGROUND

This section describes the ensemble-based algorithms used for detection. This model was first proposed in [33], primarily to help detect anomalies in the computer vision space and was therefore adapted to our purpose.

A. EGBAD model

A conventional GAN model is made from a Generator network, that tries to perfect the art of synthesizing data samples that fit the distribution of actual training samples; and a Discriminator network, that tries to discern whether a given sample is from the Generator or from the actual dataset. There are many variations of the GAN model ([35], [36], [37], [38]) with some more suited for computer vision exploits than others. In the EGBAD model [33], there is a choice between a conventional GAN and Bidirectional GAN (BiGAN) architecture. The latter performs better on tabular data type, to which the GENI-SDN and the CAIDA-UNSWB datasets belong. In the BiGAN architecture, the generator component is actually made up of an encoder and a decoder, such that the decoder output is what we use, together with real samples, to train our discriminator to distinguish between samples.

B. Ensemble

The idea behind an ensemble model is to build upon the strengths of other base machine learning models [39].

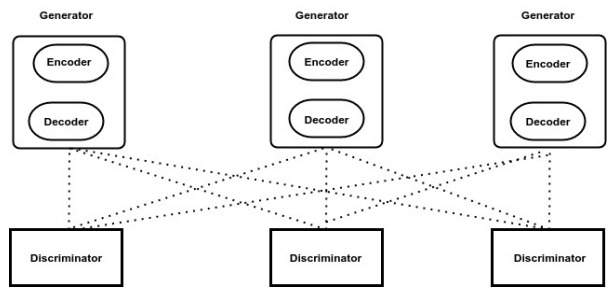

Fig. 1. A 3-generator, 3-discriminator EGBAD ensemble

Algorithm 1 EGBAD Ensemble from [32]

Input: Training set $X = \{x_i\}_{i=1}^{N}$

Output: Trained generators $\{G_e(\cdot;\phi_i), G_d(\cdot;\psi_i)\}_{i=1}^{I}$ and discriminators $\{D(\cdot;\gamma_j)\}_{j=1}^{J}$

Initialize parameters for $(\phi_i, \psi_i)_{i=1}^{I}$ and $(\gamma_j)_{j=1}^{J}$
$t \leftarrow 0$
while the objective not converge and $t <$ max_iter **do**
  Sample i from $\{1, \ldots, I\}$ and j from $\{1, \ldots, J\}$
  Sample a minibatch X^t from X
  Compute the adversarial loss $L_a^{(ij)}$
  Update $D(\cdot;\gamma_j) : \gamma_j \leftarrow \gamma_j + \nabla_{\gamma_j} L_a^{(ij)}$
  $\mathcal{L}^{(ij)} = \alpha_1 L_a^{(ij)} + \alpha_2 L_r^{(i)} + \alpha_3 L_d^{(ij)} + \alpha_4 L_e^{(i)}$
  Update $G_e(\cdot;\phi_i) : \phi_i \leftarrow \phi_i - \nabla_{\phi_i} \mathcal{L}^{(ij)}$
  Update $G_d(\cdot;\psi_i) : \psi_i \leftarrow \psi_i - \nabla_{\psi_i} \mathcal{L}^{(ij)}$
  $t \leftarrow t + 1$
end while

In this paper, there are three similar base models, each built using the BiGAN-based EGBAD algorithm for anomaly detection. Each generator network gets feedback from multiple discriminators while a discriminator receives samples from multiple generators, much like a meshed network (see Fig. 1). An anomaly score is taken from the average of anomaly scores computed for all generator-discriminator pairs; a higher anomaly score points to the sample falling outside the distribution of normality, while a lower anomaly score points to a sample that is normal. This interaction between the base models is what gives the ensemble the upper hand.

The anomaly detection problem is therefore solved in Algorithm 1, according to the implementation in [32]. Each generator, G_i, is characterized by an encoder, $G_e(\cdot;\phi_i)$ and a decoder, $G_d(\cdot;\psi_i)$. The encoder maps a given sample, x, to a latent vector, z, while the decoder computes a reconstruction, x' of the sample from z. The Discriminator on the other hand, takes a sample from the decoder and predicts the probability of the sample being from the actual sample set, X, instead of from the encoder-decoder.

IV. Proposed Framework and Experimental Setup

This section describes the implementation of our testbed, the proposed SDN controller-based framework for detection, and the datasets used to validate the efficacy of our solution.

A. GENI and SDN

The GENI [4] testbed is a very important tool for performing networking experiments and research. By harnessing Network Function Virtualization (NFV), GENI users are able to reserve and obtain geographically-separated compute resources within the United States, including layer 2 networks, protocols of their choice in layer 3 and above, and custom operating systems best suited to their research demands.

Fig. 2 shows the topology used to produce our GENI-SDN dataset. It comprises the following components:

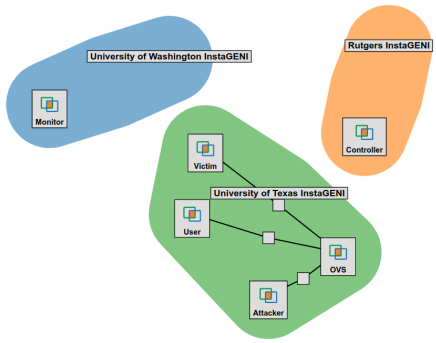

Fig. 2. Topology of experiment setup using GENI

1) SDN Controller: The controller oversees all traffic policies associated with the experiments and was deployed at Rutgers University (New Jersey). The Pox controller module (Fangtooth variant) was installed on top of a compute instance running the Ubuntu 18.04.6 LTS (Bionic Beaver) operating system.

2) OVS Switch and Associated Nodes: An Open vSwitch (OVS) was deployed at the University of Texas (Texas), together with three nodes: *Attacker*, *User* and *Victim*, all on the same local area network governed by the OVS. This switch was then linked to the remote controller for all traffic policies (done by setting the set-fail-mode to secure).

3) Monitor: This node was deployed at the University of Washington (Washington) to remotely listen for all traffic traversing the various ports of the OVS switch in Texas. To achieve this, a Generic Routing Encapsulation (GRE) tunnel was created between the monitor node and the OVS switch, and a Pox component was initiated from the controller to allow traffic duplication to said tunnel. The operating system of choice for this node and the other nodes on the LAN is Kali Linux as it is best suited for cybersecurity research.

We use three separate sites to demonstrate how, even when in different networks, packets can still be forwarded to a remote node for analyses/anomaly detection in "real-time". A simpler testbed could be setup with two sites, and with a remote controller acting as the monitoring node.

B. Proposed Framework

We show our proposed controller-based anomaly detection framework in Fig. 3 where multiple dataplane devices connect to the controller via the Southbound API, and are configured to forward traffic traces to the controller at regular intervals for analysis. The Northbound API is utilized to forward any observed anomalies to the network administrators in a timely fashion. The frequency of updates to the controller via the southbound interface would be a function of both Service Level Agreement (SLA) demands and prevailing network conditions. The controller would be the repository for the trained detection model, as well as copies of traffic traces that are analysed for potential threats.

Fig. 4 shows a basic process pipeline for anomaly detection within the controller. The controller serves as a repository for

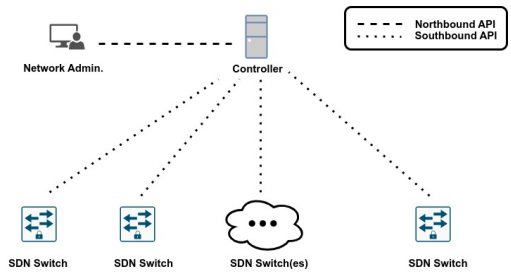

Fig. 3. Overview of controller-based anomaly detection framework

packet traces, obtained at intervals and containing information about events on the network. These traces are usually stored in *filename.pcap.gz* format. The controller would be responsible for pre-processing the traces into a machine-learning-readable format (usually *filename.csv*) with string, categorical and ordinal values properly encoded prior to training a model.

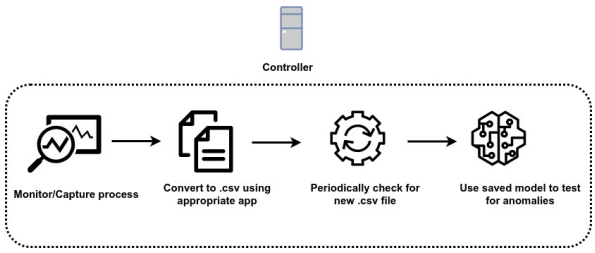

Fig. 4. Anomaly detection pipeline within controller

A detection model is trained (saved in *model.h5* format) and used to test incoming data for classification. Such a setup could process packet traces in short time intervals (e.g., five-minute intervals) to detect, and nip in the bud, nefarious actions within the network. From time to time, depending on the performance of the saved model, a re-training exercise might be scheduled to allow for a system that keeps up with changes in network behavior. In our experiments, a five-minute packet capture was used as a test case: a 28MB traffic trace was captured, pre-processed, encoded and tested. Roughly 1 second was used to extract information from the pcap file, encoding took 5 seconds to execute, and testing took 5.5 seconds with the computer used in the study, described in Table II. In all, a five-minute interval worked well in our testing, allowing for prompt testing of traces.

C. Datasets

1) GENI-SDN Dataset: The highly ranked KDD dataset was produced at a time when DDoS attacks and SDN technology had not become widespread. In order to properly validate the ensemble in view of current network trends, we produced a bespoke dataset by capturing network traffic at various intervals between the 28th of May, 2021, and the 17th of June 2021. To generate normal traffic, we used the *Iperf* application [40], running a server instance on the victim node and client instances on the other nodes directly connected to the OVS switch. Next, we describe how we created the other types of traffic.

Denial of Service (DoS): This type of attack attempts to flood a computer that provides a service in such a way that the server becomes unable to attend to the requests of legitimate clients. For the DoS attack, we run a simple HTTP server on the victim node. The attacker node uses the *Hping3* application [41] in flood-mode, in a bid to make the victim unavailable to legitimate users.

Distributed Denial of Service (DDoS): In DDoS, multiple attacking nodes combine their efforts to overwhelm a server and make it unavailable. DDoS traffic was generated using the Python-based open source tool, *Pyloris* [42], which works by spawning multiple threads (as specified by the user) to emulate the bots in a botnet that execute a DDoS exploit.

Enumeration: This type of attack is passive in that there is no action at this stage to overwhelm a server. The aim is to perform reconnaissance on a network and its associated resources to find out how many hosts are available on a network, the IP address layout, what services are listening on what open ports, what operating system is running on what host(s), etc. While it may be the most subtle of the attacks, information gleaned from this stage can be used for very harmful attacks subsequently. To implement this attack, we used the *Nmap* tool [43] to probe for different types of information about the network. It contains a total of 20 hours of network traffic traces stored on the remote monitor node. These traces were then pruned to extract a total of 32 features and 4 classes of network flow, including *Normal, DDoS, DoS* and *enumeration*. We limited the scope of our attacks since our experiments were run over an active testbed serving other researchers as well. The overall dataset contains over $1,000,000$ observations covering the aforementioned network classes and distributed in the fashion seen in Fig. IV-C1. A combination of *AWK* [44], *Bash* [45], *TCPdump* [46] and *ARGUS* [47] commands were used to generate the GENI dataset. The protocol distribution is 71.23% TCP, 28.59% UDP and 0.18% other.

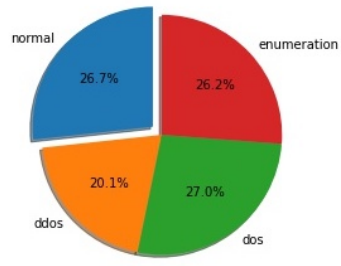

Fig. 5. Distribution of samples in the GENI-SDN dataset

2) CAIDA-UNSW Dataset: In order to properly validate our proposal, we utilized two datasets that covered diverse network anomalies - the UNSW-NB 15 Dataset [48] and the CAIDA DDoS Dataset [49].

UNSW-NB 15 Dataset: This dataset was created to assist researchers with credible data for network anomaly experiments. Initially in its raw form as hundreds of gigabytes of packet captures, the dataset was pruned to extract a total of 49 features and 9 classes of network flow, including *Fuzzers, Analysis, Backdoors, DoS Exploits, Generic, Reconnaissance, Shellcode*

and *Worms*. The overall dataset contains about 2,540,048 observations covering the aforementioned network classes. DDoS anomalies were not included in the experiments that yielded this dataset.

CAIDA Dataset: The CAIDA DDoS dataset contains tens of gigabytes of compressed packet captures which together cover approximately one hour of anonymized network traffic traces from a DDoS attack. All non-attack traffic have been removed from the get-go and traces anonymized using CryptoPAn prefix-preserving anonymization [49].

After obtaining the relevant packet traces from the two datasets and processing them to usable formats, the two were then merged to create a new dataset with a total of 10 network classes - all of the earlier 9 and an additional 'DDoS' class. This new dataset helps guarantee that a working model can at least detect normal and anomalous (including of the DDoS type) network flows to a great degree of accuracy.

Feature Selection: Many models, especially those based on regression slopes and intercepts, will estimate parameters for every included feature. Because of this, the presence of non-informative features can add uncertainty to predictions and reduce the overall effectiveness of the model [50]. Issues relating to over-fitting and computational complexity can become accentuated when redundant dataset features are used to train models. With these in mind, and considering the sizes of the datasets, we had to employ three feature-selection algorithms to remove non-informative or redundant predictors from the dataset, namely the *ANOVA, Chi-Squared* and *Mutual Information* algorithms. These were carefully selected since they work well with mixed data types (numerical, categorical and ordinal) and are better suited for classification modeling problems. The features eventually picked for the experiments are seen in Table I.

TABLE I
SELECTED FEATURES AND THEIR DESCRIPTIONS

Feature	Description
Proto	The upper protocol used in the transaction
State	The state and its dependent protocol
Dur	Record total duration
SrcBytes	Source to destination transaction bytes
DstBytes	Destination to source transaction bytes
SrcLoss	Source packets retransmitted or dropped
DstLoss	Destination packets retransmitted or dropped
SrcLoad	Source bits per second
DstLoad	Destination bits per second
SrcPkts	Source-to-destination packet count
DstPkts	Destination-to-source packet count
sMeanPktSz	Mean of the flow packet size transmitted by the source
dMeanPktSz	Mean of the flow packet size transmitted by the destination
TcpRtt	Tcp connection setup RTT, sum of synack and ackdat
SynAck	TCP connection setup time, the time between the SYN and the SYN_ACK packets.
AckDat	TCP connection setup time, the time between the SYN_ACK and the ACK packets

V. EXPERIMENTAL RESULTS

Our experimental results will be in two parts, addressing the two datasets used. Details of the environmental setup for the pre-processing and model training/testing stages are provided in Table II. The compute resources used for training and testing the algorithms mentioned in this paper were supplied by Chameleon Cloud [51], a configurable experimental environment for large-scale edge to cloud research.

The hyperparameters in Table III were used for training the ensemble model. To build an accurate but efficient ensemble model, we tried various combinations of generator and discriminator values and plotted the Receiver Operating Characteristic (ROC) curve for each scenario; it is a plot of the false positive rate $\frac{FalsePositives}{FalsePositives+TrueNegatives}$, or false alarm rate, against the true positive rate $\frac{TruePositives}{TruePositives+FalseNegatives}$ (or hit rate). The motive behind training the model on only benign observations is for such a system to be able to detect a wide range of network anomalies, including those not present in the datasets, as shown in [52].

TABLE II
ENVIRONMENTAL DETAILS

Number of flows	498136
CPU	AMD EPYC 7763 3.1GHz * 64 cores
Memory (RAM)	256GB
GPU	AMD Instinct MI100 (32GB) * 2

TABLE III
HYPERPARAMETERS USED FOR ENSEMBLE TRAINING

Parameter	Value
GAN type	BiGAN
Learning rate	0.00002
Number of GPUs	2
Latent dimension (encoder)	32
Training set	80% of all normal samples in the dataset
Number of generators	5
Number of Discriminators	5

As depicted in Fig. 6, we noticed no significant increase in performance beyond 5 generators and 5 discriminators, so we use this number for all our experiments.

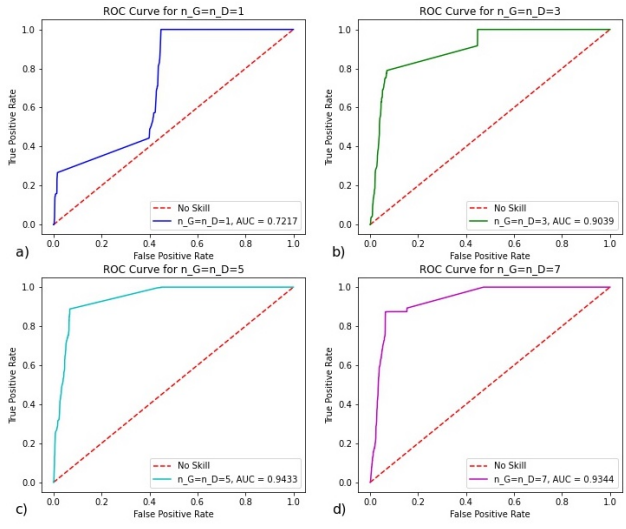

Fig. 6. ROC curve plots for ensembles built using: (a) 1 generator, 1 discriminator (b) 3 generators and 3 discriminators (c) 5 generators and 5 discriminators (d) 7 generators and 7 discriminators (based on the CAIDA-UNSWB dataset)

A. Benchmark Models and Performance Metrics

The Scikit-Learn [53] and Pytorch libraries played an integral role in the building and testing of the anomaly detector. To evaluate like-for-like algorithms, we restricted our comparisons to the following one-class models:

- Local outlier Factor (LoF): This algorithm works by computing the local density deviation of a given sample with respect to its neighbors.
- Isolation Forest (IsoF): This algorithm is an ensemble model that isolates anomalies by using binary trees instead of trying to profile normal samples.
- Minimum Covariance Determinant (MCD): This algorithm works by finding those samples in a set whose covariance matrix has the lowest determinant.

The performance metrics described in Table IV were used in our evaluation.

TABLE IV
METRICS FOR EVALUATING MODEL PERFORMANCE

Metric	Description
Precision	The fraction of the total amount of relevant instances that were actually retrieved
Recall	The fraction of relevant instances among the retrieved instances
F-1 score	The harmonic mean of precision and recall
AUROC	A measure of the model's ability to discriminate between normal and anomalous traffic
Accuracy	the ratio of correctly predicted data points to the number of all the data points

TABLE V
DETECTION PERFORMANCE FOR DIFFERENT MODELS ON THE GENI-SDN DATASET IN A TWO-CLASS SCENARIO

	accuracy	precision	recall	f1-score
EGBAD ensemble	**0.995**	**0.999**	**0.994**	**0.997**
LoF	0.989	0.998	0.989	0.994
IsoF	0.834	0.997	0.792	0.884
MCD	0.756	0.814	0.899	0.855

TABLE VI
AUROC RESULTS FOR DIFFERENT MODELS ON THE GENI-SDN DATASET USING RESPECTIVE ATTACK TYPES

	DDoS	DoS	Enumeration	Average
EGBAD ensemble	**0.999**	**0.994**	**0.999**	**0.997**
LoF	0.989	0.998	0.989	0.992
IsoF	0.834	0.997	0.792	0.874
MCD	0.756	0.814	0.899	0.823

B. GENI-SDN Results

Five BiGAN-based EGBAD models were used to train the ensemble model. Using a batch size of 1024 to fetch samples for training and a size of 32 for the latent vector, our ensemble model was able to perform remarkably when compared to other classifiers, as seen in Table V; recall values and precision for the ensemble outperform the other one-class learning algorithms used in our experiments. The plots in Fig. 7 show the AUROC performance for the various one-class models under observation and again, we observe that the EGBAD plot shows the largest area under the curve, implying a large percentage of True Positives and a low percentage of False Positives.

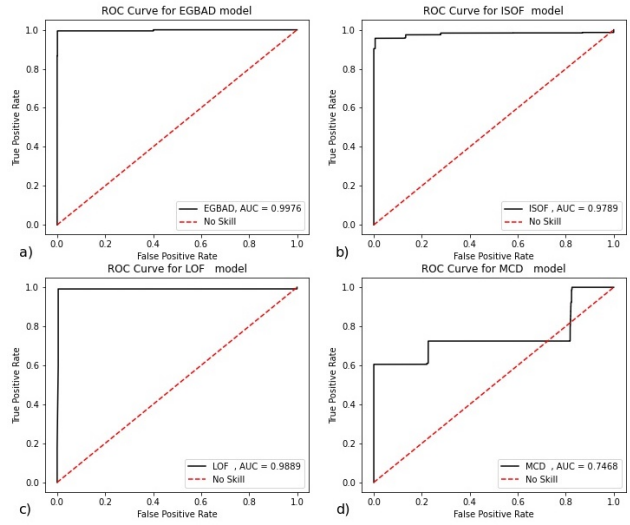

Fig. 7. ROC curve plots based for the: (a) EGBAD (b) ISOF (c) LOF (d) MCD one-class anomaly detectors based on the GENI dataset

In Table VI, we compare the AUROC results for the different models under observation using the respective attack types; we notice that the EGBAD ensemble model performs better per instance and collectively on average.

The two-class confusion matrix shown in Fig. 8 corroborates the 99.5% classification accuracy of our model, with only 0.030% of the overall testset (or 0.0015% of the anomalous part of the testset) being incorrectly classified as benign.

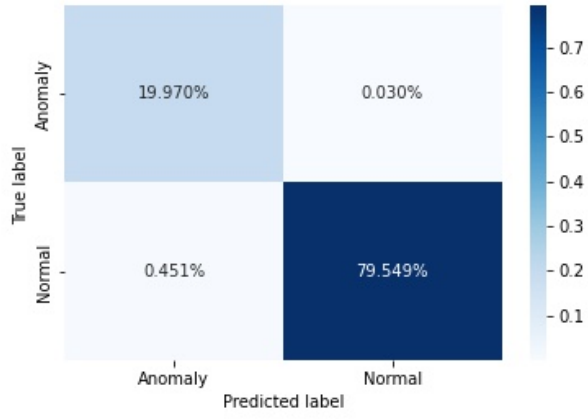

Fig. 8. Two-class confusion matrix based on GENI-SDN dataset

C. CAIDA-UNSWB Results

Like in the GENI case, a 5*x*5 ensemble model was used here, comprising five generators and five discriminators, all learning from each other. The composite dataset used here had more observation samples and overall attack classes, providing more variety with which to benchmark our proposed solution. We see in Table VII that our EGBAD ensemble records a 90% score in detection accuracy, with the next best model scoring

a distant 83% in detection accuracy. The same is observed for the other metrics except the recall score in which the Local Outlier Factor model performs marginally better. A tight race is also observed in the AUROC performance (see Fig. 9) where the Isolation Forest model, also an ensemble, wins it at 94%, but only narrowly.

TABLE VII
DETECTION PERFORMANCE FOR DIFFERENT MODELS ON THE CAIDA-UNSWB DATASET

	accuracy	precision	recall	f1-score
EGBAD ensemble	**0.905**	**0.973**	0.973	**0.944**
LoF	0.806	0.808	**0.993**	0.891
IsoF	0.833	0.833	0.989	0.904
MCD	0.738	0.798	0.900	0.846

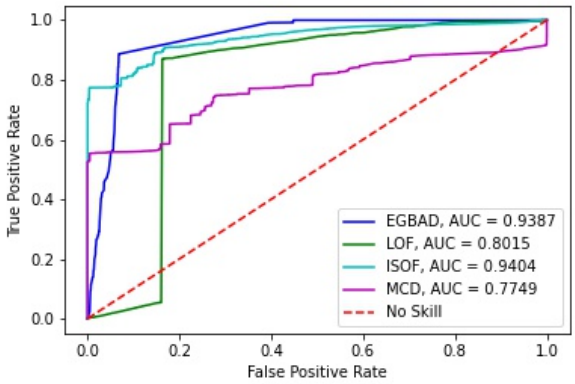

Fig. 9. AUROC results for the different models based on the CAIDA-UNSWB dataset

VI. CONCLUSIONS AND FUTURE WORK

To be able to properly detect anomalies in software-defined networks, we proposed a controller-based detection framework, including an ensemble learning technique, built on five diverse base EGBAD learners. We curated a bespoke SDN-based dataset and performed experiments in various anomaly detection scenarios, entailing a scenario with four classes (normal, DDoS, DoS and enumeration), as well as a binary-class scenario (normal and anomalous). The ensemble model showed consistently better detection performance numbers than its base learners, as well as when compared against other established one-class anomaly-detection algorithms. Similar behaviors were observed when the EGBAD model was applied to the CAIDA-UNSWB dataset even though it more observations and more attack classes than the GENI-SDN dataset. In terms of future work, we are working on a more robust dataset with even more attack classes; this would allow for an ensemble model that can accurately identify even more types of network anomalies when they occur. This presents the opportunity to create datasets from a purely SDN perspective. We also plan to test a multiple-controller-based framework that exploits a distributed approach to anomaly detection with the added benefits of resilience and redundancy.In summary, we observed an average detection accuracy above 90% over the two datasets when collapsed to just two classes (normal and abnormal). This demonstrates the potential for GAN-based algorithms to be used for network anomaly detection after training a robust discriminator network on what normal network events look like. Because this is an offline detector, training is done once, and without any impact on active network functions, as is testing with new data. The datasets and associated codes will be made available on our Github page accordingly.

ACKNOWLEDGMENT

Results presented in this paper were obtained using the Chameleon and GENI testbeds supported by the National Science Foundation.

REFERENCES

[1] D. Ezeh, "On packet classification using a decision-tree ensemble," in *Proceedings of the Student Workshop*, ser. CoNEXT'20. New York, NY, USA: Association for Computing Machinery, 2020, p. 17–18. [Online]. Available: DOI: 10.1145/3426746.3434054

[2] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits and Systems Magazine*, vol. 6, no. 3, pp. 21–45, 2006. [Online]. Available: DOI: 10.1109/MCAS.2006.1688199

[3] A. Mahfouz, A. Abuhussein, D. Venugopal, and S. Shiva, "Ensemble classifiers for network intrusion detection using a novel network attack dataset," *Future Internet*, vol. 12, no. 11, 2020. [Online]. Available: https://www.mdpi.com/1999-5903/12/11/180

[4] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskar, "Geni: A federated testbed for innovative network experiments," *Computer Networks*, vol. 61, pp. 5–23, 2014, special issue on Future Internet Testbeds - Part I. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128613004507

[5] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecur.*, vol. 2, p. 20, 2019.

[6] A. Altaher, "An improved android malware detection scheme based on an evolving hybrid neuro-fuzzy classifier (ehnfc) and permission-based features," *Neural Computing and Applications*, vol. 28, pp. 4147–4157, 2016.

[7] A. H. Hamamoto, L. F. Carvalho, L. D. H. Sampaio, T. Abrão, and M. L. Proença, "Network anomaly detection system using genetic algorithm and fuzzy logic," *Expert Systems with Applications*, vol. 92, pp. 390–402, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741741730619X

[8] S. Elhag, A. Fernández, S. Alshomrani, and F. Herrera, "Evolutionary fuzzy systems: A case study for intrusion detection systems," *Studies in Computational Intelligence*, 2018.

[9] W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "Cann: An intrusion detection system based on combining cluster centers and nearest neighbors," *Knowl. Based Syst.*, vol. 78, pp. 13–21, 2015.

[10] A. P. Meshram and C. Haas, "Anomaly detection in industrial networks using machine learning: A roadmap," in *ML4CPS*, 2016.

[11] S. Elhag, A. Fernández, A. Bawakid, S. Alshomrani, and F. Herrera, "On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on intrusion detection systems," *Expert Syst. Appl.*, vol. 42, pp. 193–202, 2015.

[12] J. feng Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. meng Wang, and Y. Liu, "A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, pp. 393–430, 2019.

[13] S. Alhaidari and M. Zohdy, "Network anomaly detection using two-dimensional hidden markov model based viterbi algorithm," in *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*, 2019, pp. 17–18.

[14] T. Kim, S. C. Suh, H. Kim, J. Kim, and J. Kim, "An encoding technique for cnn-based network anomaly detection," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 2960–2965.

[15] P. Amangele, M. J. Reed, M. Al-Naday, N. Thomos, and M. Nowak, "Hierarchical machine learning for iot anomaly detection in sdn," in *2019 International Conference on Information Technologies (InfoTech)*, 2019, pp. 1–4.

[16] A. Kumagai, T. Iwata, and Y. Fujiwara, "Semi-supervised anomaly detection on attributed graphs," *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2021.

[17] L. Ruff, R. A. Vandermeulen, N. Görnitz, A. Binder, E. Müller, K. Müller, and M. Kloft, "Deep semi-supervised anomaly detection," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. [Online]. Available: https://openreview.net/forum?id=HkgH0TEYwH

[18] M. E. Villa-Pérez, M. A. Álvarez Carmona, O. Loyola-González, M. A. Medina-Pérez, J. C. Velazco-Rossell, and K.-K. R. Choo, "Semi-supervised anomaly detection algorithms: A comparative summary and future research directions," *Know.-Based Syst.*, vol. 218, no. C, apr 2021. [Online]. Available: DOI: 10.1016/j.knosys.2021.106878

[19] A. Shorman, H. Faris, and I. Aljarah, "Unsupervised intelligent system based on one class support vector machine and grey wolf optimization for iot botnet detection," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, 07 2020.

[20] I. Razzak, K. Zafar, M. Imran, and G. Xu, "Randomized nonlinear one-class support vector machines with bounded loss function to detect of outliers for large scale iot data," *Future Generation Computer Systems*, vol. 112, 06 2020.

[21] K. Yang, S. Kpotufe, and N. Feamster, "An efficient one-class svm for anomaly detection in the internet of things," *ArXiv*, vol. abs/2104.11146, 2021.

[22] H. Xiang, J. Wang, K. Ramamohanarao, Z. Salcic, W. Dou, and X. Zhang, "Isolation forest based anomaly detection framework on non-iid data," *IEEE Intelligent Systems*, vol. 36, no. 3, pp. 31–40, 2021.

[23] N. Paulauskas and F. Bagdonas, "Local outlier factor use for the network flow anomaly detection," *Security and Communication Networks*, vol. 8, 08 2015.

[24] A. Vikram and Mohana, "Anomaly detection in network traffic using unsupervised machine learning approach," *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, pp. 476–479, 2020.

[25] T. Li, Z. Wang, S. Liu, and W.-Y. Lin, "Deep unsupervised anomaly detection," in *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2021, pp. 3635–3644.

[26] A. Allahdadi and R. Morla, "Anomaly detection and modeling in 802.11 wireless networks," *J. Netw. Syst. Manag.*, vol. 27, no. 1, pp. 3–38, 2019. [Online]. Available: DOI: 10.1007/s10922-018-9455-2

[27] C. Han, L. Rundo, K. Murao, T. Noguchi, Y. Shimahara, Z. Milacski, S. Koshino, E. Sala, H. Nakayama, and S. Satoh, "Madgan: unsupervised medical anomaly detection gan using multiple adjacent brain mri slice reconstruction," *BMC Bioinformatics*, vol. 22, p. 31, 04 2021.

[28] T. Schlegl, P. Seeböck, S. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," 03 2017, pp. 146–157.

[29] Q. Yang and X. Li, "Bigan: Lncrna-disease association prediction based on bidirectional generative adversarial network," *BMC Bioinformatics*, vol. 22, no. 1, p. 357, Jun 2021. [Online]. Available: DOI: 10.1186/s12859-021-04273-7

[30] Y. Zhong, W. Chen, Z. Wang, Y. Chen, K. Wang, Y. Li, X. Yin, X. Shi, J. Yang, and K. Li, "Helad: A novel network anomaly detection model based on heterogeneous ensemble learning," *Comput. Netw.*, vol. 169, no. C, mar 2020. [Online]. Available: DOI: 10.1016/j.comnet.2019.107049

[31] G. Seni and I. John F. Elder, "Ensemble methods in data mining: Improving accuracy through combining predictions," in *Ensemble Methods in Data Mining*, 2010.

[32] X. Han, X. Chen, and L.-P. Liu, "Gan ensemble for anomaly detection," 2020.

[33] H. Zenati, C. S. Foo, B. Lecouat, G. Manek, and V. R. Chandrasekhar, "Efficient gan-based anomaly detection," 2019.

[34] P. Lin, K. Ye, and C.-Z. Xu, *Dynamic Network Anomaly Detection System by Using Deep Learning Techniques*, 06 2019, pp. 161–176.

[35] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," 2018. [Online]. Available: https://arxiv.org/abs/1809.11096

[36] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014. [Online]. Available: https://arxiv.org/abs/1411.1784

[37] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," 2016. [Online]. Available: https://arxiv.org/abs/1612.03242

[38] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," 2017. [Online]. Available: https://arxiv.org/abs/1701.07875

[39] A. Geron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: concepts, tools, and techniques to build intelligent systems*, second edition. ed. Sebastopol, CA: O'Reilly Media, 2019.

[40] Iperf, "Iperf, https://iperf.fr/," 2021, last Accessed: Nov., 1.

[41] Hping, "Hping, http://hping.org/," 2021, last Accessed: Nov., 1.

[42] Pyloris, "Pyloris, https://github.com/darkerego/pyloris," 2021, last Accessed: Nov., 1.

[43] Nmap, "Nmap, https://nmap.org," 2021, last Accessed: Nov., 1.

[44] AWK, "Awk manual, https://www.gnu.org/software/gawk/manual/gawk.html," 2021, last Accessed: Nov., 1.

[45] BASH, "Bourne again shell, https://www.gnu.org/software/bash/," 2021, last Accessed: Nov., 1.

[46] TCPdump, "Tcpdump, https://tcpdump.org/," 2021, last Accessed: Nov., 1.

[47] ARGUS, "Argus, https://openargus.org/," 2021, last Accessed: Nov., 1.

[48] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6. [Online]. Available: DOI: 10.1109/MilCIS.2015.7348942

[49] CAIDA, "Caida ddos dataset, https://www.caida.org/data/passive/ddos-20070804_dataset.xml," 2021, last Accessed: Nov., 1.

[50] M. Kuhn, *Applied predictive modeling*. New York: Springer, 2013. [Online]. Available: DOI: 10.1007/978-1-4614-6849-3

[51] K. Keahey, J. Anderson, Z. Zhen, P. Riteau, P. Ruth, D. Stanzione, M. Cevik, J. Colleran, H. S. Gunawi, C. Hammock, J. Mambretti, A. Barnes, F. Halbach, A. Rocha, and J. Stubbs, "Lessons learned from the chameleon testbed," in *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC '20)*. USENIX Association, July 2020.

[52] D. Ezeh, "Exploring advanced machine learning solutions for traffic classification, anomaly detection, and adaptive data transmission in software defined networks," Ph.D. dissertation, Drexel University, 2023.

[53] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. [Online]. Available: DOI: 10.48550/arXiv.1201.0490

Dubem A. Ezeh received his B.Eng degree in Electrical Engineering from Ahmadu Bello University, Nigeria, in 2009, his MS degree in Telecommunications Engineering from Staffordsdshire University, UK, in 2012, and is currently pursuing a PhD degree in Computer Engineering from Drexel University, USA. His research interests include Software-Defined Networks, Traffic Engineering, Network Security and Machine Learning Applications in Computer Networks.

Jaudelice de Oliveira is an Associate Professor in the Department of Electrical and Computer Engineering (ECE) at Drexel University. She earned her B.S.E.E. degree from Federal University of Ceara, Ceara, Brazil, M.S.E.E. from the State University of Campinas, São Paulo, Brazil, and her Ph.D. degree in ECE from the Georgia Institute of Technology. Currently, her research interests include Software Defined Networks, Smart Grid and IoT. She is a Member of the IEEE and Senior Member of the ACM.