A Novel Study: GAN-Based Minority Class Balancing and Machine-Learning-Based Network Intruder Detection Using Chi-Square Feature Selection

Amerah Alabrah

Abstract: The network security problem becomes a routine problem for networks and cyber security specialists. The increased data on every minute not only creates big data problems, but also it expands the network size on the cloud and other computing technologies. Due to the big size and data, the network becomes more vulnerable to cyber-attacks. However, the detection of cyber-attacks on networks before or on time is a challenging task to solve. Therefore, the network intruder detection system (NIDS) is used to detect it. The network provided data-based NIDS were proposed previously, but still needed improvements. From the network data, it is also essential to find the most contributing features to avoid overfitting and lack of confidence in NIDS. The previously proposed solutions of NIDS mostly ignored the class imbalance problems that were normally found in the training of machine learning (ML) methods used in NIDS. However, few studies have tried to solve class imbalance and feature selection separately by achieving significant results on different datasets. The performance of these NIDS needs improvements in terms of classification and class balancing robust solutions. Therefore, to solve the class imbalance problem of minority classes in public datasets of NIDS and to select the most significant features, the proposed study gives a framework. In this framework, the minority class instances are generated using Generative Adversarial Network (GAN) model hyperparameter optimization and then the chi-square method of feature selection is applied to the fed six ML classifiers. The binary and multi-class classifications are applied on the UNSW-NB15 dataset with three versions of it. The comparative analysis on binary, multi-class classifications showed dominance as compared to previous studies in terms of accuracy (98.14%, 87.44%), precision (98.14%, 87.81%), F1-score (98.14%, 86.79%), Geometric-Mean (0.976, 0.923) and Area Under Cover (0.976, 0.94).

Keywords: chi-square; class imbalance; feature selection; GAN; network intruder detection; machine learning

1. Introduction

Internet usage and growing technologies raise the risk of cyber security installments on cloud computing, edge computing, and other networks. These malicious attacks lead to financial and reputational losses. However, the network intrusion detection system (NIDS) installed on these networks prevents these cyber-attacks [1–3]. NIDS basically monitor the network continuously within its cyberspace [4]. The history of intrusion detection systems starts at 1980 where J. Anderson et al. proposed a method [5] that was enough to secure a network at that time with required security. However, the immense progress in technologies in recent decades has created many challenges regarding network security. The big data created by technologies nowadays makes millions of gigabytes (GBs) of data that are shared across the network nodes.

The network itself becomes larger in size and it also challenging to maintain the network safely. Network security also becomes more challenging when certain types of
cyber-attacks come around. The false rate of NIDS increased with the arrival of these different types of attacks [6]. Therefore, it has become a worldwide issue to make such types of NIDS that not only detect the attack accurately and precisely, but also detect the attack type to tackle it correspondingly.

Although these are the specific challenges of NIDS that need to be solved, certain challenges also occur while the NIDS are proposed. NIDS typically made using machine [7,8] and deep-learning [9,10] intelligent methods. However, to make these models, the network-logs-based generated data are used to train and test the models. The class imbalance problem is very common in these datasets. The class imbalance occurs when the ratio of certain categories, such as positive class, is increased as compared to another class or negative class [11,12]. It increases the prediction probability of trained ML and Deep Learning (DL) models being more likely toward the class that contains more instances in a given dataset. This class imbalance ratio reached 1:50%+ in the UNSW-NB15 dataset against each category [13] as compared to the normal class.

The dataset used in NIDS is mostly the remaining class imbalance risen to compromise the previously proposed NIDS with a higher false rate of NIDS [14]. Further, the class imbalance issue almost remains unsolved while any NIDS is proposed. However, few studies have solved this class imbalance issue and proposed their frameworks of NIDS. These proposed NIDS still need improvement in terms of improved precision rate and use the robust method to solve the problem of class imbalance [15,16]. The class imbalance problem is solved mainly on two levels [17]. It could be solved using the data level scheme that focuses on class balancing using the class distribution methods, such as the synthetic minority oversampling technique (SMOTE) [18].

The other method to solve class imbalance is algorithm-based solutions. In this method, the cost metric uses a cost-sensitive function to solve misclassified instances on validated data. The neural network is a simple example of this that uses a cost-calculating method to assign reweighing to misclassified instances [19]. In this way, the algorithm reaches a certain condition such that it removes the misclassification behavior of misclassified instances and makes a balanced dataset. The other problem after the data disbalancing is the overfitting of data, which is the problem for ML and DL models. The overfitting could be reduced using appropriate features selection that ultimately leads to the performance enhancement of NIDS models.

The wrapper, embedded, and filter are a few famous approaches that are used for feature-selection purposes. However, ML and evolutionary computing also provided a few methods that were used as feature selection methods, such as genetic algorithm (GA), particle swarm optimization (PSO), etc. [20]. However, these applied methods ultimately aimed to improve the model performance, which was validated by their proposed studies.

However, to solve the discussed problems on NIDS, the conducted study proposed a framework. In this framework, we have the following:

• The class imbalance problem is solved using the generative adversarial network (GAN) model hyperparameters optimization for tabular or numeric data generation.
• The appropriately generated new data using the UNSW-NB15 dataset reduced the class imbalance problem against different categories of network attacks.
• The generated dataset-based results are compared with the original UNSW-NB15 dataset that proved the validity of the dataset and enhanced its precision rates of classification.
• The chi-square method is used for features selection, where classical ML methods are used for the binary and multi-class classification of network attacks on original and newly generated datasets.
• The results and comparative analysis showed the outperformance of the proposed framework as compared to previous studies such that the proposed framework is a more reliable and valid method of network intrusion detection.

The rest of the article is divided into five sections. The related work section reviews the previous studies about NIDS; the proposed methodology and discussion in Section 3 high-
light the operational functionality of a proposed framework; Section 4 is about the results
and validity of proposed study; and Section 5 is about a comparison to previous studies,
and lastly the whole framework is concluded, with weaknesses and future directions.

2. Related Work

The proposed framework specially targets the class imbalance issue using an algorithm-
based approach and, later on, an applied feature selection method to provide efficient NIDS.
Therefore, a summary of the recently applied relevant studies are discussed here and shown
in Table 1 with their weaknesses and strengths to enhance understanding about the pro-
posed framework. The class imbalance issue was discussed in a study by X. Tan et al. [21]
and the KDD 99 Cup dataset was used in it to make NIDS, where the random forest classi-
fier was used for classification. It showed in the results that the classification was improved
(from 92.39% to 92.57%) while using the SMOTE-based solution of the class-imbalance
problem. Other classifiers were also used, and the best one is discussed here.

Another study [22] used SMOTE and Gaussian mixture model (GMM)-based methods
to oversample and solve the class imbalance problem. Two datasets, UNSW-NB15 and
CICIDS2017, were used. The binary classification using the proposed SGM-CNN on
UNSW-NB15 obtained an accuracy of 98.82%, and a precision of 95.53% F1-score, where the
multi-class achieved 96.54% accuracy and 97.26% F1-score. The other dataset was classified
on multi-class only and showed 99.85% accuracy with 99.86% F1-score.

The deep-learning-based study [23] proposed a Bi-LSTM model that assigns the
attention weights via adaptive synthetic sampling (ADASYN) to handle the class imbalance
problem found in the NSL-KDD dataset. The accuracy reaches up to 90.73% and 89.65% F1-score. T. Wao et al. [24] proposed a NID based on SMOTE and the KNN clustering
method to solve the class imbalance problem found in the NSL-KDD dataset and then
applied the random forest classifier to solve the classification problem. It achieved 78.47%
accuracy on the test set.

The Focal loss is a function mostly used in ML to measure the loss-based performance
of a model to classify the instances. A study [25] used this cost-sensitive function to solve
the class imbalance problem and proposed a deep neural network (DNN) and convolutional
neural networks (CNN) to classify data on three datasets (NSL-KDD, UNSW-NB15, and
Bot-IoT). The results section of this study is enriched with scalability-based data, the layer-
wise results evaluation for binary and multi-class classification on all datasets. However,
the final results for binary DNN showed 83.92%, 90.41%, and 79.24% F1-score for the
NSL-KDD, UNSW-NB15, and Bot-IoT datasets. For multi-class, it showed 47.33%, 39.78%
and 98.90% F1-scores, respectively. On the second model CNN for binary classification, it
showed 84.87%, 86.03%, and 95.57% F1-scores, where for multi-class, it showed, 51.96%,
39.52% and 95.51% F1-scores, respectively.

The algorithm level class imbalance problem is solved [26] by applying several ML
classifiers on the NSL-KDD and UNSW-NB15 datasets. It modifies the cross-entropy
function that is used as a loss function and by applying data normalization. The final
achieved classification scores on binary data are 90.76% for the UNSW-NB15 dataset, and
85.56% for the NSL-KDD dataset.

Secondly, the features-selection methods reduce the training and testing times of NIDS
solutions with the most appropriate features engagement while performing classification.
One of the recent studies [20] applied various methods, such as GA, PSO and many others
to check the effect, where the top16 features were selected from the UNSW-NB15 dataset to
classify the binary data. The maximum scores achieved by the J48 algorithm were 90.48%
accuracy, 84.136% precision, 97.141% sensitivity, and 90.172% F1-score.

However, we have seen a lot of studies applied on class imbalance problem solutions,
where the proposed study not only solved the class imbalance problem, but also applied
feature selection based upon the chi-square method of features ranking and selected the
topmost-ranked features. Similarly, we can see that few studies have been applied sepa-
rate for class imbalance solutions and feature-selection methods. Therefore, the proposed
work gives both solutions with appropriate combination and solves the binary and multi-class classification problems with improved performance as compared to previous studies.

Table 1. Summary of recently applied NIDS studies using class imbalance problem solving.

| References | Year | Methods | Dataset | Results |
|------------|------|---------|---------|---------|
| [21]       | 2019 | SMOTE to solve class imbalance and classical classification method | KDD 99 Cup | Highest Accuracy = 92.57% |
| [22]       | 2020 | SMOTE, GMM, and CNN | UNSW-NB15 | Binary-Accuracy = 98.8%, F1 = 95.53% |
| [22]       | 2020 | SMOTE, GMM, and CNN | UNSW-NB15 | Multi-Accuracy = 96.54%, F1 = 97.26% |
| [22]       | 2020 | SMOTE, GMM, and CNN | CICIDS2017 | Multi-Accuracy = 99.85F1 = 99.86% |
| [25]       | 2020 | Focal-Loss based DNN and CNN | NSL-KDD | DNN Bi.-F1 = 83.92%, M.-F1 = 47.33% CNN Bi.-F1 = 84.87%, M.-F1 = 51.96% |
| [25]       | 2020 | Focal-Loss based DNN and CNN | UNSW-NB15 | DNN Bi.-F1 = 79.24%, M.-F1 = 98.90% CNN Bi.-F1 = 95.57%, M.-F1 = 95.51% |
| [23]       | 2022 | Bi-LSTM and ADASYN method of assigning attention weights for class imbalance | NSL-KDD | Accuracy = 90.73%, F1-Score = 89.65% |
| [24]       | 2022 | SMOTE-KNN for class imbalance and random forest for classification | NSL-KDD | Testing-Accuracy = 78.47% |
| [26]       | 2022 | Modified cross entropy applied for class imbalance and Neural Network applied for Classification | NSL-KDD | Accuracy = 85.56% |
| [26]       | 2022 | Modified cross entropy applied for class imbalance and Neural Network applied for Classification | UNSW-NB15 | Accuracy = 90.76% |

3. Proposed Methodology

The proposed framework applies certain schemes to make balanced, robust, and improved NIDS. It includes certain steps to perform class balancing and features-selection-based NIDS that are shown in Figure 1.

The proposed framework includes dataset cleaning and the removal of unnecessary features from the given dataset. The meaningful features are then further selected before classification. However, at first, the minority class is balanced with the GAN method, and instances are increased with the appropriate method. The detailed discussion in coming sections.
3.1. GAN-Based Minority Class Data Generation

GAN is basically a method that works on two main objects or networks in it: generator and discriminator. The generator is given with a latent space regarding instances based on some values to it, and then random noise is added into it. The noise is then given to a discriminator that compares it with a real dataset sample to check if the value is similar or nearer to it. The sigmoid and other loss functions are used in it. Each time the noise is rejected from the discriminator; the value is changed with the help of the weight update functions.

The loss function is set to reduce each time, and when it is reduced to a certain threshold, the new value of the generated sample is accepted. By using this basic principle of the GAN, the tabular data generator is used by the proposed study to create minority class instances. The tabgan API [27,28] is used to obtain the GAN model solution, the hyperparameters optimization is discussed in Section 4. The generated values are compared and shown in Figure 2 for few feature vectors that show similarity with the original data.

All data of features from the UNSW-NB15 dataset are fed to the GAN model to generate new instances where, for better understanding, only 10,000 instances of data are shown in Figure 2 to see the similarity between the generated data and original data. If we look at the left side of Figure 2, at the proto feature, the range is equalized to 100 or slightly more than 100, where on the right side, the generated data also contain a similar range, even if the range is higher at the initial values; then, the range is again higher in GAN-based proto data. This is similarly true for service, state, spkts, sbytes, etc.

For all the features, we can look at the left and right histograms showing very similar behavior among the original and newly generated data. These features are basically generated for each attack type given the dataset. The data for the normal class are not considered for giving and training while making the new instances, as they already have many instances. However, the data are later checked with both binary and multi-class data type classifications.
3.2. Dataset Preprocessing

The values in all features are converted into numeric form, even if those are in categorical format. After obtaining the refined numeric values in feature columns, clamping is performed with certain conditions as described in Equations (1)–(3).

\[ F_1 = \text{Max}(FV_i) > 10 \]  
\[ \text{Equation (1)} \]

Equation (1) represents a flag \( F_1 \) that is made to check if the iterating feature vector \( (FV_i) \) maximum value is greater than 10 times:

\[ F_2 = \text{Med}(FV_i) \ast 10 > F_1 \]  
\[ \text{Equation (2)} \]

The second flag \( F_2 \) in Equation (2) checks whether the feature vector value is 10 times greater than the median of all data of this particular feature: \( (FV_i) \) maximum value is more than 10 times or not.

\[ \text{Prun}_x = if (F_1 \& F_2) = 1 \]  
\[ \text{Equation (3)} \]

In Equation (3), if the feature vector values are flagged to true means, then at a particular instance of the feature vector, the value is found to be 10 times greater than the median of this feature, and then that particular instance value with 98th percentile of that column should be replaced. In this way, all values of all features do not go beyond the limits and are pruned over the 98% percentile. The clamping plays an important role in normalizing the values.

The greater the categorical values, the greater the skewness. Therefore, in order to reduce the skewness among those features that have more than 50 categorical values, the log function is applied, and the mathematical representation is shown in Equation (4):

\[ \text{Norm}_{FV} = \log(\text{Prun}_i) \]  
\[ \text{Equation (4)} \]
In Equation (4), we can see that all features that are collected after performing the pruning operations are given to a log function to remove the skewness in those columns that have uniqueness greater than 50 times.

3.3. Features Selection Using Chi-Square

Selection of meaningful and most significant features increases the precision value and reduces the training and testing time with the reduction in overfitting problems as well. Therefore, the features ranking using chi-square is applied, where how much features should be selected is interpreted manually by giving 16 features as the final selected features. The manual selection is also based upon performance, as the further increase of features does not increase the classification models’ performance, where reducing the features by more than 16 reduces their performance. Therefore, the most valuable 16 features are selected based upon their ranking. The operational functionality of the chi-square method is shown in Equation (5):

\[ SF_c^2 = \sum \left( \frac{O_i - E_i}{E_i} \right)^2 \]  

The selected features (SF) are calculated as the degree of freedom (c) assigned while applying the chi-square method, where O and E are the observed and expected values. The i is calculated over each feature. The top 16 features are shown in Figure 3 on UNSW-NB15 binary classification. The y-axis of the figure shows the feature names selected from all 42 features of the UNSW-NB15 dataset, where the x-axis shows the scores achieved by each feature in order to predict the normal and attacked instances of the dataset.

![Figure 3. The most valuable 16 features for binary classification using chi-square features selection.](image-url)
There are five main steps involved in the chi-square method of feature selection. The hypothesis-based operation is performed, which is based upon contradicted behaviors. The contingency table is built upon values of given data. The expected value is calculated, and then the chi-square value is calculated according to Equation (5). The acceptance of the hypothesis is performed based on the chi-square value, where the rule is to select the higher values that show a greater tendency toward the output classes. The chi-square selected 16 features are given to the ML classifiers. The features selection shows different features on binary and multi-class classification. It means the different features show varying levels of importance in the multi-class classification of network security attacks.

3.4. ML Classification

There are three datasets that are evaluated and classified on binary and multi-class tasks. The original dataset is given to chi-square to select the top 16 features and then fed to ML classifiers to apply binary and multi-classification. However, to validate the newly generated data attacked categories instances and the original data, the (1) original (UNSW-NB15), (2) GAN and (3) original + GAN datasets are used to show the consistent behavior of all classifiers on binary and multi-class classifications.

The holdout validation scheme is used by randomly splitting all data into an 80:20 ratio of training and testing. The decision tree, extra trees, random forest, logistic regression, k-nearest neighbor (KNN), and multi-layer perceptron (MLP) types of classifiers are applied to validate the performance of GAN-based data, and the feature-selection method is used by the proposed study. These are all discussed in detail in this section. The full framework-based algorithm is shown in Algorithm 1.

**Algorithm 1** Proposed-framework-based algorithm from features input to classification.

Input: UNSW-NB15 dataset features \((FV_i)\)

Output: Classification of three datasets \(D_1, D_2, D_3\)

Step 1: Take all features \((FV_i)\).

Step 2: Separate out minority class instances \((MV_i)\).

Step 3: Generate new minority class instances \((NMV_i)\) using hyperparameter-optimized GAN model used in proposed study.

Step 4: Separate out three datasets: Original UNSW-NB15 Dataset \((D_1)\), combined data of newly generated minority class in Step 3 and original dataset based normal class instances \((D_2)\) and combined data of original UNSW-NB15 instances + GAN based newly generated minority class instances \((D_3)\).

Step 5: Features normalization using Equations 1, 2, 3 and 4 on \(D_1, D_2, D_3\).

Step 6: Apply chi-square feature selection on \(D_1, D_2, D_3\).

Step 7: Obtained three feature sets \((GV_i)\) based upon chi-square method using \(D_1, D_2, D_3\)

Step 8: Conducted Experiments 1, 2 and 3 using \(D_1, D_2, D_3\) by feeding them to ML classifiers

The full algorithm is shown to give a step-by-step understanding of the proposed study. The input is taken from the UNSW-NB15 dataset by separating out the class attribute columns. However, at Step 2, based upon the category of each network attack type, the class-wise instances are separated out. In Step 3, the class-wise new instances from the given instances of UNSW-NB15 \((D_1)\) are generated using the GAN model that is optimized with the hyper-parameters selection. To validate the performance of newly generated minority class instances and to cross check the performance as compared to the original dataset \((D_1)\), three datasets \(D_1, D_2, D_3\) are made in step 4. Features normalization on all three sets is performed using Equations (1)–(4) in Step 5. After applying features normalization, the most important features are selected using the chi-square method on all feature sets in Step 6. The selected features \((GV_i)\) for each feature set are fed to six classifiers in Step 8, where both binary and multi-class classifications are performed in three experiments.
4. Results and Discussion

The original UNSW-NB15 dataset, the GAN-based a balanced dataset of minority-classes and the original + GAN-based datasets are used to perform experimentation on NIDS development. The dataset instances before and after GAN are discussed in detail. The feature-selection-based results are collected in three different experiments on six different classifiers of the ML domain.

4.1. Datasets Description

The original dataset contains labels for both multi-class and binary class classification. The class-wise number of instances in each class is shown in Table 2. The table shows that the instances for the normal class are too many as compared to other categories of network attacks. Therefore, the more instances for attacked classes are generated using the GAN model, and their numbers are shown in Table 2. The normal class instances are enough to use in classification; therefore, more instances for the normal class are not generated. The final collective dataset using original and GAN datasets is also described in Table 2, in correspondence to the total instances against each class.

| Classes       | Original Instances (Train + Test) | GAN-Based Instances (Train + Test) | Original + GAN-Based Instances |
|---------------|----------------------------------|------------------------------------|--------------------------------|
| Analysis      | 2000 + 677 = 2677                | 1477 + 453 = 1930                  | 4607                           |
| Backdoor      | 1746 + 583 = 2329                | 1264 + 366 = 1630                  | 3959                           |
| DoS           | 12,264 + 4089 = 16,353           | 9588 + 2746 = 12,334               | 28,687                         |
| Exploits      | 33,393 + 11,132 = 44,525         | 26,217 + 7588 = 33,805             | 78,330                         |
| Fuzzers       | 18,184 + 6062 = 24,246           | 14,089 + 4150 = 18,239             | 42,485                         |
| Generic       | 40,000 + 18,871 = 58,871         | 31,609 + 12,984 = 44,593           | 103,464                        |
| Reconnaissance| 10,491 + 3496 = 13,987           | 8132 + 2348 = 10,480               | 24,467                         |
| Shellcode     | 1133 + 378 = 1511                | 827 + 207 = 1034                   | 2545                           |
| Worms         | 130 + 45 = 175                   | 54 + 15 = 69                       | 244                            |
| Normal        | 56,000 + 37,000 = 93,000         | 56,000 + 37,000 = 93,000           | 93,000                         |
| Total (Attacked + Normal) | 164,674 + 93,000 = 257,674 | 124,060 + 93,000 = 217,060        | 288,788 + 93,000 = 381,788    |

The dataset in detail described against each category is shown in Table 2. The instances of each category in Column 2 show the training and testing sets provided by the UNSW-NB15 dataset originally. The binary classification is based on all attacks considered as attacks, where the normal is considered a negative class in that category. The last row shows almost 64 attacked class data as compared to the normal class instances that are the remaining 36% of data. We see here that the percentage against a single class is 36%, which is a problem in binary classification, where in multi-class classification, it is greater, as a few of the categories, such as worms, are very low in the whole dataset. To solve it separately against each class, the GAN is applied on each category separately and the new corresponding instances are made against them, shown in Column 3.

The number of new instances against each category is slightly lower as compared to the original dataset instances of similar categories. Higher number of given instances for a category of GAN create new instances in a similar way, either in terms of the value or number of instances. However, the normal and attacked instances using GAN show 57% for attacked classes and 43% for normal class. Although it is less compared to the original dataset instances, it plays an important role in the creation of a new and more balanced dataset in favor of attacked classes. The new dataset contains collectively 75% for attacked classes and 25% for the normal class, which is better compared to original
and GAN-based instances. However, the classification for multi-class and binary is discussed on each of these datasets to check the validity of the created dataset and its instances.

4.2. GAN Hyperparameters Optimization and Learning Environment

The GAN method used in the proposed framework was proposed originally in 2020. The authors developed it to create both tabular and image-based synthetic data generation. However, the hyperparameters are case-specific to create relevant and efficient data. The proposed study used different values for data generation, which are shown in Table 3.

There are many parameters that can be called to optimize the learning as much as we want. To check the performance of the GAN model, the root mean square error (RMSE) is used; others could also be used depending upon the case study. The adversarial model parameters, such as estimators, are set to 100, where the batch size is used to pick up the chunk of instances at once while training, and it is set to 500. The patience rate is set to 25 which is used when RMSE is not improving during the GAN model training. After 25 times with no improvement in RMSE, the training is stopped, as the saturated stage of model learning is reached. The learning environment is Python 3.9 with required libraries, where TensorFlow is used as a backend environment.

Table 3. Hyperparameters and their values used in GAN for data generation.

| Serial Number | Parameter            | Value    |
|---------------|----------------------|----------|
| 1             | Generator time       | 1.1      |
| 2             | Bot filter quantile  | 0.0001   |
| 3             | Top filter quantile  | 0.99     |
| 4             | Loss                 | RMSE     |
| 5             | Maximum depth        | 2        |
| 6             | Maximum bin          | 100      |
| 7             | Learning rate        | 0.001    |
| 8             | Random state         | Yes      |
| 9             | Estimators           | 100      |
| 10            | Batch size           | 500      |
| 11            | Patience             | 25       |

4.3. Experiment 1: Original UNSW-NB15 Dataset

The proposed study first applied features selection using the chi-square method and then fed the 16 selected features for binary and multi-class classification separately. The six classifiers were employed with an 80:20 holdout validation split. The training-data-based model was then tested, where the training and tested times were also estimated to obtain the time efficiency for all of the models’ performance results. The accuracy, precision, recall, and F1-score were calculated. The detailed results are shown in Table 4.

In Table 4, the classifier-wise performance of multi-class and binary-class data is shown. As we can see, different classifiers showed different behavior on both types of classification. If we look at binary classification, we can see that the best results in terms of accuracy are achieved by random forest classifier (95.59% accuracy), where the recall, precision, F1-score, geometric mean (G-mean) and area under cover (AUC) for this model are also greater than all of the other classifiers having 95.59%, 95.60%, 95.60%, 0.953 and 0.953 scores, respectively. The extra trees also achieved the best scores but here, we can say it value remains at second place among all classifiers. The other classifiers showed scores greater than 90% in all reported metrics.

If we look at the multi-class classification results, random forest showed the best results with 83.36% accuracy, 83.36% recall, 83.31% precision, 82.35% F1-score, 0.90 G-mean
and 0.89 AUC. The second best is still extra trees with 83.16% accuracy and recall. If we look at logistic regression that is basically used as a binomial classifier, it showed appropriate results in the binary classifier but failed in multi-class classification, as it showed 65.47% accuracy. However, we need to see the performance of classifiers in terms of time efficiency as well in the table. The time taken to train and to predict could be another metric to measure the performance of a classifier. However, DL took more time to train, as we can see in Table 4 that MLP took the highest time to train among all classifiers in both binary and multi-class classification at more than two minutes. The other classifiers, KNN and logistic regression, took less time in training but took more time in testing or prediction. The best performing classifiers random forest, and extra trees, showed 9 and 7.2 seconds in total for binary classification, where in multi-class classification, they took 11.5 seconds and 8.3 seconds in total. Therefore, as compared to classifiers other than the decision tree, the performed classifiers are also time-efficient for both binary and multi-class classification.

Table 4. Classification results on UNSW-NB15 dataset using chi-square features selection.

| Classification | Method     | Accuracy | Recall  | Precision | F1-Score | G-Mean | AUC   | Time(sec) Train + Test = Total |
|----------------|------------|----------|---------|-----------|----------|--------|------|-----------------------------|
| Binary         | MLP        | 92.61%   | 92.61%  | 92.74%    | 92.64%   | 0.928  | 0.928 | 126.6 + 0.1 = 126.7         |
|                | KNN        | 93.36%   | 93.36%  | 93.36%    | 93.36%   | 0.928  | 0.928 | 0.1 + 13.4 = 13.5           |
|                | Logistic Regression | 93.36%   | 93.36%  | 93.36%    | 93.36%   | 0.84   | 0.85  | 0.1 + 13.4 = 13.5           |
|                | Decision Tree | 93.97%   | 93.97%  | 93.97%    | 93.97%   | 0.935  | 0.935 | 3.2 + 0.0 = 3.2             |
|                | Random Forest | 95.59%   | 95.59%  | 95.60%    | 95.60%   | 0.953  | 0.953 | 8.8 + 0.2 = 9.0             |
|                | Extra Trees | 95.35%   | 95.35%  | 95.36%    | 95.36%   | 0.950  | 0.950 | 7.0 + 0.2 = 7.2             |
| Multi-class    | MLP        | 78.63%   | 78.63%  | 75.73%    | 75.36%   | 0.86   | 0.91  | 149.2 + 0.1 = 149.3         |
|                | KNN        | 78.41%   | 78.41%  | 79.40%    | 78.82%   | 0.87   | 0.78  | 0.1 + 14.7 = 14.8           |
|                | Logistic Regression | 65.47%   | 65.47%  | 61.53%    | 62.59%   | 0.77   | 0.78  | 15.2 + 0.0 = 15.2           |
|                | Decision Tree | 81.38%   | 81.38%  | 81.18%    | 80.90%   | 0.88   | 0.80  | 5.3 + 0.0 = 5.4             |
|                | Random Forest | 83.36%   | 83.36%  | 83.31%    | 82.35%   | 0.90   | 0.89  | 10.9 + 0.6 = 11.5           |
|                | Extra Trees | 83.16%   | 83.16%  | 82.91%    | 82.35%   | 0.89   | 0.88  | 7.7 + 0.6 = 8.3             |

4.4. Experiment 2: GAN-Based Dataset

The second dataset contains the GAN-based instances only for attacked categories and the normal instances taken from the original dataset. Then the contingency of new instances is proved, and the results are reported for binary and multi-class, as shown in Table 5.

Table 5 explores the score achieved by the six classifiers on GAN-based created instances and the normal class instances taken from the original dataset. However, the performance with these generated instances is improved as compared to the original ones. There is no significant difference, but it is improved, as we can see that for binary classification, the random forest again showed the highest accuracy of 95.41%, recall was the same, and the precision and F1-score were also highest with 95.44% and 95.42% with 0.954 G-Mean, AUC scores. The multi-class classification also showed the highest accuracy of 84.53% against random forest classifiers, and the nearer or second score classifier is the same.

However, in the binary class results, the results this time go down against logistic regression, where the performance of other classifiers is improved. Further, the multi-class classification results were the worst in the case of the original dataset against logistic regression, and in this experiment, worst with 68.62% accuracy. Now let us check again the time efficiency and the training and testing times of experiment 2.
The time efficiency could be seen such that in a few classifiers, training took time, and in a few, testing took more time. However, as proven previously, MLP took more total time as compared to other classifiers. The best classifiers are still the same here: random forest and extra trees took 8 and 5.9 seconds, respectively, in binary classification. However, in multi-class classification again the MLP is the worst, and the best ones took 12.3 and 9.6 total time. The training and testing times were greater as compared to the Experiment 1 in terms of multi-class classification, where in binary classification, Experiment 2 took less time.

### Table 5. Classification results on GAN dataset using chi-square features selection.

| Classification | Method   | Accuracy | Recall  | Precision | F1-Score | G-Mean | AUC  | Time(sec) | Train + Test = Total |
|----------------|----------|----------|---------|-----------|----------|--------|------|-----------|----------------------|
| Binary         | MLP      | 91.61%   | 91.61%  | 91.78%    | 91.55%   | 0.923  | 0.92 | 70.0      | 0.0 + 0.0 = 70.0    |
|                | KNN      | 93.03%   | 93.03%  | 93.05%    | 93.04%   | 0.930  | 0.929| 0.1 + 11.2 = 11.3 |
|                | Logistic Regression | 86.56% | 86.56% | 87.12%   | 86.34%   | 0.847  | 0.852| 2.9 + 0.0 = 3.0 |
|                | Decision Tree | 93.80% | 93.80%  | 93.80%    | 93.80%   | 0.937  | 0.937| 2.5 + 0.0 = 2.5 |
|                | Random Forest | 95.41% | 95.41%  | 95.44%    | 95.42%   | 0.954  | 0.954| 7.9 + 0.2 = 8.0 |
|                | Extra Trees | 95.25% | 95.25%  | 95.27%    | 95.25%   | 0.953  | 0.952| 5.7 + 0.2 = 5.9 |
| Multi-class    | MLP      | 81.02%   | 81.02%  | 79.29%    | 77.87%   | 0.87   | 0.91 | 123.2 + 0.1 = 123.3 |
|                | KNN      | 79.95%   | 79.95%  | 80.62%    | 80.21%   | 0.87   | 0.766| 0.1 + 15.1 = 15.3 |
|                | Logistic Regression | 68.62% | 68.62%  | 62.22%    | 64.27%   | 0.77   | 0.78 | 12.1 + 0.0 = 12.2 |
|                | Decision Tree | 82.39% | 82.39%  | 82.14%    | 82.04%   | 0.888  | 0.807| 9.6 + 0.0 = 9.6  |
|                | Random Forest | 84.53% | 84.53%  | 83.84%    | 83.58%   | 0.90   | 0.88 | 11.8 + 0.5 = 12.3 |
|                | Extra Trees | 84.36% | 84.36%  | 83.70%    | 83.59%   | 0.90   | 0.88 | 9.0 + 0.5 = 9.6   |

### 4.5. Experiment 3: Original UNSW-NB15 + GAN Dataset

The original and GAN-based generated instances from attacked categories were combined to give a new dataset with normal class data of original data. To check the validity upon solving the class imbalance ratio to some extent, the proposed study-based dataset was fed to the same six ML classifiers, and the results for binary and multi-class were calculated. The results are shown in Table 6.

Table 6 first shows the binary and then the multi-class classification results. The classification results reported for logistic regression were the worst results among all other classifiers, as was also the case in previous experiments. However, the binary classification on this new dataset achieved 98.14% highest accuracy using the extra trees methods after applying the chi-square method of feature selection. In Experiments 1 and 2, we see that random forest remains the highest score achiever for both binary and multi-class classification.

However, in Experiment 3, random forest is second with respect to the six classifiers of the binary category. If we look at recall, precision and F1-scores, then extra trees is also first with 98.14% recall, precision, F1-score, and 0.976 G-mean and AUC scores. Further, all classifiers remain greater than 95% except logistic regression which is less than or nearer to 90%. The confusion-matrix-based instance-based true positive, false positive, true negative, and false negative are also shown in Figure 4.

If we look in the MLP confusion matrix, it shows that in the 0 class, there are 23,064 predictions which correctly predicted where 2996 were predicted to be wrong and 56,568 were predicted as right in Class 1, where only 1190 were wrongly predicted. This was an MLP classifier only that did not perform as the best classifier among all. The best performing, random forest and extra trees need to be discussed here. Random forest predicted 1146 as being wrong predictions from Class 0, and 490 wrongly predicted from Class 1.
Table 6. Classification results on GAN + original dataset using chi-square feature selection.

| Classification | Method       | Accuracy | Recall | Precision | F1-score | G-mean | AUC   | Time(sec) Train + Test = Total |
|----------------|--------------|----------|--------|-----------|----------|--------|-------|-------------------------------|
| Binary         | MLP          | 95.00%   | 95.00% | 95.00%    | 94.95%   | 0.920  | 0.922 | 216.2 + 0.1 = 216.3           |
|                | KNN          | 95.84%   | 95.84% | 95.83%    | 95.84%   | 0.950  | 0.949 | 0.4 + 60.0 = 60.4             |
|                | Logistic Regression | 89.39%   | 89.39% | 89.35%    | 89.14%   | 0.849  | 0.855 | 8.2 + 0.0 = 8.2               |
|                | Decision Tree | 97.68%   | 97.68% | 97.68%    | 97.68%   | 0.971  | 0.972 | 9.6 + 0.1 = 9.7               |
|                | Random Forest | 98.05%   | 98.05% | 98.05%    | 98.04%   | 0.974  | 0.973 | 30.0 + 0.5 = 30.5             |
|                | Extra Trees  | 98.14%   | 98.14% | 98.14%    | 98.14%   | 0.976  | 0.976 | 23.0 + 1.2 = 24.2             |
| Multi-class    | MLP          | 80.25%   | 80.25% | 78.86%    | 77.50%   | 0.88   | 0.92  | 335.5 + 0.1 = 335.6           |
|                | KNN          | 80.69%   | 80.69% | 81.81%    | 81.20%   | 0.88   | 0.83  | 0.2 + 35.3 = 35.5             |
|                | Logistic Regression | 67.38%   | 67.38% | 64.50%    | 64.64%   | 0.78   | 0.74  | 22.9 + 0.0 = 22.9             |
|                | Decision Tree | 86.80%   | 86.80% | 87.04%    | 86.24%   | 0.919  | 0.90  | 6.4 + 0.1 = 6.5               |
|                | Random Forest | 87.38%   | 87.38% | 87.84%    | 86.67%   | 0.923  | 0.939 | 20.2 + 1.0 = 21.2             |
|                | Extra Trees  | 87.44%   | 87.44% | 87.81%    | 86.79%   | 0.923  | 0.94  | 15.7 + 1.0 = 16.7             |

Figure 4. Confusion matrices of binary classification on original + GAN-based new dataset.

The wrong predictions did not improve much as compared to MLP for Class 0, but for Class 1, they were reduced to 490. Extra trees reduced the 0 and 1 Classes’ wrong predictions, as 983 were predicted to be wrong for Class 0 and 574 for Class 1. The Class 1 predictions increased here as compared to the Random forest classifier but the Class 0 wrong predictions were also reduced. Therefore, in this way, the extra trees showed a slightly higher score in binary classification as compared to the random forest method.
Now, if we look at multi-class classification, the Extra trees is again the highest score achiever, where 87.44% recall, 87.81% precision, 86.79% F1-score, 0.923 G-mean, and 0.94 AUC scores are achieved. The Second place was taken by random forest again in binary classification with scores of 87.38% accuracy, and recall, 87.84% precision, 86.67% F1-score, 0.923 G-mean and roundly 0.94 AUC.

Logistic regression again failed here to classify the multi-class instances, with 67.38% accuracy and recall, 64.50% precision, 64.64% F1-score, 0.78 G-mean, and 0.74 AUC. However, MLP and KNN remain above 80%, where three others showed more than 86% results. The results showed improvement compared to Experiments 1 and 2 for both binary and multi-class classification. It shows that the class imbalance not only reduces the classification accuracy, but also compromises to achieve a score with a greater number of samples as a normal category in it. The confusion matrices are shown in Figures 5 and 6 regarding multi-class classification to check each classifier’s performance by instance.

The confusion matrices of random forest and extra trees need to be discussed here as the best performers. The behavior of predictions in random forest remains good for all classes, but those for 3, 4, 5, 6, and 9, are better compared to other classes. The correctly predicted instances are 3866, 7207, 20,000, 4062, and 24,915 for the 3, 4, 5, 6 and 9 class numbers, respectively. These numbers of rightly predicted instances are greater than those of other classes. These class are DoS, exploits, fuzzers, generic and normal classes, where other classes are predicted to a good extent as well. To compare Extra trees with it, we will again look at these predicted instances.

![Figure 5. Confusion matrices of multi-class classification on original + GAN-based new dataset (first three classifiers).](image)

In the case of extra trees, these numbers remain at 3779, 7114, 20,000, 4046, and 25,044 for DoS, exploits, fuzzers, generic, and normal classes. Although, as compared to the random forest results of these five classes, the number of predictions is less for DoS, exploits, and generic classes, where normal class instances are higher as compared to random forest. The rightly predicted normal class instances were 24,915 in random forest, whereas for extra trees, it were 25,044 that contain 100 more right predictions. We can say that with slight differences in the performance of random forest and extra trees classifiers, the normal class right predictions play an incremental role. However, in order to classify the attacked categories, the random forest classifier is good as compared to extra trees. Time efficiency is also discussed here as shown in the table for all classifiers among binary and multi-classification tasks. The worst method with respect to the total time is again MLP, with 216.3 seconds for binary and 335.6 seconds for multi-class classification tasks. However, the best performers are random forest and extra trees which showed 30.5
and 24.2 seconds in binary classification, but this time increased to 21.2 and 16.7 seconds in multi-class classification tasks. Logistic regression and decision trees took less time compared to random forest and extra trees, which was also observed in Experiments 1 and 2, but the best performer also did not take much time. We can see that only one or less than one second are the prediction time once they are trained on binary and multi-class classifications. Therefore, the best performer is also time-efficient while performing predictions.

Figure 6. Confusion matrices of multi-class classification on original + GAN-based new dataset (second 3 classifiers).

5. Comparison

The proposed study uses the UNSW-NB15 dataset by applying binary and multi-class classification as the original and oversampled datasets. Therefore, we compared it with binary and multi-class studies that used original or oversampled datasets. The comparative summary is shown in Table 7.

Table 7. Comparative analysis of UNSW-NB15 dataset with respect to recently applied state-of-the-art studies.

| Study               | Year | Methods                                                                 | Dataset       | Results                                                                 |
|---------------------|------|-------------------------------------------------------------------------|---------------|-------------------------------------------------------------------------|
| [29]                | 2018 | Two-stage approach for network intruder detection using SMOTE          | UNSW-NB15     | Multi-class classification Accuracy = 85.78%                           |
|                     |      |                                                                        |               |                                                                           |
| [25]                | 2020 | Focal-loss-based DNN and CNN                                            | UNSW-NB15     | DNN Binary-F1 = 90.41%, Multi-F1 = 39.78%, CNN Binary-F1 = 86.03%, Multi-F1 = 39.52% |
| [30]                | 2021 | Image-based network intrusion detection and DL-based classification     | UNSW-NB15     | ML-Net Binary Micro-Accuracy = 92.87%, Multi-class Micro-Accuracy = 72.31% |
| [26]                | 2022 | Modified cross entropy applied for class imbalance and neural network applied for classification | UNSW-NB15     | Accuracy = 90.76%                                                      |
| Proposed study      | 2022 | GAN-based class balancing and chi-square feature selection-based ML classification | UNSW-NB15     | Binary Accuracy = 98.14%, Precision = 98.14%, F1-score = 98.14%, G-Mean = 0.976, AUC = 0.976 |
| Proposed study      | 2022 | GAN based class balancing and chi-square feature selection-based ML classification | UNSW-NB15     | Multi-class Accuracy = 87.44%, Precision = 87.81%, F1-score = 86.79%, G-Mean = 0.923, AUC = 0.94 |
The first comparison [29] includes SMOTE in its proposed two-step approach for network intruder detection and showed that the multi-class 10 category classification approach reached up to 85.78%. The second comparison [25] study was discussed in the related work section in detail, where the achieved results by this study showed a maximum of 90.41% binary class F1-score and 39.78% F1-score on multi-class classification using the DNN method. The third study [30] used the image-conversion-based approach using the DL approach to classification on UNSW-NB15 and other datasets, where oversampling is used to increase the instances. The applied methods achieved the highest macro accuracies: 92.87%, in binary and, 72.31%, multi-class classification.

The fourth comparison used only the binary class to classify and achieved 90.76% accuracy by solving the class unbalancing problem using the modified cross entropy function. We compared here both binary and multi-class classification samples to increase the validity of our proposed framework. However, by looking at comparative studies, some of them proposed class imbalance problem solutions and some applied oversampling techniques; none reached the proposed framework results of binary and multi-class results regarding accuracy, precision, or F1-score metrics. It proved the outperformance of the proposed study on the recently applied state-of-the-art approaches.

6. Conclusions

The proposed study uses the UNSW-NB15 dataset to solve binary and multi-class classification problems. The dataset contains a minority class imbalance that was solved by many recent studies, using different methods and achieved significant results. However, the proposed framework not only solved the minority class imbalance problem using the GAN-based model to generate new instances from the UNSW-NB15 dataset, but achieved improved classification scores as compared to previous studies. Upon solving class imbalance, the proposed study collected three main datasets and performed three experiments on them. In experimentation, the newly generated data-based dataset outperformed the original dataset performance score when the chi-square method of features selection and six ML classifiers were used. However, the comparative analysis showed that the highest achieved results by the proposed framework are better than those of previous studies. The proposed framework suggests using the GAN method to get more appropriate hyperparameter optimization to solve class imbalance problems. It could be applied on other domains as well. More than one of the feature selection methods could be applied to check if better performance could be achieved.

Funding: There is no external funding received for this research.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Dataset is publicly available at: https://www.kaggle.com/datasets/mrwellsdavid/unsw-nb15 (accessed on 5 October 2022)

Acknowledgments: This research was supported by the Researchers Supporting Project (RSP2022R476), King Saud University, Riyadh, Saudi Arabia.

Conflicts of Interest: The author declares no conflict of interest.

References
1. Folino, G.; Sabatino, P. Ensemble based collaborative and distributed intrusion detection systems: A survey. J. Netw. Comput. Appl. 2016, 66, 1–16. [CrossRef]
2. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: Techniques, datasets and challenges. Cybersecurity 2019, 2, 1–22. [CrossRef]
3. Bayerl, P.S.; Karlović, R.; Akhgar, B.; Markarian, G. Community Policing—A European Perspective; Springer: Berlin/Heidelberg, Germany, 2017.
4. Li, J.; Qu, Y.; Chao, F.; Shum, H.P.; Ho, E.S.; Yang, L. Machine learning algorithms for network intrusion detection. AI Cybersecur. 2019, 151–179.
