IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 8 August 2022, accepted 1 September 2022, date of publication 8 September 2022, date of current version 19 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3205337

RESEARCH ARTICLE

Effect of Balancing Data Using Synthetic Data on the Performance of Machine Learning Classifiers for Intrusion Detection in Computer Networks

AYESHA SIDDIQUA DINA, A. B. SIDDIQUE, AND D. MANIVANNAN, (Senior Member, IEEE)
Department of Computer Science, University of Kentucky, Lexington, KY 40508, USA

Corresponding author: D. Manivannan (mani@cs.uky.edu)

This work was supported by the Department of Computer Science, University of Kentucky, Lexington, KY, USA.

ABSTRACT Attacks on computer networks have increased significantly in recent days, due in part to the availability of sophisticated tools for launching such attacks as well as the thriving underground cyber-crime economy to support it. Over the past several years, researchers in academia and industry used machine learning (ML) techniques to design and implement Intrusion Detection Systems (IDSes) for computer networks. Many of these researchers used datasets collected by various organizations to train ML classifiers for detecting intrusions. In many of the datasets used in training ML classifiers in such systems, data are imbalanced (i.e., not all classes had equal number of samples). ML classifiers trained with such imbalanced datasets may produce unsatisfactory results. Traditionally, researchers used over-sampling and under-sampling for balancing data in datasets to overcome this problem. In this work, in addition to random over-sampling, we also used a synthetic data generation method, called Conditional Generative Adversarial Network (CTGAN), to balance data and study their effect on the performance of various widely used ML classifiers. To the best of our knowledge, no one else has used CTGAN to generate synthetic samples to balance intrusion detection datasets. Based on extensive experiments using widely used datasets NSL-KDD and UNSW-NB15, we found that training ML classifiers on datasets balanced with synthetic samples generated by CTGAN increased their prediction accuracy by up to 8% and improved their MCC score by up to 13%, compared to training the same ML classifiers over imbalanced datasets. We also show that this approach consistently performs better than some of the recently proposed state-of-the-art IDSes on both datasets. Our experiments also demonstrate that the accuracy of some ML classifiers trained over datasets balanced with random over-sampling decline compared to the same ML classifiers trained over original imbalanced dataset.

INDEX TERMS Cyber security, conditional generative adversarial network (CTGAN), data imbalance problem, intrusion detection, machine learning, over-sampling, under-sampling.

I. INTRODUCTION

There has been significant increase in the number of intrusions into computer networks over the past few years due in part to the availability of sophisticated tools to launch such attacks as well as a thriving underground economy to support such attacks [19]. According to a 2017 report [35], data breaches cost an average of $141 per record. It is estimated that 60% of small businesses that suffer a data breach will cease operations within six months. Symantec's Internet Security Threat Report for 2017 indicated that the number and intensity of attacks were significantly higher than those in previous years [30]. Traditional tools such as firewalls can not cope with these sophisticated attacks.

To prevent/detect network intrusions, hardware and software tools can be installed to continuously monitor the network. James Anderson published a report on the need for detecting network intrusions in computer systems [3] in 1972 [6]. Since then, several intrusion detection systems (IDSes) have been proposed and implemented. These systems

The associate editor coordinating the review of this manuscript and approving it for publication was Antonio J. R. Neves.

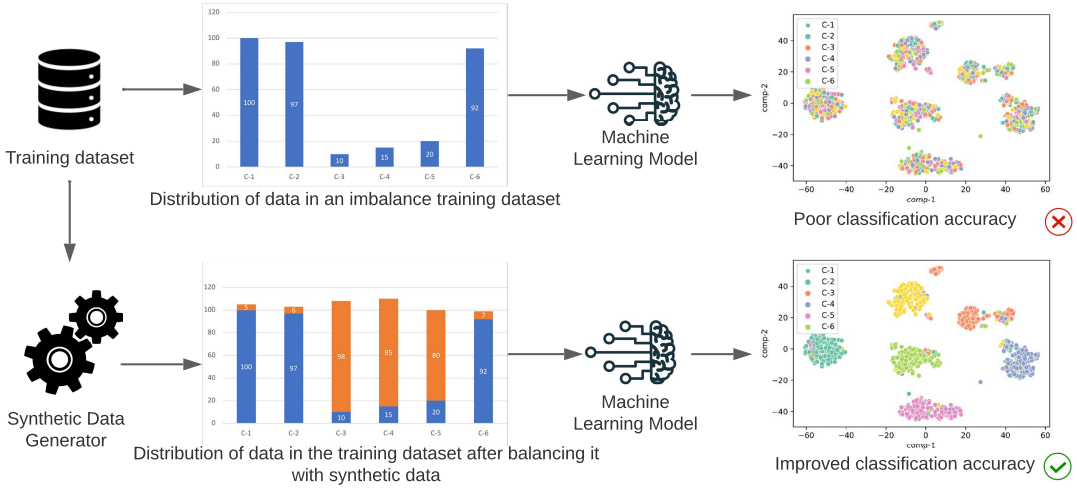


FIGURE 1. ML classifiers trained on imbalanced datasets show poor performance – no matter what type of machine learning classifier is employed. Augmenting minority classes in the training dataset with synthetic data can help in overcoming the data imbalance issue and improve the performance of ML classifiers.

can be further classified as host-based, network-based, and hybrid [59]. System architectures for intrusion detection can be centralized, distributed, or hybrid, based on how intrusion/attack events are collected, processed, and acted upon. Certain approaches are superior to others based on factors such as cost, performance, and other metrics. These systems can be further classified based on the techniques used for intrusion detection – signature-based or anomaly-based. A signature-based IDS detects attacks based on the signatures of previously known attacks. These IDSes cannot detect zero-day attacks. In contrast, anomaly-based IDSes are capable of detecting zero-day attacks by modeling users' behaviors. In the training phase of an anomaly-based approach, legitimate users' behaviors are first collected and analyzed in order to build a model of legitimate users' behavior. The model is then used to determine whether the current observed behavior is that of legitimate user or not. Some methods used for such classification are [59]: **Statistical approach**: classification is based on univariate, multivariate, or time-series models. **Knowledge based approach**: expert system is used to model legitimate behavior according to a set of rules. **Machine learning based approach**: automatically classified based on some clustering algorithms. However, anomaly-based IDSes often generate more false positives and signature-based IDSes generally generate more false negatives.

ML based IDSes have been extensively studied in the literature. For example, following ML based approaches have been tested by various researchers for intrusion detection: Artificial Neural Networks, Association Rules and Fuzzy Association Rules, Bayesian Networks, Clustering, Decision Trees, Evolutionary Computation, Hidden Markov Models, Inductive Learning, Naïve Bayes, Sequential Pattern Mining, and Support Vector Machine [7], [8], [19], [53]. In many of the datasets used for training ML classifiers in such studies, datasets are not balanced. That is, number of samples in one class surpasses the number of samples in another class [1]. The classes that have a large number of samples are called majority classes, while the classes that have a small number of samples are called minority classes. The ratio of the number of samples in a minority class to the number of samples in a majority class may be as small as 1:100, or as large as 1:1000, or even larger [12]. Figure 1 illustrates how imbalance in datasets can affect the performance of ML classifiers. Many of the researchers (i) ignored this problem, or (ii) balanced the training dataset using over-sampling (randomly replicating samples in minority classes) or under-sampling (randomly eliminating samples in majority classes) techniques. Over-sampling and under-sampling help in balancing data. However, since the new samples added under over-sampling are exact copies of the original samples, it may lead to overfitting. Similarly, since random samples are eliminated from majority classes in under-sampling, the dataset may become too simple to build an effective model, resulting in underfitting problem. In general, an overfit model has low bias and high variance, while an underfit model has high bias and low variance.

In this paper, we studied the effect of balancing training datasets on the performance of various ML classifiers; we used (i) the most commonly used random over-sampling method, and (ii) synthetic data generated using the Conditional Generative Adversarial Network (CTGAN) [66] for balancing training datasets. We compared the performance of various ML classifiers (i) after training them on original imbalanced data, (ii) after training them on the original data balanced with over-sampling, (ii) after training them on the original data, balanced with synthetic samples generated using CTGAN [66]. CTGAN exploits a conditional generative adversarial network, learns from input data (i.e., both discrete and continuous features), and generates high-fidelity synthetic samples. It is important to emphasize that the new

synthetic samples generated by CTGAN are not copies of the samples in the original dataset but look-alike instances. To the best of our knowledge, this is the first time CTGAN has been used to generate synthetic data for balancing data related to intrusion detection, even though it has been used for image data.

We used the datasets NSL-KDD (Network Socket Layer-Knowledge Discovery in Database) [62] and UNSW-NB15 [41] in our experiments; these are some of the widely used datasets for studying intrusion detection in computer networks. We evaluated the performance of the following ML classifiers through extensive experiments: Decision Tree (DT), Support Vector Machine (SVM), Random Forest (RF), Naive Bayes (NB), Feed Forward Network (FNN), Long Short Term Memory (LSTM), and Convolutional Neural Network (CNN). Additionally, we compared our proposed approach against some of state-of-the-art IDSes, namely, CNN-BiLSTM [58] and PB-DID [71]; experimental results on both datasets demonstrate that our method performs better than CNN-BiLSTM and PB-DID.

Our focus is on multi-class classification rather than binary classification – a more challenging problem setup. Multi-Class classification makes it possible to evaluate the performance of various classifiers with respect to different types of intrusions. Our experimental results show that on the NSL-KDD dataset, with training data balanced with synthetic data generated using CTGAN, prediction accuracy of some of the ML classifiers increased by as much as 8% and their MCC score improved by as much as 13%. **Following is a summary of our contribution in this paper:**

- We show that, using improved algorithms for generating synthetic data for balancing the datasets used for training ML classifiers, could improve the performance of ML classifiers in detecting intrusions in computer networks more accurately.
- We used CTGAN to generate synthetic samples to balance the training datasets in NSL-KDD and UNSW-NB15. To the best of our knowledge, this is the first time CTGAN has been used to generate synthetic data for balancing data associated with intrusion detection. It is noteworthy to mention that CTGAN has been used for image augmentation in the literature.
- We evaluated the performance of several widely used ML classifiers. Our evaluations show that ML classifiers trained on training datasets of NSL-KDD and UNSW-NB15, balanced with synthetic samples generated by CTGAN, performed better compared to their performance when trained on (i) the original imbalanced training datasets, and (ii) the original training datasets balanced using random over-sampling. Moreover, we also show that the proposed approach performs better than some of the state-of-the-art IDSes CNN-BiLSTM [58] and PB-DID [71].

The rest of the paper is organized as follows. In Section II, we present our proposed approach as well as discuss the various ML classifiers used for evaluation. In Section III, we present our experimental setup and results. In Section IV, we discuss related works, and Section V concludes the paper.

II. PROPOSED APPROACH AND CLASSIFICATION METHODS USED

In this section, we discuss how we model data, preprocess data, and use CTGAN to generate synthetic data to balance data in the training datasets of NSL-KDD and UNSW-NB15. Then, we discuss various ML classifiers that we used in our experimental evaluation.

A. MODELING DATA

We modeled the input data as a two-dimensional matrix $X = (x_1, x_2, x_3, \cdots, x_N)$, where $x_i \in \mathbb{R}^D$ $(1 \leq i \leq N)$ is a vector with D dimensional network feature space. Each x_i $(1 \leq i \leq N)$ is associated with a label y_i and $y_i \in \{1, \cdots, L\}$. In our case, N is the number of samples in the dataset and L is the number of distinct attack categories. The feature vectors are mapped to labels by a function $Y = f(x)$ that is unknown. As part of supervised learning, the training dataset was used to obtain an estimate of f. This estimated function is referred to as $\hat{f}(x)$. The goal is to make $\hat{f}(x)$ as close as possible to $f(x)$.

B. PREPROCESSING OF DATA

We transformed all the categorical variables into numerical variables during the preprocessing step. For this transformation, we used label encoding [24], [40]. During this process, each label of a categorical feature is assigned a unique numerical value in alphabetical order. Imagine a two-dimensional matrix X containing column C_i. Column C_i contains four categorical labels – *tcp*, *smtp*, *ftp* and *http*. These are different types of protocols. Our label encoding assigns the values 1, 2, 3, and 4 to the labels *ftp*, *http*, *smtp*, and *tcp*, respectively, in alphabetical order.

In the next step, we normalized the input data. In this study, we used L_2 normalization or Euclidean normalization [65]. We used the same input matrix X and i^{th} feature C_i. The feature C_i is normalized using Equation 1.

$$C_i = \frac{C_i}{||C_i||_2} \tag{1}$$

where

$$||C_i||_2 = \sqrt{\sum_{k=1}^{K} c_{k_i}^2},$$

and $C_i = [c_{1_i}, c_{2_i}, c_{3_i}, \cdots, c_{K_i}]$, a vector of length K. $||C_i||_2$ is the L_2 norm of the vector C_i.

C. SYNTHETIC DATA GENERATION USING CTGAN TO BALANCE DATA

Data imbalance occurs when the number of samples in some classes is significantly higher than those in other classes [12]. Consequently, ML classifiers will be overwhelmed by the majority classes (which have higher number of samples compared to some of the other classes) and ignore the minority

classes (which have fewer instances). In the literature several methods such as over-sampling, under-sampling, stratified sampling (SS), etc. [19], have been used to address the data imbalance problem, as we mentioned earlier.

In addition to over-sampling, we used CTGAN [66] for generating synthetic data for balancing data. To generate synthetic tabular data from original tabular data, CTGAN uses a GAN-based (generative adversarial network) model.

CTGAN introduces **mode specific normalization**, which allows it to deal with columns with complex distributions. This procedure consists of the following three steps.

- Each continuous column C_i is identified by using a variational Gaussian mixture model (VGM) [60] to determine the number mode m_i and fit it in a Gaussian mixture.
- In order to compute the probability density for each mode, it computes the value of c_{ij} in column C_i for *jth* row.
- Then, samples one mode using the calculated probability density and uses the sampled mode to normalize the value.

A new row is resampled in such a way that all categories from the columns are equally distributed at the time of training so that it can be used to capture the actual distribution of data during testing. Let k be the value of the i^{th} column C_i. Suppose $\hat{r}$ is a generated sample, and the original value has to be matched with the generated samples $\hat{r}$ in a way that the generator can be explained as the conditional distribution of rows, given that particular value at that particular column, where

$$\hat{r} \sim P_g(row|C_i = k). \tag{2}$$

One of the most important tasks for the conditional generator is to learn the real distribution of data, i.e., $P_g(row|C_i = k) = P(row|C_i = k)$. The following equation can be used to reconstruct the original distribution.

$$P(row) = \sum_{k \epsilon C_i} P_g(row|C_i = k)P(C_i = k) \tag{3}$$

In order to achieve this, three methods were introduced: conditional vectors, generator losses, and sampling-based training. Two fully connected hidden layers were used in both the generator and discriminator of the network architecture in order to capture all possible correlations between columns. In the generator, batch normalization and relu activation function are used.

D. ML CLASSIFIERS

In this subsection, we discuss various ML classification algorithms we evaluated in this paper.

1) DECISION TREE (DT)

In many applications, DT has been used to classify different types of data such as power quality disturbance, Parkinson's disease, product review classification, etc. [2], [31], [51], [61], [72]. A DT is tree structure, in which each leaf node represents a class label and each internal node is a decision node or a chance node [19]. DT constructs a tree by segmenting the feature space into several subregions. Hence, tree is constructed by recursively binary splitting the feature space [34]. Two splitting methods are usually used to split the tree, namely, cross entropy and Gini index. We used Gini index-based splitting [48]. Gini index can be calculated using Equation 4.

$$Gini = \sum_{l=1}^{L} p_l(1 - p_l) = 1 - \sum_{l=1}^{L} p_l^2, \tag{4}$$

where L is the number of classes and p_l is the set of items in class $l \in \{1, 2, 3, \cdots, L\}$.

2) SUPPORT VECTOR MACHINE (SVM)

SVM model is a renowned machine learning classifier that can be used for both classification and regression tasks. It is, however, primarily used for classification tasks [10], [11], [63], [64]. SVM uses Statistical learning theory to find the optimal hyperplane as a decision function in high dimensional space [45]. We used supervised learning for classification, and considered a input set with N vectors from the d-dimensional feature space X. For each vector x_i, there is a target y_i [5]. The goal of SVM is to identify an optimal hyperplane that maximizes the separation margin. The data are first mapped to a high dimensional feature space using a kernel method, i.e., $\phi(X)$. The optimal hyperplane can be defined as

$$f(x_i) = w \cdot \phi(x_i) + b \tag{5}$$

Here $f(x)$ represents the discriminant function, w is weight vector and b is the bias. b minimizes a cost function. The cost function can be expressed as

$$\psi(w, \xi) = \frac{1}{2}||w||^2 + C \sum_{i=1}^{N} \xi_i. \tag{6}$$

Here ξ_i is a slack variable used for nonseparable data. The constant C is a regularization parameter to control the shape of the discriminant function.

3) Naïve BAYES (NB)

NB classifiers are a family of probabilistic classifiers based on Bayes' Theorem. NB classifiers, combined with kernel density estimation, can achieve high accuracy levels. NB is widely used by researchers to solve various classification problems that arise in their research [14], [18], [23]. NB classifier is based on conditional probability [42]. The probability of one attribute does not affect another attribute, given the class label. Therefore, the presence of an attribute in a class is unrelated to any other attribute. The Naive Bayes probability is defined as

$$P(L|C) = \frac{P(C|L)P(L)}{P(C)}, \tag{7}$$

where L is the class variable and C is the feature set $C_1, C_2, C_3, \cdots, C_Q$. $P(L|C)$, $P(C|L)$, $P(L)$, and $P(C)$ are respectively the posterior probability, probability of feature set given class, prior probability of class, and prior probability of feature set.

4) RANDOM FOREST (RF)

Due to its simplicity and diversity, RF is also one of the most commonly used algorithms. Both regression and classification can be performed using RF [21], [32], [49], [55]. RF combines multiple decision trees to make more accurate and stable predictions. It builds a decision forest based on several decision trees, usually trained with the bagging method. A bagging method, based on the concept that combining different learning classifiers, increases overall performance. In our approach, we used the Gini Index (Equation 4) to determine how a node in a decision tree should be split.

5) FEED-FORWARD NEURAL NETWORK (FNN)

FNNs have been successfully used for pattern classification, clustering, regression, association, optimization, control, and forecasting [4], [9], [28], [67]. FNN contains one input layer, one output layer, and H number of hidden layers. Let $W_h \in \mathbb{R}^{Q\times P}$, and $W_o \in \mathbb{R}^{N\times M}$ be the weight matrices for hidden layer and output layer respectfully where Q is number of input neurons, P is the number neurons in a hidden layer and M is the number of output neurons. Each row of these matrices represents a weight vector for a neuron. Now we can write the equation of output matrix of a hidden layer as:

$$H = f(XW_h + b_h), \tag{8}$$

where $X = \{x_1, x_2, x_3, \cdots, x_N\}$ is the input matrix with N rows, b_h is the bias matrix and $f(.)$ is the activation function of the hidden layer.

We can express the equation of the output layer as:

$$\hat{Y} = g(HW_o + b_o), \tag{9}$$

where $g(.)$ is the activation function of the output layer and b_o is the bias matrix of the output layer.

6) LONG SHORT TERM MEMORY (LSTM)

Although LSTM is a recurrent neural network, it is better in terms of memory than traditional recurrent networks. By memorizing certain patterns, LSTM is able to perform relatively better [26], [43], [52], [68]. LSTM can have multiple hidden layers and as data passes through each layer, the relevant information is retained and the irrelevant information is discarded. An LSTM consists of an input gate i_t, an output gate o_t, and a forget gate f_t. The equations for the LSTM gates at time step t can be expressed as:

$$i_t = g(W_i[h_{t-1}, x_t] + b_i), \tag{10}$$

$$f_t = g(W_f[h_{t-1}, x_t] + b_f), \tag{11}$$

and

$$o_t = g(W_o[h_{t-1}, x_t] + b_o), \tag{12}$$

where $g(.)$ is a activation function of a gate, W_x is the weight of the corresponding gate, h_{t-1} is the output of the previous LSTM block, x is the input vector at time t, and b_x is the bias for the respective gate.

7) CONVOLUTIONAL NEURAL NETWORK (CNN)

In addition to computer vision, CNNs have shown outstanding performance in many other fields [17], [39], [50], [70]. Convolutions are used in this neural network to transform the input features into meaningful information, which is then used to build the subsequent layers of neural network computations. The convolutional layer is used to extract features to perform linear operations, and is usually a combined convolution. In convolution, multiple kernels or filters are used. A convolutional operation is usually defined as:

$$\hat{Y} = x \times k + b. \tag{13}$$

The kernel k has a dimension of $n \times m$. The input and bias are represented by x and b, respectively. The input and bias have the same dimensions k.

III. EXPERIMENTAL RESULTS

In this section, we discuss evaluation criteria, metrics used, datasets used, experimental setup, details about some of the state-of-the-art competing methods, implementation details of classifiers, and performance results of various classifiers.

A. EVALUATION CRITERIA AND METRICS USED FOR EVALUATION

To evaluate the performance of various ML-classifiers, we used the following quantitative metrics: (i) Accuracy (Acc), (ii) Precision (Pre), (iii) Recall (Rec), and (iv) F_1-score, following the relevant literature [29], [44]. Recently, a more robust metric, called The Matthews correlation coefficient (MCC) [13], has been proposed. So, in addition to these metrics (mentioned above and discussed in detail below), we also used MCC in our evaluation. MCC is not affected by the imbalance in datasets. MCC is based on a contingency matrix method used to calculate the Pearson product-moment correlation coefficient. Next, we describe in detail the metrics mentioned above:

- **Accuracy (Acc)**: **Acc** is the measure of how well the algorithm correctly predicts the occurrence of an event. That is, how well an event is predicted as normal or a type of intrusion.
- **Precision (Pre)**: **Pre** refers to how frequently the algorithm correctly predicts the types of intrusions.
- **Recall (Rec)**: **Rec** refers to the proportion of actual intrusions that the algorithm predicted as intrusions.
- F_1**-Score**: F_1**-Score** is the reciprocal of the arithmetic mean of **Pre** and **Rec**, which is the harmonic mean of both variables.

The formulas for calculating these metrics are given in Table 1. To calculate these metrics for various ML classifiers studied in this paper, we counted the True positives (TP),

True negatives (*TN*), False positives (*FP*), and False negatives (*FN*). In this work, all of our ML classifiers are *multi-class*.

TABLE 1. Performance metrics and how they are computed.

Metric	Formula for computing the metric
Accuracy (Acc)	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision (Pre)	$\frac{TP}{TP+FP}$
Recall (Rec)	$\frac{TP}{TP+FN}$
F_1 Score	$2 \cdot \frac{\mathbf{P}re \cdot Rec}{\mathbf{P}re+Rec}$
MCC Score	$\frac{\mathbf{T}P \cdot TN - FP \cdot FN}{\sqrt{(TP+FP) \cdot (TP+FN) \cdot (TN+FP) \cdot (TN+FN)}}$

B. DATASETS USED FOR EVALUATION

We used the datasets NSL-KDD and UNSW-NB15, which are widely used datasets in the intrusion detection literature. Next, we discuss these datasets.

1) NSL-KDD DATASET

NSL-KDD dataset consists of training dataset (KDDTrain+) and two test datasets: KDDTest+ and KDDTest-21. They contain 41 features and it does not contain they do not duplicate records [62]. They have one normal class and four attack type classes.

The four attack types [25] are:

- **Denial of Service (DoS) Attack**: In this type of attack, the attacker blocks resources or services in a system or network through malicious means.
- **User to Root Attack (U2R)**: In this type of attack, the attacker uses a normal user account to gain access to the system and exploits vulnerabilities to take over the system.
- **Remote to Local (R2L) Attack**: In this type of attack, an attacker sends data packets over the network to gain users' access or root access to do unauthorized acts.
- **Probing Attack**: In this type of attack, an attacker gathers information about potential vulnerabilities of target systems so that he/she can launch attacks later.

It is important to highlight that there is significant difference in the sizes of the samples of **U2R** and **R2L** classes in the training dataset. Table 2 shows distribution of samples for various attack types in the training and testing datasets. We can easily see that each of these datasets (training and test datasets) is imbalanced. Figures 2(a) and 2(b) show the partial T-Distributed Stochastic Neighboring Entities (T-SNE) projections for the NSL-KDD testing datasets KDDTest+ and KDDTest-21, respectively. From these projections, we can see that large number of samples are of normal and DoS class types in each of these test datasets.

2) UNSW-NB15 DATASET

UNSW-NB15 [41] is the other dataset we used in our experiments. We obtained this dataset from the University of New South Wales. In the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS), the authors of the dataset generated a hybrid of the realistic modern normal activities and the synthetic contemporary attack behaviors through the use of the IXIA PerfectStorm tool [41]. It has 42 features and includes nine types of attack classes and one normal class. We describe these attack types below:

- **Fuzzers attack:** An attack in which the attacker attempts to discover security holes in software, operating systems, or networks by overloading them with large amounts of random data in order to cause the software to crash.
- **Analysis attack:** An intrusion method for infiltrating the Internet via ports (e.g., port scanning), emails (e.g., spam), and web scripts (e.g., HTML files).
- **DoS attack:** NSL-KDD also has this type of attack class, which we already described in Section III-B1.
- **Backdoor attack**: A way of bypassing normal authentication and securing unauthorized high level access (e.g., root access) and remain undetected.
- **Exploit attack:** A series of instructions that takes advantage of a security flaw, bug, or vulnerability that is caused by an unforeseen action taken by a host or network.
- **Generic attack:** Employs a hash function to establish a collision against every block cipher, regardless of the configuration of the block cipher.
- **Reconnaissance attack:** Also known as **probe**, this is an attack that gathers information about a computer network in order to evade its security measures.
- **Shellcode attack:** A technique used by attackers to gain control of the compromised system by manipulating a small part of the code.
- **Worm attack:** A computer virus, in which the attacker's code replicates itself in order to spread to other computers. Sometimes, it uses a computer network to spread itself by taking advantage of security flaws in the target computer.

Table 3 shows the distribution of the above attack type samples in the training and testing datasets of UNSW-NB15. UNSW-NB15 comes with one training dataset and one testing dataset.

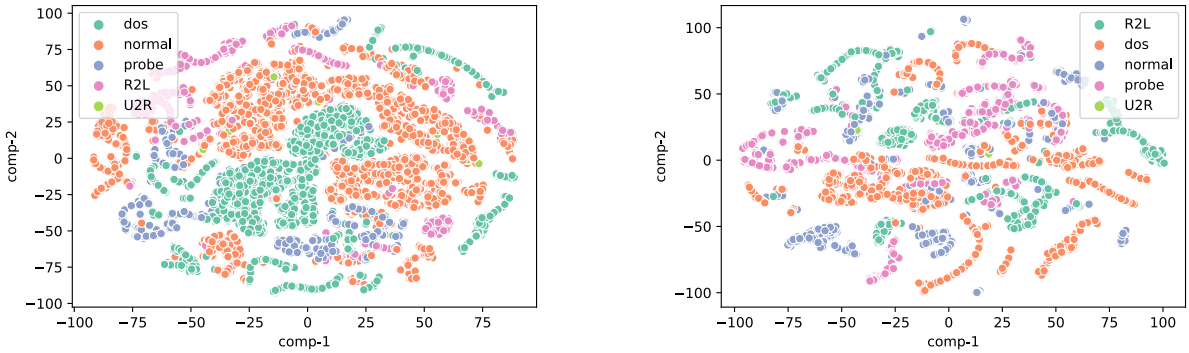
C. EXPERIMENTAL SETUP

To evaluate the performance of various ML classifiers on each of the two datasets NSL-KDD and UNSW-NB15, we conducted the following three experiments.

- **Experiment ORG:** In this experiment, we used the original training datasets of both NSL-KDD and UNSW-NB15 to train the ML classifiers and evaluated their performance.
- **Experiment RandomSamp:** In this experiment, we used random over-sampling [46] to balance the training datasets of both NSL-KDD and UNSW-NB15 and trained the ML classifiers on the balanced training datasets and evaluated their performance.

TABLE 2. Data distribution in NSL-KDD training and testing datasets.

Class	KDDTrain+	(%)	KDDTest+	(%)	KDDTest-21	(%)
Normal	67343	53.5	9711	43.1	13449	53.3
DoS	45927	36.4	7458	33.1	9234	36.7
Probe	11656	9.3	2421	10.7	2289	9.1
U2R	52	0.04	67	0.3	11	0.04
R2L	995	0.78	2887	12.8	209	0.83



(a) KDDTest+ (b) KDDTest-21

FIGURE 2. T-SNE projection of NSL-KDD test datasets.

TABLE 3. Data distribution in UNSW-NB15 training and testing datasets.

Class	Train	(%)	Test	(%)
Normal	37000	75.5	56000	32.0
DoS	4089	8.3	12264	7.0
Generic	18871	38.5	40000	22.8
Exploit	11132	22.7	33393	2.0
Fuzzers	6062	12.4	18184	10.4
Reconnaissance	3496	7.1	10491	6.0
Analysis	677	1.4	2000	1.2
Backdoor	583	1.2	1746	1.0
Shellcode	378	0.77	1133	0.6
Worms	44	0.09	130	0.1

- **Experiment CTGANSamp:** In this experiment, we used the synthetic samples generated by CTGAN to balance the training datasets of both NSL-KDD and UNSW-NB15 and trained the ML classifiers on the balanced training datasets and evaluated their performance.

Next, we describe the setup for each of the above three experiments.

1) SETUP FOR EXPERIMENT ORG

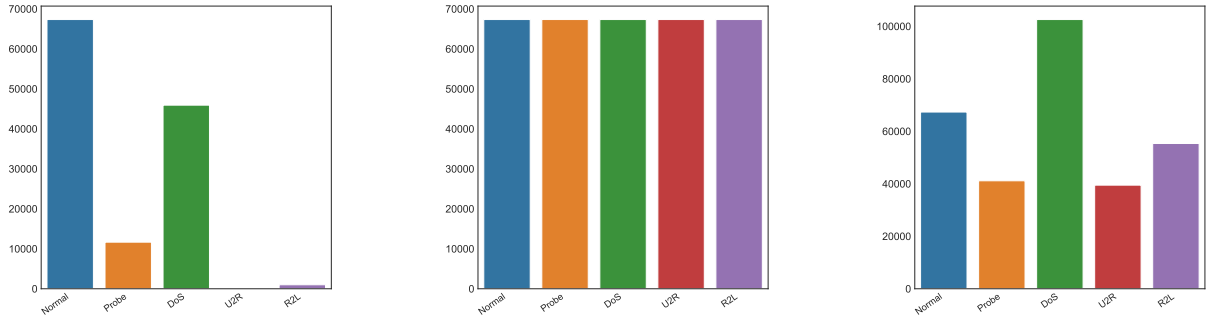
For experiment ORG, we used the original training datasets from both NSL-KDD and UNSW-NB15 to train the ML classifiers and studied their performance. The bar graph in Figure 3(a) shows the distribution of samples under various classes in the original training dataset of NSL-KDD. As shown in Table 2, the percentage of samples under normal and DoS class types of NSL-KDD training dataset are 53% and 36%, respectively. However, the percentage of samples under Probe, U2R, and R2L attack types are approximately 9%, 0.04%, and 0.83% respectively. Similarly, Figure 4(a) shows the distribution of data for various attack classes in the training dataset of UNSW-NB15. As we can see from Table 3, approximately 75% of the samples are in normal class type. On the other hand, samples in Analysis, Backdoor, Shellcode, and Worms attack classes are only 1.2%, 1.2%, 0.77%, and 0.09% respectively. So, training datasets of both NSL-KDD and UNSW-NB15 are highly imbalanced.

2) SETUP FOR EXPERIMENT RandomSamp

For experiment RandomSamp, we balanced the original training datasets of both NSL-KDD and UNSW-NB15 using random over-sampling technique and used the resulting balanced datasets to train the ML classifiers and studied their performance. Random over-sampling is a naive technique for balancing distribution of data under various class types. It involves duplicating samples randomly from minority classes to balance the dataset. In this method, each member of the population in a minority class stands an equal chance of being selected for addition to the dataset. During the entire sampling process, each subject is independently selected from the other members of the population [54]. Figure 3 (b), shows the distribution of data in the NSL-KDD training dataset after balancing the dataset using random over-sampling; there are approximately 67000 samples in each class. After balancing UNSW-NB15 training dataset using random over-sampling, each class had 37000 samples as shown in Figure 4(b).

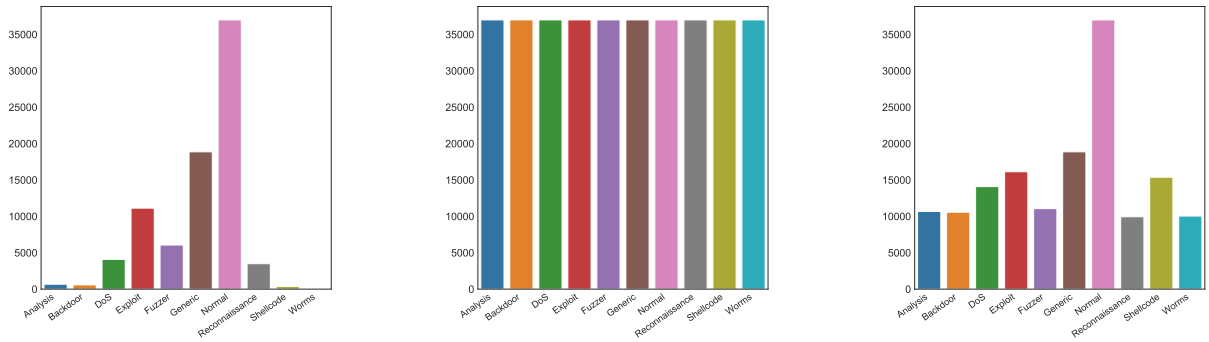
3) SETUP FOR EXPERIMENT CTGANSamp

For experiment CTGANSamp, we balanced the original training datasets from both NSL-KDD and UNSW-NB15, using synthetic data generated with CTGAN [66] and used the resulting balanced datasets to train the ML classifiers and studied their performance. In the original NSL-KDD training



(a) ORG (b) RandomSamp (c) CTGANSamp

FIGURE 3. Data distribution in training datasets (generated from NSL-KDD training dataset) for the three experiments.



(a) ORG (b) RandomSamp (c) CTGANSamp

FIGURE 4. Data distribution in training datasets (generated from UNSWNB-15 training dataset) for the three experiments.

dataset, the total number of samples in **normal** class type is 67343 and total number of samples in **Probe Attack, DoS Attack, U2R Attack**, and **R2L Attack** are 11656, 45927, 52, and 995 respectively. Since the number of samples for normal class type is already high, we decided not to add more synthetic samples to that class type using CTGAN generated synthetic samples. As shown in Figure 3 (c), the distribution of samples after balancing data using synthetic samples generated by CTGAN are 41149, 102589, 39483, and 55350 for the attack types **Probe Attack, DoS Attack, U2R Attack**, and **R2L Attack**, respectively. As we already observed, the original training dataset in UNSW-NB15 is highly imbalanced. After balancing this training dataset with synthetic samples generated by CTGAN, the total number of samples in the attack classes **Analysis, Backdoor, DoS, exploit, Fuzzer, Generic, Normal, Reconnaissance, Shellcode**, and **Worms** are respectively 10677, 10572, 14089, 16132, 18871, 37000, 9946, 15378, and 10044, as shown in Figure 4(c).

D. COMPETING METHODS

To demonstrate the effectiveness of our proposed approach, we conducted extensive experiments using several ML-classifiers. Moreover, we also compared our approach with two recently proposed state-of-the-art models in the literature, namely, CNN-BiLSTM and PB-DID. Next, we discuss these two competing new models.

1) CNN-BiLSTM

CNN-BiLSTM [16] uses 1D-CNN layer with activation function relu and maxpooling of size five. It uses batch normalization to prevent slower training times. In this model, two Bi-LSTM layers have been arranged in a manner that doubles the kernel size in each iteration. The first Bi-LSTM layer had 64 units and the second layer had 128 units. A fully connected dense layer with softmax activation function was used as the final layer. We used the open-source code provided for CNN-BiLSTM [16] in our experiments for performance comparison.

2) PB-DID

PB-DID [71] uses an auxiliary dataset that has common features with the main dataset. Therefore, as in [71], we used the two training datasets NSL-KDD and UNSW-NB15 (with one as main and the other as auxiliary) to train PB-DID for comparison because these two datasets have six common features and two common class types, namely, normal and DoS. So, to study the performance of PB-DID, we added the samples from the training dataset of UNSW-NB15 that are in these common class types to the NSL-KDD training dataset. This merged dataset was used to train the model, and the original testing sets of NSL-KDD were used to test the model. A similar approach (added samples from NSL-KDD to UNSW-NB15) was used to process the UNSW-NB15 training and testing datasets. *Since PB-DID classifier relies on*

an additional dataset to compensate for the data imbalance issue, we did not conduct the experiments RandomSamp and CTGANSamp with PB-DID. We used the open-source code supplied for PB-DID [15] to conduct our experiments.

E. IMPLEMENTATION DETAILS

As we mentioned earlier, we evaluated the performance of various ML classifiers under three experiments, namely, ORG, RandomSamp and CTGANSamp; for each of these three experiments, we used the two datasets - NSL-KDD and UNSW-NB15. We evaluated the following ML classifiers: Decision Tree (DT), multinomial Naive Bayes (NB), Random Forest (RF), Support Vector Machine (SVM), Feed-Forward Neural Network (FNN), Long Short term Memory Network (LSTM), and Convolutional Neural Network (CNN). We implemented DT, NB, RF, and SVM algorithms using the scikit-learn python package version 1.1 [36]. We used the Gini index splitting criteria for DT and L1 regularization for SVM. When estimating an RF, we considered the number of trees in the forest to be 100. With respect to NB, we used alpha 1.0, as smoothing parameter.

We also evaluated the following three neural networks: FNN, LSTM, and CNN. With FNN, we used three hidden layers, each containing 50, 30, and 20 neurons respectively; and the output layer had five neurons. In the final layer, we used softmax function to do the final classification. We used relu as activation function, adam as optimizer, and categorical cross entropy as a loss function. In total, this network had 5285 trainable parameters. We designed a two-layer LSTM. Each layer in the LSTM had 100 units. The activation function, optimizer, loss function and final layer are same as in FNN. Since, our datasets consisted of one-dimensional sequence of data, we used a single one dimensional CNN (conv1D). We used 32 filters with kernel size of 3. We used maxpooling with a pool size of 2. Then, we used a dense hidden layer with 100 neurons and the final layer had five neurons. Like FNN and LSTM, we used the same activation function, optimizer, and loss function for CNN. Each of these networks had been trained for 100 epochs with early stopping. We implemented all three neural networks using tensorflow and Nvidia GPU driver version 455.32.00 with cuda version 11.1. We also evaluated the performance of state-of-the-art models CNN-BiLSTM [16] and PB-DID [71]. Next, we discuss the results from our experiments.

F. RESULTS FROM OUR EXPERIMENTS

1) PERFORMANCE COMPARISON BASED ON THE MCC SCORE

For comparing the performance of ML classifiers, MCC score has been shown to be more robust compared to other metrics such as F_1-score. This is a statistical technique used to evaluate the performance of models. The higher the number, the better the model. So, in addition to the other metrics, we also used MCC score for evaluating the performance of various ML classifiers.

In all three experiments (ORG, RandomSamp, and CTGANSamp), we compared the performance of all the classifiers based on their MCC scores. Tables 4 and 5, contain the MCC scores of different classifiers for the test sets KDDTest+ and KDDTest-21, respectively for the three experiments ORG, RandomSamp and CTGANSamp. The MCC scores of the ML-classifiers under CTGANSamp are consistently higher than their MCC scores under ORG and RandomSamp. Specifically, after ML classifiers were trained on NSL-KDD training dataset, balanced with synthetic data generated with CTGAN, they showed up to 4% improvement in performance with respect to MCC score on the dataset KDDTest+ and up to 7.52% improvement on the dataset KDDTest-21. It is worth noting that when synthetic data generated by CTGAN is used to balance the training dataset, performance of state-of-the-art CNN-BiLSTM also improved; the MCC score of CNN-BiLSTN on KDDTest+ and KDDTest-21 improved by 2.7% and 3.32% respectively, compared to its MCC scores when trained using original training dataset.

The MCC scores of various ML classifiers on the dataset UNSW-NB15 under all three experiments are presented in Table 6. For this dataset also, the MCC score under the experiment CTGANSamp is higher compared to their MC scores under the other two experiments for all ML classifiers. It is worth noting that the MCC score of the state-of-the-art competitor PB-DID on the original dataset of UNSW-NB15 is 35%, is lower than MCC score of all the other classifiers under CTGANSamp. Specifically, the MCC score for all classifiers under the experiment CTGANSamp is up to 61% higher than that of PB-DID, which signifies that the proposed data balancing technique using synthetic data generated with CTGAN, not only improves the performance for well established ML classifiers but also improves the performance of recently published state-of-the-art models such as PB-DID and CNN-BiLSTM.

2) COMPARISON OF PERFORMANCE BASED ON OTHER METRICS

In this subsection, we compare the performance of various ML-classifiers with respect to the following metrics: accuracy (Acc), precision (Pre), recall (Rec), and F_1 score. Table 7 and Table 8 how various classifiers perform under the experiments ORG, RandomSamp, and CTGANSamp, tested on KDDTest+ and KDDTest-21 respectively, with respect to these metrics. The values shown in these tables for various metrics are weighted-average scores [19], [47]. These scores clearly show that the performance improvement in accuracy varied from 1% to 8% under CTGANSamp for both datasets KDDTest+ and KDDTest-21. We also notice that the performance of DT, LSTM, and CNN classifiers is consistently better under CTGANSamp compared to their performance under ORG and RandomSamp for both KDDTest+ and KDDTest-21 datasets with respect to all of the the following quantitative metrics: accuracy, precision, recall, and the F_1 scores. Similarly, we observe significant improvement in

TABLE 4. MCC scores of all classifiers on the KDDTest+ test dataset under the three experiments ORG, RandomSamp and CTGANSamp.

Classifier	MCC-ORG	MCC-RandomSamp	MCC-CTGANSamp
DT	0.6078	0.6437	**0.6518**
SVM	0.5643	0.5547	**0.5999**
RF	0.6288	0.6209	**0.6472**
NB	0.3892	0.2790	**0.4244**
FNN	0.6501	0.6514	**0.6700**
LSTM	0.6607	0.6326	**0.6635**
CNN	0.6374	0.6417	**0.6506**
CNN-BiLSTM	0.6201	0.6343	**0.6471**
PB-DID	0.0794	-	-

TABLE 5. MCC scores of all classifiers on the test dataset KDDTest-21 under the three experiments ORG, RandomSamp and CTGANSamp.

Classifier	MCC-ORG	MCC-RandomSamp	MCC-CTGANSamp
DT	0.4038	0.4639	**0.4790**
SVM	0.3378	0.3379	**0.3822**
RF	0.4443	0.4324	**0.4650**
NB	0.0316	0.0533	**0.0944**
FNN	0.4741	0.4654	**0.4922**
LSTM	0.4684	0.4411	**0.4746**
CNN	0.4406	0.4497	**0.4615**
CNN-BiLSTM	0.4278	0.4363	**0.4610**
PB-DID	0.0012	-	-

TABLE 6. MCC scores of all classifiers on the UNSW-NB15 dataset under the three experiments ORG, RandomSamp and CTGANSamp.

Classifier	MCC-ORG	MCC-RandomSamp	MCC-CTGANSamp
DT	0.5582	0.5319	**0.5788**
SVM	0.3934	0.3387	**0.3941**
RF	0.5579	0.5221	**0.5675**
NB	0.3946	0.2638	**0.3947**
FNN	0.3987	0.3373	**0.5295**
LSTM	0.5300	0.4374	**0.5342**
CNN	0.5336	0.4811	**0.5459**
CNN-BiLSTM	0.4007	0.4559	**0.4854**
PB-DID	0.3587	-	-

recall score for all ML classifiers under CTGANSamp compared to their *recall* scores under ORG and RandomSamp. It is evident that the *FN* rate is low and the *TP* rate is high under CTGANSamp. We would also like to highlight that the accuracy of all the classifiers has been consistently better under CTGANSamp compared to their accuracy under ORG and RandomSamp. With respect to F_1 score, all the classifiers either have better score or have competitive (e.g., within 1.5 percentage point) score under CTGANSamp compared to their scores under ORG and RandomSamp. However, for SVM and NB, the precision scores under CTGANSamp are not as good as their precision scores under ORG or RandomSamp. Additionally, the results also show that the accuracy of some of the classifiers decreased under RandomSamp, compared to their accuracy under ORG. Overall, it can be conclusively claimed that all the classifiers show significant improvement with respect to various metrics on both datasets under CTGANSamp. Next, we discuss the results in more detail.

As shown in Table 7, for dataset KDDTest+, accuracy of DT under ORG is 73.15%; it increases to 74.58% under RandomSamp; it further increases to 75.22% under

TABLE 7. Performance comparison of various ML classifiers on the test dataset KDDTest+ under the three experiments ORG, RandomSamp, and CTGANSamp with respect to the metrics Acc, Pre, Rec and F1-score.

Classifier	ORG				RandomSamp				CTGANSamp			
Name	Acc	Pre	Rec	F_1	Acc	Pre	Rec	F_1	Acc	Pre	Rec	F_1
DT	0.7315	0.7144	0.7315	0.6830	0.7458	0.7992	0.7458	0.7036	**0.7522**	**0.7995**	**0.7522**	**0.7203**
SVM	0.7014	0.6547	0.7014	0.6561	0.6935	**0.7169**	0.6935	**0.6964**	**0.7326**	0.6869	**0.7326**	0.6842
RF	0.7393	**0.8162**	0.7393	**0.6917**	0.7355	0.7485	0.7355	0.6876	**0.7394**	**0.8162**	**0.7394**	0.6916
NB	0.6105	0.5395	0.6105	0.5254	0.4483	**0.5695**	0.4483	0.4937	**0.6273**	0.5168	**0.6273**	**0.5592**
FNN	0.7534	0.7304	0.7534	0.7200	0.7587	0.7567	0.7587	**0.7399**	**0.7736**	**0.8081**	**0.7736**	0.7372
LSTM	0.7629	0.8010	0.7629	0.7260	0.7498	0.7899	0.7498	0.7130	**0.7762**	**0.8182**	**0.7762**	**0.7462**
CNN	0.7505	0.6887	0.7505	0.7021	0.7517	0.7849	0.7517	0.7156	**0.7717**	**0.8037**	**0.7717**	**0.7344**
CNN-BiLSTM	0.7462	0.7932	0.7462	0.7102	0.7603	**0.7596**	0.7603	**0.7350**	**0.7656**	0.7021	**0.7656**	0.7175
PB-DID	0.4526	0.6154	0.4526	0.2878	-	-	-	-	-	-	-	-

TABLE 8. Performance comparison of various ML classifiers on the test dataset KDDTest-21 under the three experiments ORG, RandomSamp, and CTGANSamp with respect to the metrics Acc, Pre, Rec and F1-score.

Classifiers	ORG				RandomSamp				CTGANSamp			
Name	Acc	Pre	Rec	F_1	Acc	Pre	Rec	F_1	Acc	Pre	Rec	F_1
DT	0.4917	0.5956	0.4917	0.4573	0.5248	0.7344	0.5248	0.5020	**0.5423**	**0.7560**	**0.5423**	**0.5345**
SVM	0.4349	0.5120	0.4349	0.4248	0.4823	0.5585	0.4823	**0.4910**	**0.4993**	**0.5664**	**0.4993**	0.4761
RF	0.5036	0.8085	0.5036	0.4846	0.4963	0.6804	0.4963	0.4772	**0.5043**	**0.8092**	**0.5043**	**0.4855**
NB	0.2659	**0.3144**	0.2659	0.2023	0.2772	0.3006	0.2772	**0.2787**	**0.3503**	0.2402	**0.3503**	0.2699
FNN	0.5344	0.5975	0.5344	0.5148	0.5464	0.6425	0.5464	0.5548	**0.5719**	**0.7656**	**0.5719**	**0.5569**
LSTM	0.5535	0.7528	0.5535	0.5370	0.5279	0.7265	0.5279	0.5088	**0.5774**	**0.7837**	**0.5774**	**0.5752**
CNN	0.5289	0.5558	0.5289	0.4997	0.5315	0.7352	0.5315	0.5236	**0.5661**	**0.7575**	**0.5661**	**0.5531**
CNN-BiLSTM	0.5200	**0.7417**	0.5200	0.5066	0.5470	0.6700	0.5470	**0.5236**	**0.5589**	0.5458	**0.5589**	0.5121
PB-DID	0.3382	0.1144	0.3382	0.1710	-	-	-	-	-	-	-	-

TABLE 9. Performance comparison of various ML classifiers on UNSW-NB15 test dataset under the three experiments ORG, RandomSamp, and CTGANSamp with respect to the metrics Acc, Pre, Rec and F1-score.

Classifiers	ORG				RandomSamp				CTGANSamp			
Name	Acc	Pre	Rec	F_1	Acc	Pre	Rec	F_1	Acc	Pre	Rec	F_1
DT	0.656	0.6226	0.6566	0.6268	0.6291	**0.6413**	0.6291	0.6113	**0.6731**	0.6369	**0.6731**	**0.6477**
SVM	**0.5304**	0.3648	**0.5304**	**0.3899**	0.3602	**0.5856**	0.3602	0.3892	0.5257	0.4151	0.5257	0.3832
RF	0.6515	0.6273	0.6515	0.5949	0.5938	**0.7139**	0.5938	**0.6239**	**0.6595**	0.6356	**0.6595**	0.5977
NB	0.5256	0.4080	0.5256	0.3728	0.3095	**0.4424**	0.3095	0.2866	**0.5257**	0.4417	**0.5257**	**0.373**
FNN	0.5295	0.4854	0.5295	0.3860	0.4057	0.5261	0.4057	0.3836	**0.6291**	**0.7656**	**0.6291**	**0.5529**
LSTM	0.6123	0.5933	0.6123	0.5156	0.5054	**0.6729**	0.5054	0.5566	**0.6333**	0.5758	**0.6333**	**0.5622**
CNN	0.6330	0.5746	0.6330	0.5607	0.5547	**0.6818**	0.5547	**0.5944**	**0.6454**	0.5881	**0.6454**	0.5815
CNN-BiLSTM	0.5314	0.375	0.5314	0.3875	0.5250	**0.7093**	0.5250	**0.5795**	**0.5837**	0.5041	**0.5837**	0.4637
PB-DID	0.5471	0.3545	0.5471	0.4292	-	-	-	-	-	-	-	-

CTGANSamp. Similarly, for the dataset KDDTest-21, accuracy of DT under ORG, RandomSamp, and CTGANSamp are 49.17%, 52.48%, and 54.23%, respectively, as shown in Table 8 for KDDTest-21. For KDDTest+, F_1 scores of the classifiers DT, NB, LSTM, and CNN are significantly higher under CTGANSamp compared to their F_1 scores under ORG and RandomSamp. For instance, NB achieved 55.92% F_1 score under CTGANSamp, while its F_1 score under ORG

and RandomSamp are 52.54% and 49.37%, respectively. Table 7 shows that for KDDTest+, SVM, RF, and FNN, had better F_1 scores under ORG and RandomSamp compared to their F_1 scores under CTGANSamp. However, the difference F_1 score is small (e.g., difference in F_1 scores of RF under ORG and CTGANSamp is only 0.01%). Table 8 also shows that there is significant improvement in F_1 scores for DT, RF, FNN, LSTM, and CNN under CTGANSamp compared to their F_1 scores under ORG and RandomSamp. For the same dataset, SVM and NB, had slightly higher F_1 scores under RandomSamp compared to their F_1 scores under CTGANSamp; again, the difference is small (i.e. around 1%). Table 7 indicates that the accuracy of NB on KDDTest+ under RandomSamp decreases by 16% compared to its accuracy under ORG. This is because, performance of the NB classifier is determined by the distribution of samples in the training dataset; so, when duplicate samples are added to balance the training dataset, NB becomes more biased to some identical samples and hence achieved lower accuracy. Several other classifiers including SVM, RF, and LSTM also have lower accuracy score under RandomSamp compared to their accuracy under ORG as seen from Table 7. As shown in Table 8, the accuracy of RF and LSTM on the dataset KDDTest-21 decrease by 0.73% and 2.56%, respectively under RandomSamp compared to their accuracy under ORG. **However, on both datasets KDDTest+ and KDDTest-21, accuracy of all classifiers is higher under CTGANSamp compared to their accuracy under both ORG and RandomSamp.**

Performance of the various machine learning classifiers on the dataset UNSW-NB15 under the three experiments (ORG, RandomSamp and CTGANSamp), with respect to various metrics, shown in Table 9, is similar to their performance on KDDTest+ and KDDTest-21; their performance under CTGANSamp is better compared to their performance under ORG and RandomSamp. In general, accuracy and recall are good performance indicators for the classifiers DT, RF, NB, FNN, LSTM, CNN, and CNN-BiLSTM. Accuracy of SVM under ORG is about 1% higher than its accuracy under CTGANSamp. Other than that, accuracy of all the other classifiers under CTGANSamp is higher than their accuracy under ORG as well as RandomSamp. Precision achieved by various classifiers on the dataset UNSW-NB15, under the three experiments, is similar to their precision on KDDTest+ and KDDTest-21. With respect to F_1 score, all the classifiers performed reasonably well under CTGANSamp compared to their performance under ORG and RandomSamp.

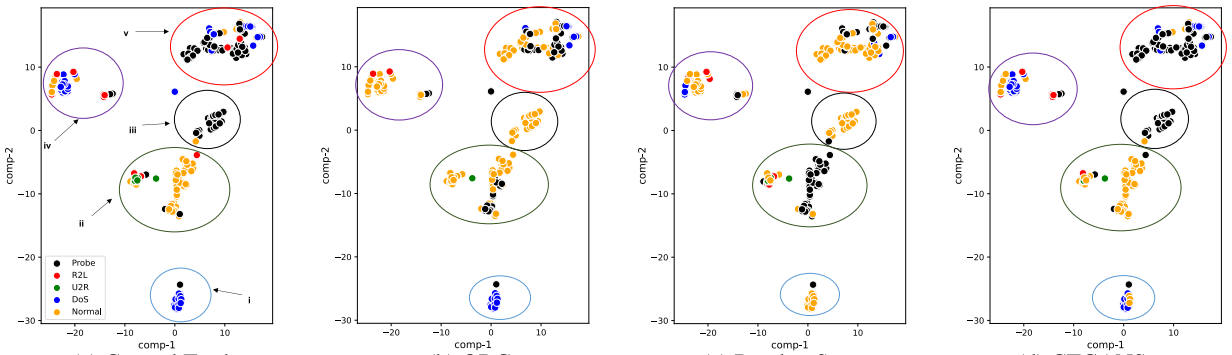
From Tables 7, 8, and 9, it is clear that the training data augmented with synthetic data generated by CTGAN for balancing data resulted in improvement in the accuracy of CNN-BiLSTM on the datasets KDDTest+, KDDTest-21 and UNSW-NB15 by 2%, 3%, and 5% respectively. *We reiterate that we have not conducted RandomSamp and CTGANSamp experiments for PB-DID because PB-DID implementers employed an auxiliary dataset to address the data imbalance issue; thus established data augmentation techniques such as over-sampling or under-sampling have not been utilized in their experiments.*

G. T-SNE PROJECTION OF SELECTED DATA SAMPLES

We randomly drew 300 samples from KDDTest+ and performed T-SNE projection on the selected data samples for qualitative analysis. Figure 5 shows the visualization of T-SNE projection for all three experiments alongside ground truth for the classifier CNN. Figure 5(a) shows the actual classes (ground truth), Figure 5(b) shows the classification under ORG, Figure 5(c) shows classification under RandomSamp and Figure 5(d) shows classification under CTGANSamp. In Figure 5, we drew five circles (i.e., one for each class or cluster) shown in blue (I), green (II), black (III), purple (IV), and red (V) colors. We can see that the classification based on CTGANSamp is much closer to the ground truth.

In the ground truth (Figure 5(a)), majority of the samples inside the blue circle are of type DoS. ORG (Figure 5(b)) performs very well in predicting DoS and its prediction is in line with the ground truth. RandomSamp (Figure 5(c)), on the other hand, predicts them all as normal. CTGAN (Figure 5(d)) has incorrectly predicted some DoS samples as normal, but overall it is able to detect most of the DoS samples. The majority of the samples in the green circle of ground truth are normal, and only a few are U2R, R2L, and probe samples. ORG is not able to detect any of the U2R and R2L samples in the green circle, and some of the normal samples are also labeled as probes. Among all the experiments, RandomSamp performs the worst in predicting the samples in the green circle correctly; RandomSamp labels many normal samples as probes. CTGANSamp, on the other hand, predicts all the normal samples along with the minority classes U2R and R2L correctly. Inside the black circle of the ground truth, majority of the samples are probes; however, ORG and RandomSamp predict all these samples to be normal. CTGANSamp's prediction, on the other hand, is similar to the ground truth. It is also the case with the purple and red circles; CTGANSamp's prediction results of samples in these two circles are the same as ground truth. In contrast, both ORG and RandomSamp fail to capture the actual truth of samples in these two circles as well.

Finally, we also performed statistical significant test based on the performance of the three experiments (ORG, RandomSamp, and CTGANSamp) on the dataset KDDTest+. P-values are presented in Table 10. We notice that the P-values are consistently very small when we compare the performance of CTGANSamp with ORG and RandomSamp. On the other hand, when we compare the performance of ORG with RandomSamp, the P-values are not always small (for example, the P-value of RF is 0.91). This result also signifies that ML classifiers show improvement when they are trained with datasets balanced with synthetic samples generated using CTGAN.



(a) Ground Truth (b) ORG (c) RandomSamp (d) CTGANSamp

FIGURE 5. Qualitative analysis of the CNN classifier under ORG, RandomSamp, and CTGANSamp on KDDTest+.

IV. RELATED WORKS

In this section, we first discuss some intrusion detection techniques based on Machine Learning, presented in the literature. Then, we discuss data augmentation techniques used in various applications.

A. MACHINE LEARNING BASED INTRUSION DETECTION

Machine Learning has been used extensively in designing and implementing IDSes. Ever *et al.* [20] used three machine learning classifiers, ANN, SVM, and DT in their study. The primary goal of this study was to determine the optimal machine learning technique. As part of their experiments, they used 60% and 70% of the dataset NSL-KDD for training, and the rest of the dataset for testing. Based on their experiments, they showed that DT achieved better accuracy compared to the other two.

A new approach to detect intrusion in computer networks was introduced by Gao *et al.* [22]. In order to address the data imbalance problem in NSL-KDD dataset, they proposed a MultiTree algorithm using DT of four levels, with the proportions of the types of classes adjusted accordingly. The authors introduced a model in which they ensembled DT, RF, K-NN, and DNN and used their adaptive voting algorithm to decide on classification.

To build effective IDSes, in depth analysis of network data is mandatory, as the volume of network data increases. Due to the different types of protocols used on the Internet, we have diverse network data. Therefore, it is difficult to distinguish between normal network traffic and attack traffic. Shone *et al.* [56] studied the feasibility and sustainability of current approaches in network intrusion detection. Deep and shallow learning were combined in their model. For unsupervised feature learning, the authors applied two layers of non-symmetric deep auto-encoders (NDAE). Unlike conventional auto encoders, the NDAE contains no decoder. In order to perform the final classification of the network traffic into normal and attack, RF was used. Based on NSL-KDD and KDD99 datasets, the authors evaluated their model using five and thirteen layers of classification. To overcome the problem of over-fitting and under-fitting, they performed a 10-fold cross validation. Due to the imbalanced nature of the datasets, the false alarm rate was high in some attack classes.

Yin *et al.* [69] presented a two-step approach for intrusion detection based on deep learning. One hot encoding was used to transform categorical data to numerical values during the preprocessing stage. In the following step, min-max method was used to normalize the dataset due to large variations in the data distribution. To classify data, recurrent neural networks (RNNs) with forward propagation and backward propagation were used. In the forward propagation method, output values were calculated, and the backward propagation method calculated the error and updated the weights. Cross-entropy was used to compute the difference between the output values produced by forward propagation and the true value. Using this methodology, both binary and multi-class classification were performed.

Javaid *et al.* [27] introduced a deep learning technique based on Auto Encoder (AE) for feature representation and feature learning. They used softmax regression for classification. Additionally, in the preprocessing stage, they transformed categorical features into continuous features and normalized the whole dataset using min-max method. They performed two types of evaluations. In order to do cross validation, they used training data for both training and testing. In the second approach, they used different datasets for testing and training.

In all of the above works, NSL-KDD dataset was used. As we saw, this dataset is imbalanced. *However, none of the authors addressed this issue. The purpose of our study is to focus on the data imbalance problem and to investigate how this impacts the overall performance of various machine learning classifiers.*

B. AUGMENTATION TECHNIQUES APPLIED TO VARIOUS APPLICATIONS

Synthetic data generation or data augmentation has been used in a variety of applications such as image classification and natural language processing. Various augmentation techniques have been proposed, primarily based on deep learning models. In this subsection, we review some recent

TABLE 10. Results of T-Test on the performance of various classifiers on KDDTest+ under ORG, RandomSamp, and CTGANSamp.

Compared \ Classifier	DT	SVM	RF	NB	FNN	LSTM	CNN	CNN-BiLSTM
	P-value	P-value	P-values	P-value	P-value	P-value	P-value	P-value
ORG, CTGANSamp	$3.10e^{-10}$	$3.34e^{-78}$	$2.52e^{-16}$	$3.01e^{-253}$	0.05	0.001	$1.64e^{-37}$	0.004
ORG, RandomSamp	$2.64e^{-17}$	$1.73e^{-301}$	0.91	0.0	$1.79e^{-16}$	$6.65e^{-13}$	$2.05e^{-18}$	$7.75e^{-11}$
RandomSamp, CTGANSamp	$2.69e^{-12}$	$5.70e^{-128}$	$9.99e^{-17}$	$7.94e^{-151}$	$3.46e^{-26}$	$8.01e^{-05}$	0.00028	$5.55e^{-05}$

works on data augmentation and how this technique was applied in different areas of research.

Shorten *et al.* [57] presented a critical survey on image data augmentation using deep learning techniques. They explored the use of data augmentation, a data-space approach to the problem of limited data. Additionally, they state that data augmentation encompasses a suite of techniques to augment the size and quality of training datasets in order to build better deep learning models. This survey discussed image augmentation algorithms including geometric transformations, color space augmentations, kernel filters, mixing images, random erasing, feature space augmentation, adversarial training, generative adversarial networks, neural style transfer, and meta-learning. A significant portion of the survey is devoted to the application of GANs for augmentation.

Li *et al.* [33] proposed a novel deep learning technique for rotating machinery fault diagnosis. Generally, the following five data augmentation techniques were examined: additional Gaussian noise, masking noise, signal translation, amplitude shifting, and time stretching. Sample-based as well as dataset-based augmentation techniques were considered. They used two datasets to conduct their experiments, namely: Bearing Data Center of Case Western Reserve University (CWRU) and Intelligent Maintenance System (IMS). Their approach was able to achieve 99.9% accuracy.

Zhou *et al.* [73] proposed a novel approach combining data augmentation and deep learning methods, which addresses the issue of a lack of training samples in deep learning when used to forecast emerging technologies. In order to construct a sample dataset, Gartner's hype cycle and multiple patent features were utilized. As a second step, a generative adversarial network was used to create many synthetic samples (i.e., data augmentation) in order to expand the sample dataset. Lastly, a deep neural network classifier was trained with the augmented dataset to forecast emerging technologies, and it was able to accurately predict up to 77% of the emerging technologies in a given year. Based on patent data from 2000-2016, this approach was used to predict emerging technologies in Gartner's hype cycles for 2017. A total of four out of six emerging technologies were accurately predicted, demonstrating the precision and accuracy of the proposed method. This article showed that deep learning now can be used to forecast emerging technologies with limited training samples.

Hilda *et al.* [38] used the idea of gradient boosting-based ensembles, such as gradient boosting machine (GBM), extreme gradient boosting (XGBoost), LightGBM, and CatBoost. The goal of this paper is to assess the performance of various imbalanced datasets using metrics such as Matthew correlation coefficient (MCC), area under the receiver operating characteristic curve (AUC), and F_1 score. In this article, an example of anomaly detection in an industrial control network is presented; more specifically, threat detection in a cyber-physical smart grid is discussed. According to the test results, CatBoost outperformed its competitors, regardless of the imbalance ratio in the datasets. In [37], the same authors also address the issue related to data imbalance. They have developed a framework for comparing nine cost-sensitive individual and ensemble models designed specifically to deal with imbalanced data, including cost-sensitive C4.5, roughly balanced bagging, random under-sampling bagging, synthetic minority over-sampling bagging, random under-sampling boosting, synthetic minority over-sampling boosting, AdaC2, and EasyEnsemble.

Sinha *et al.* proposed a hierarchical model by combining 1D-CNN layers and Bi-LSTM layers [58]. CNN is used to identify spatial features of a dataset, while LSTM (essentially a subset of RNN) is used to identify long-term temporal patterns of the dataset, enabling a predictive model to be constructed. They performed both binary and multi class classification on two state-of-the-art datasets, i.e. NSL-KDD, and UNSW-NB15. Additionally, they used random over-sampling in order to balance the two datasets. It is important to note that in both datasets, they initially combined the training and testing sets. Later, they randomly divided the training and testing sets. In our experiments, we evaluated the performance of this approach.

Zeeshan *et al.* [71] introduced an architecture based on Protocol Based Deep Intrusion Detection (PB-DID), in which the authors compared features of UNSW-NB15 and Bot-IoT data-sets based on flow and Transmission Control Protocol (TCP). It was then possible to classify Normal traffic, DoS traffic, and DDoS traffic uniquely by taking into account problems such as imbalance and overfitting. First they found the common features between the UNSW-NB15 dataset and the Bot-IoT dataset. For addressing the imbalance issue in the dataset, they combined both data-sets based on features that fall into the flow and TCP categories. Then, a deep neural network was used to classify them as normal, DoS, and DDoS attacks. During preprocessing, they combined training and testing datasets, and later separated them randomly as training and testing datasets.

ML-classifiers trained with imbalanced datasets affect their performance. We utilized synthetic data generated

with CTGAN, to augment and balance well known training datasets and studied its effect on the performance of various well-known ML-classifiers.

V. CONCLUSION

Over the past several years, many researchers used Machine Learning in designing and implementing IDSes. They used different datasets for training ML classifiers. Most commonly used dataset are: NSL-KDD, UNSW-NB15. In many of the datasets used for designing IDSes, data are imbalanced (i.e., not all classes have equal number of samples). ML classifiers trained on such imbalanced datasets may produce unsatisfactory results which would affect accuracy in predicting intrusions. Traditionally, researchers used over-sampling and under-sampling techniques to balance data in datasets, to overcome this problem. In this work, we used over-sampling, and also used a synthetic data generation method, called Conditional Generative Adversarial Network (CTGAN) to balance data and study their effect on the performance of various ML classifiers. To the best of our knowledge, no one else has used CTGAN to generate synthetic samples to balance training datasets designed for intrusion detection in computer networks. Based on extensive experiments with the widely used datasets NSL-KDD, and UNSW-NB15, we found that training various ML classifiers on data balanced with synthetic samples generated by CTGAN increased not only prediction accuracy by as much as 8%, but also performed well with respect to other metrics such as recall, precision and F_1 score. Moreover, their MCC score increased by as much as 13%, compared to training the same ML classifiers over imbalanced data. Our experiments also show that the accuracy of some ML classifiers trained over data balanced with random over-sampling declined compared to the same ML classifiers trained over imbalanced data. We also compared the performance of two recently proposed models. They also perform much better under our new approach for balancing data.

REFERENCES

[1] S. M. A. Elrahman and A. Abraham, "A review of class imbalance problem," *J. Netw. Innov. Comput.*, vol. 1, no. 8, pp. 332–340, 2013.

[2] S. Aich, K. Younga, K. L. Hui, A. A. Al-Absi, and M. Sain, "A nonlinear decision tree based classification approach to predict the Parkinson's disease using different feature sets of voice data," in *Proc. 20th Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2018, pp. 638–642.

[3] P. J. Anderson, "Computer security technology planning study," Deputy Command Manage. Syst. HQ Electron. Syst. Division (AFSC), Laurence G. Hanscom Field, Bedford, MA, USA, Tech. Rep. ESD-TR-73-51, Oct. 1972, vol. 2.

[4] R. Arulmurugan and H. Anandakumar, "Early detection of lung cancer using wavelet feature descriptor and feed forward back propagation neural networks classifier," in *Computational Vision and Bio Inspired Computing*. Cham, Switzerland: Springer, 2018, pp. 103–110.

[5] Y. Bazi and F. Melgani, "Toward an optimal SVM classification system for hyperspectral remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 44, no. 11, pp. 3374–3385, Nov. 2006.

[6] R. Bridges, T. Glass-Vanderlan, M. Iannacone, M. Vincent, and Q. Chen, "A survey of intrusion detection systems leveraging host data," *ACM Comput. Surv.*, vol. 52, no. 6, pp. 1–35, 2020.

[7] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016.

[8] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016.

[9] A. Catic, L. Gurbeta, A. Kurtovic-Kozaric, S. Mehmedbasic, and A. Badnjevic, "Application of neural networks for classification of patau, edwards, down, turner and klinefelter syndrome based on first trimester maternal serum screening data, ultrasonographic findings and patient demographics," *BMC Med. Genomics*, vol. 11, no. 1, pp. 1–12, 2018.

[10] J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, and A. Lopez, "A comprehensive survey on support vector machine classification: Applications, challenges and trends," *Neurocomputing*, vol. 408, pp. 189–215, Sep. 2020.

[11] M. A. Chandra and S. S. Bedi, "Survey on SVM and their application in image classification," *Int. J. Inf. Technol.*, vol. 13, no. 5, pp. 1–11, Oct. 2021.

[12] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Special issue on learning from imbalanced data sets," *ACM SIGKDD Explor. Newslett.*, vol. 6, no. 1, pp. 1–6, Jun. 2004.

[13] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, pp. 1–13, 2020.

[14] A. Churcher, R. Ullah, J. Ahmad, S. ur Rehman, F. Masood, M. Gogate, F. Alqahtani, B. Nour, and W. J. Buchanan, "An experimental analysis of attack classification using machine learning in IoT networks," *Sensors*, vol. 21, no. 2, p. 446, Jan. 2021.

[15] Code Link. *PB-DID Repo*. Accessed: Jul. 18, 2022. [Online]. Available: https://github.com/nuttysunday/protocol-based-deep-intrusion-detection-for-dos-normal-and-ddos-attacks

[16] Code Link. *PB-DID Repo*. Accessed: Jul. 18, 2022. [Online]. Available: https://github.com/razor08/efficient-cnn-bilstm-for-network-ids

[17] G. Dai, J. Zhou, J. Huang, and N. Wang, "HS-CNN: A CNN with hybrid convolution scale for EEG motor imagery classification," *J. Neural Eng.*, vol. 17, no. 1, Jan. 2020, Art. no. 016025.

[18] X. Deng, Y. Li, J. Weng, and J. Zhang, "Feature selection for text classification: A review," *Multimedia Tools Appl.*, vol. 78, no. 3, pp. 3797–3816, 2019.

[19] A. S. Dina and D. Manivannan, "Intrusion detection based on machine learning techniques in computer networks," *Internet Things*, vol. 16, Dec. 2021, Art. no. 100462.

[20] Y. K. Ever, B. Sekeroglu, and K. Dimililer, "Classification analysis of intrusion detection on NSL-KDD using machine learning algorithms," in *Proc. Int. Conf. Mobile Web Intell. Inf. Syst.* in Lecture Notes in Computer Science, vol. 11673. Cham, Switzerland: Springer, 2019, pp. 111–122.

[21] M. Ezuma, F. Erden, C. K. Anjinappa, O. Ozdemir, and I. Guvenc, "Detection and classification of UAVs using RF fingerprints in the presence of Wi-Fi and Bluetooth interference," *IEEE Open J. Commun. Soc.*, vol. 1, pp. 60–76, 2019.

[22] X. Gao, C. Shan, C. Hu, Z. Niu, and Z. Liu, "An adaptive ensemble machine learning model for intrusion detection," *IEEE Access*, vol. 7, pp. 82512–82521, 2019.

[23] J. Hartmann, J. Huppertz, C. Schamp, and M. Heitmann, "Comparing automated text classification methods," *Int. J. Res. Marketing*, vol. 36, no. 1, pp. 20–38, 2019.

[24] M. Hasan, M. M. Islam, M. I. I. Zarif, and M. Hashem, "Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches," *Internet Things*, vol. 7, Sep. 2019, Art. no. 100059.

[25] B. Ingre and A. Yadav, "Performance analysis of NSL-KDD dataset using ANN," in *Proc. Int. Conf. Signal Process. Commun. Eng. Syst.*, Jan. 2015, pp. 92–96.

[26] B. Jang, M. Kim, G. Harerimana, S.-U. Kang, and J. W. Kim, "Bi-LSTM model to increase accuracy in text classification: Combining Word2vec CNN and attention mechanism," *Appl. Sci.*, vol. 10, no. 17, p. 5841, 2020.

[27] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proc. 9th EAI Int. Conf. Bio-Inspired Inf. Commun. Technol. (Formerly BIONETICS)*, 2016, pp. 21–26.

[28] K. Kamali, "Deep learning (Part 1)—Feedforward neural networks (FNN) (galaxy training materials)," Tech. Rep., Jun. 2021. Accessed: Mar. 16, 2022. [Online]. Available: https://training.galaxyproject.org/

[29] Y. Khourdifi and M. Bahaj, "Heart disease prediction and classification using machine learning algorithms optimized by particle swarm optimization and ant colony optimization," *Int. J. Intell. Eng. Syst.*, vol. 12, no. 1, pp. 242–252, 2019.

[30] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.
[31] S.-H. Kim, I.-J. Moon, S.-H. Won, H.-W. Kang, and S. K. Kang, "Decision-tree-based classification of lifetime maximum intensity of tropical cyclones in the tropical western north Pacific," *Atmosphere*, vol. 12, no. 7, p. 802, Jun. 2021.
[32] H. Komkov, L. Pocher, A. Restelli, B. Hunt, and D. Lathrop, "RF signal classification using Boolean reservoir computing on an FPGA," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2021, pp. 1–9.
[33] X. Li, W. Zhang, Q. Ding, and J.-Q. Sun, "Intelligent rotating machinery fault diagnosis based on deep learning using data augmentation," *J. Intell. Manuf.*, vol. 31, no. 2, pp. 433–452, Feb. 2020.
[34] Link. *Decision Tree*. Accessed: Mar. 10, 2022. [Online]. Available: https://ekamperi.github.io/machine20learning/2021/04/13/gini-index-vs-entropy-decision-trees.html
[35] Link. *Lastline Article*. Accessed: Feb. 21, 2022. [Online]. Available: https://www.lastline.com/blog/examine-tco-of-a-of-a-network-intrusion-detection-system/
[36] Link. *Scikit-Learn*. Accessed: Feb. 21, 2022. [Online]. Available: https://scikit-learn.org/dev/versions.html
[37] M. H. L. Louk and B. A. Tama, "Exploring ensemble-based class imbalance learners for intrusion detection in industrial control networks," *Big Data Cognit. Comput.*, vol. 5, no. 4, p. 72, 2021.
[38] M. H. L. Louk and B. A. Tama, "Revisiting gradient boosting-based approaches for learning imbalanced data: A case of anomaly detection on power grids," *Big Data Cognit. Comput.*, vol. 6, no. 2, p. 41, Apr. 2022.
[39] J. Lu, L. Tan, and H. Jiang, "Review on convolutional neural network (CNN) applied to plant leaf disease classification," *Agriculture*, vol. 11, no. 8, p. 707, 2021.
[40] A. Mottini and R. Acuna-Agost, "Relative label encoding for the prediction of airline passenger nationality," in *Proc. IEEE 16th Int. Conf. Data Mining Workshops (ICDMW)*, Dec. 2016, pp. 671–676.
[41] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6.
[42] S. Mukherjee and N. Sharma, "Intrusion detection using naive Bayes classifier with feature reduction," *Proc. Technol.*, vol. 4, pp. 119–128, Feb. 2012.
[43] P. Nagabushanam, S. T. George, and S. Radha, "EEG signal classification using LSTM and improved neural network algorithms," *Soft Comput.*, vol. 24, pp. 9981–10003, Jul. 2019.
[44] T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Commun. Surveys Tuts.*, vol. 10, nos. 1–4, pp. 56–76, 4th Quart., 2008.
[45] M. Pal and G. M. Foody, "Feature selection for classification of hyperspectral data by SVM," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 5, pp. 2297–2307, May 2010.
[46] Y. Pang, Z. Chen, L. Peng, K. Ma, C. Zhao, and K. Ji, "A signature-based assistant random oversampling method for malware detection," in *Proc. 18th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun./13th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, Aug. 2019, pp. 256–263.
[47] V. M. Patro and M. R. Patra, "Augmenting weighted average with confusion matrix to enhance classification accuracy," *Trans. Mach. Learn. Artif. Intell.*, vol. 2, no. 4, pp. 77–91, 2014.
[48] K. Peng, V. Leung, L. Zheng, S. Wang, C. Huang, and T. Lin, "Intrusion detection system based on decision tree over big data in Fog environment," *Wireless Commun. Mobile Comput.*, vol. 2018, Mar. 2018, Art. no. 4680867.
[49] L. Peng, A. Hu, J. Zhang, Y. Jiang, J. Yu, and Y. Yan, "Design of a hybrid RF fingerprint extraction and device classification scheme," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 349–360, Feb. 2019.
[50] H. Phan, F. Andreotti, N. Cooray, O. Y. Chén, and M. D. Vos, "Joint classification and prediction CNN framework for automatic sleep stage classification," *IEEE Trans. Biomed. Eng.*, vol. 66, no. 5, pp. 1285–1296, Oct. 2018.
[51] O. Reges, A. E. Krefman, S. T. Hardy, Y. Yano, P. Muntner, D. M. Lloyd-Jones, and N. B. Allen, "Decision tree-based classification for maintaining normal blood pressure throughout early adulthood and middle age: Findings from the coronary artery risk development in young adults (CARDIA) study," *Amer. J. Hypertension*, vol. 34, no. 10, pp. 1037–1041, 2021.
[52] S. Saadatnejad, M. Oveisi, and M. Hashemi, "LSTM-based ECG classification for continuous monitoring on personal wearable devices," *IEEE J. Biomed. health Informat.*, vol. 24, no. 2, pp. 515–523, Apr. 2019.
[53] T. Saranya, S. Sridevi, C. Deisy, T. D. Chung, and M. A. Khan, "Performance analysis of machine learning algorithms in intrusion detection system: A review," *Proc. Comput. Sci.*, vol. 171, pp. 1251–1260, Jan. 2020.
[54] G. Sharma, "Pros and cons of different sampling techniques," *Int. J. Appl. Res.*, vol. 3, no. 7, pp. 749–752, 2017.
[55] M. Sheykhmousa, M. Mahdianpari, H. Ghanbari, F. Mohammadimanesh, P. Ghamisi, and S. Homayouni, "Support vector machine versus random forest for remote sensing image classification: A meta-analysis and systematic review," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 6308–6325, 2020.
[56] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.
[57] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, pp. 1–48, Dec. 2019.
[58] J. Sinha and M. Manollas, "Efficient deep CNN-BiLSTM model for network intrusion detection," in *Proc. 3rd Int. Conf. Artif. Intell. Pattern Recognit.*, Jun. 2020, pp. 223–231.
[59] W. Stallings and L. Brown, *Computer Security Principles and Practice*, 4th ed. London, U.K.: Pearson, 2018.
[60] C. M. Bishop and N. M. Nasrabadi, *Pattern Recognition and Machine Learning*, vol. 4, no. 4. Springer, 2006.
[61] M. Syaala and N. J. Nalini, "A filter based improved decision tree sentiment classification model for real-time Amazon product review data," *Int. J. Intell. Eng. Syst.*, vol. 13, no. 1, pp. 191–202, Jan. 2020.
[62] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, Jul. 2009, pp. 1–6.
[63] D. C. Toledo-Pérez, J. Rodríguez-Reséndiz, R. A. Gómez-Loenzo, and J. C. Jauregui-Correa, "Support vector machine-based EMG signal classification techniques: A review," *Appl. Sci.*, vol. 9, no. 20, p. 4402, Oct. 2019.
[64] R. Vijayarajeswari, P. Parthasarathy, S. Vivekanandan, and A. A. Basha, "Classification of mammogram for early detection of breast cancer using SVM classifier and Hough transform," *Measurement*, vol. 146, pp. 800–805, Nov. 2019.
[65] X. Wang, L. Wang, and Y. Qiao, "A comparative study of encoding, pooling and normalization methods for action recognition," in *Proc. Asian Conf. Comput. Vis.* Cham, Switzerland: Springer, 2012, pp. 572–585.
[66] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional GAN," 2019, *arXiv:1907.00503*.
[67] J. Yang and J. Ma, "Feed-forward neural network training using sparse representation," *Expert Syst. Appl.*, vol. 116, pp. 255–264, Feb. 2019.
[68] O. Yildirim, U. B. Baloglu, R.-S. Tan, E. J. Ciaccio, and U. R. Acharya, "A new approach for arrhythmia classification using deep coded features and LSTM networks," *Comput. Methods Programs Biomed.*, vol. 176, pp. 121–133, Jul. 2019.
[69] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.
[70] C. Yu, R. Han, M. Song, C. Liu, and C.-I. Chang, "A simplified 2D-3D CNN architecture for hyperspectral image classification based on spatial–spectral fusion," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 2485–2501, 2020.
[71] M. Zeeshan, Q. Riaz, M. A. Bilal, M. K. Shahzad, H. Jabeen, S. A. Haider, and A. Rahim, "Protocol-based deep intrusion detection for DoS and DDoS attacks using UNSW-NB15 and Bot-IoT data-sets," *IEEE Access*, vol. 10, pp. 2269–2283, 2021.
[72] W. Zhao, L. Shang, and J. Sun, "Power quality disturbance classification based on time-frequency domain multi-feature and decision tree," *Protection Control Modern Power Syst.*, vol. 4, no. 1, pp. 1–6, Dec. 2019.
[73] Y. Zhou, F. Dong, Y. Liu, Z. Li, J. Du, and L. Zhang, "Forecasting emerging technologies using data augmentation and deep learning," *Scientometrics*, vol. 123, no. 1, pp. 1–29, 2020.

AYESHA SIDDIQUA DINA received the B.S. degree in computer science and engineering from the University of Chittagong. She is currently pursuing the doctorate degree in computer science with the University of Kentucky. She has published several articles in different journals and conferences. Her current project involves intrusion detection using data driven techniques in a computer networks. Her research interests include computer security, machine learning, the IoT, cloud computing, and bioinformatics.

A. B. SIDDIQUE is an Assistant Professor with the Computer Science Department, University of Kentucky. He explores research problems in unsupervised learning, zero-shot learning, and conversational AI. He also investigates emerging research topics in open-domain natural language interfaces and knowledge-grounded response generation, and builds reliable and practical solutions that may seamlessly adapt to new unseen domains on-the-fly. He is interested in exploring opportunities to leverage NLP and ML methods across disciplines. His research interests include machine learning, natural language processing, and scalable data science.

D. MANIVANNAN (Senior Member, IEEE) received the B.Sc. degree in mathematics from the University of Madras, Madras, India, the M.S. degree in mathematics, the M.S. degree in computer science and the Ph.D. degree in computer and information science from The Ohio State University, Columbus, Ohio, USA, in 1992, 1993, and 1997, respectively.

He is currently an Associate Professor with the Computer Science Department, University of Kentucky, Lexington, Kentucky, USA. He has more than 80 publications in refereed international journals (a vast majority of which were published by IEEE, ACM, Elsevier, and Springer) and proceedings of international conferences. He has funded by grants from the U.S. National Science Foundation and the U.S. Department of Treasury. His research interests include fault-tolerance and synchronization in distributed systems, security and fault-tolerance in cloud computing systems, routing in wormhole networks, routing in mobile ad hoc networks and vehicular ad hoc networks, vehicular clouds, channel allocation in cellular networks, and wireless personal area networks.

Dr. Manivannan is a Senior Member of ACM. He was a recipient of the Faculty Early Career Development Award (CAREER award) from the U.S. National Science Foundation. He was the Program Co-Chair of three International Conferences in the areas of reliable distributed systems and wireless networks and the program committee member for over 40 International conferences. He was an Associate Editor of IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, *IEEE Communications Magazine*, and *Wireless Personal Communications Journal*. He is currently on the Editorial Board of *Information Sciences*, *Internet of Things*, and *Network* journals. He was a Reviewer for more than 40 international journals published by ACM, IEEE, Elsevier, Springer, Oxford University Press, Taylor and Francis, and others. He was on several proposal review panels of U.S. National Science Foundation and as an external tenure reviewer for other Universities.