


Low-sample classification in NIDS using the EC-GAN method


Marko Zekan

(Faculty of Organization and Informatics, University of Zagreb, Varaždin, Croatia
marko.zekan@gmail.com)

Igor Tomičić

(Faculty of Organization and Informatics, University of Zagreb, Varaždin, Croatia
 <https://orcid.org/0000-0002-8626-9507>, igor.tomicic@foi.hr)

Markus Schatten

(Faculty of Organization and Informatics, University of Zagreb, Varaždin, Croatia
 <https://orcid.org/0000-0001-6910-8675>, markus.schatten@foi.hr)

Abstract: Numerous advanced methods have been applied throughout the years for the use in Network Intrusion Detection Systems (NIDS). Among these are various Deep Learning models, which have shown great success for attack classification. Nevertheless, false positive rate and detection rate of these systems remains a concern. This is mostly because of the low-sample, imbalanced nature of realistic datasets, which make models challenging to train.

Considering this, we applied a novel semi-supervised EC-GAN method for network flow classification of CIC-IDS-2017 dataset. EC-GAN uses synthetic data to aid the training of a supervised classifier on low-sample data. To achieve this, we modified the original EC-GAN to work with tabular data. In our approach, WCGAN-GP is used for synthetic tabular data generation, while a simple deep neural network is used for classification. The conditional nature of WCGAN-GP diminishes the class imbalance problem, while GAN itself solves the low-sample problem. This approach was successful in generating believable synthetic data, which was consequently used for training and testing the EC-GAN.

To obtain our results, we trained a classifier on progressively smaller versions of the CIC-DIS-2017 dataset, first via a novel EC-GAN method and then in the conventional way, without the help of synthetic data. We then compared these two sets of results with another author's results using accuracy, false positive rate, detection rate and macro F1 score as metrics. Our results showed that supervised classifier trained with EC-GAN can achieve significant results even when trained on as little as 25% of the original imbalanced dataset.

Keywords: cybersecurity, network security, GAN, NIDS, synthetic tabular data, classification, semi-supervised learning, Wasserstein GAN

Categories: C.2.3, I.2.1, K.6.5

DOI: 10.3897/jucs.85703

1 Introduction

The success of Network Intrusion Detection Systems (NIDS) that rely on machine/deep learning to detect anomalies depends largely on the data on which the models are trained. Data for deep supervised model training must be abundant, high-quality, labeled, and well balanced. Problems arise because such datasets are very difficult and expensive to create [Azeez et al., 2020].

With that in mind, the network traffic data that is available for training the NIDS classifiers is often sparse, low quality, imbalanced or just insufficient for successful training of a Deep Learning (DL) model. Additional problem relates specifically to the field of NIDS system, where a low false positive rate is a must. The goal of every Intrusion Detection System (IDS) is to have as few false alarms as possible, meaning that they recognize malicious packets only when they do occur. A lot of false positives can cause real attacks to go unnoticed or for an attack to be overlooked by security operators [Ho et al., 2012].

The main motivation of this paper stems from these observations, and the following research question was derived to indicate the purpose of this work and to drive the research: Is there a classification method available that could be utilised within the field of network intrusion detection systems that could show competitive results compared to other existing methods, but on the significantly smaller data-set?

This paper seeks to solve the problem of using small realistic datasets to train classifiers to be used in NIDS systems, by using the novel External Classifier Generative Adversarial Networks (EC-GAN) method [Haque, 2021]. This semi-supervised learning method was recently introduced as a solution for training a fully supervised classifier by using combination of real and synthetic data for the purpose of model training on a small, realistic data set [Haque, 2021].

In this paper we present a modified version of EC-GAN model that seeks to aid the classification of small tabular datasets by employing the Wasserstein Conditional Generative Adversarial Network with Gradient Penalty (WCGAN-GP) for synthetic tabular data generation. We conducted an experiment in which we trained two identical Deep Neural Network (DNN) classifiers on multiple versions of the CIC-IDS-2017 dataset of varying sizes. The first DNN was trained as part of the EC-GAN model with the aid of synthetic data and the other in a conventional way, without synthetic data. Finally, the results were evaluated and compared to other results in this field. We showed that classifier trained as part of the EC-GAN model can achieve results similar to other results in this field by using as much as 75% less data to do so.

Thus, the main novelty and contribution of this paper is the modified EC-GAN method that uses WCGAN-GP for creating synthetic tabular data that is then used by a simple DNN for supervised classification in the field of network intrusion detection systems, with final goal of improving network traffic in Network Intrusion Detection Systems.

2 Related Work

The use of artificial intelligence (AI) techniques in NIDS systems has achieved rapid growth during the past decade. Ahmad et al. as well as Lio and Lang in their review papers [Ahmad et al., 2021, Liu and Lang, 2019] mention various Machine Learning (ML) and Deep Learning methods that are used for network attack classification. Models like Autoencoders (AE), Deep Neural Networks, Decision Trees (DT), Naive Bayes and Logistic Regression are just some of the algorithms that have been applied to this problem.

Another type of AI model that has gained popularity in NIDS systems in recent years is Generative Adversarial Networks (GAN). Because GANs are generative models, their role in network attack classification is to supplement low-sample data sets with artificially generated data in order to improve classification results.

One such example is the AE-CGAN-RF presented by Lee and Park [Lee and Park, 2019] in which they use an Autoencoder network for dimension reduction and feature extraction, conditional GAN (CGAN) for class oversampling to help combat performance degradation caused by data imbalance, and the Random Forest (RF) algorithm for classification. The authors demonstrated that their proposed model outperforms the RF model trained on data over sampled using the SMOTE technique as well as the standalone RF algorithm in terms of precision, recall, and F1 score. Another example of GANs in NIDS is the SynGAN framework developed by Charlier, Singh, Ormazabal, State and Schulzrinne [Charlier et al., 2019]. The authors successfully generated believable Distributed Denial of Service (DDoS) attack data based on the NSL-KDD and CIC-IDS-2017 datasets using the Wasserstein GAN with Gradient Penalty (WGAN-GP) for synthetic data generation.

While classifiers trained with the help of GANs show promising results, those trained without the use of synthetic data should not be overlooked. Toupas et al. [Toupas et al., 2019] presented one such model in which they used a DNN with 8 hidden layers, uniform initialization on input and output layers and SMOTEENN oversampling method on the training set to achieve outstanding results.

Botnet attack detection problem within the field of industrial Internet of Things was tackled in [Alharbi et al., 2021], where authors used Local-Global best Bat Algorithm for Neural Networks (LGBA-NN) in order to "select both feature subsets and hyper-parameters for efficient detection of botnet attacks". Authors have tested the proposed LGBA-NN algorithm on an N-BaIoT data set and showed promising results over other recent approaches such as weight optimization using Particle Swarm Optimization and BA-NN. The approach did not tackle the problem of small data sets, however.

Authors in [Woźniak et al., 2020] argue that the trained classifier efficiency increases with the amount of information in the dataset, employing the RNN-LSTM (recurrent neural network Long short-term memory) classifier with the use of the NAdam optimization algorithm and presenting the results of 99.9957% efficiency in numerical experiments.

A modified bio-inspired meta-heuristic algorithm - Grey Wolf Optimization algorithm (GWO) - was presented in [Alzaqebah et al., 2022] and argued that it enhances the efficacy of the intrusion detection systems in detecting both normal and anomalous network traffic. Authors have used UNSWNB-15 dataset with the Extreme Learning Machine (ELM), and used the modified GWO to tune the ELM's parameters. The primary goal of their paper was detection of generic attacks in network traffic, mainly because of the restrictions of the dataset attack types. Among other promising results presented within the paper which show improved performances over existing methods, the method has also apparently minimized the crossover error rate and false positive rate to less than 30%.

A research field of image recognition was fused into the field of NIDS in [Toldinas et al., 2021], where authors propose an approach using multistage deep learning image recognition, evaluating the approach on UNSW-NB15 (showing 99.8% accuracy in the detection of the generic attack) and BOUN DDoS datasets (showing 99.7% accuracy in the detection of the DDoS attack and 99.7% accuracy in the detection of the normal traffic).

Although the last few overviewed research papers show promising results on the performance metrics within the NIDS field, they do not explicitly tackle the problem of the sparse datasets, which is the main research problem of this paper.

3 External Classifier GAN (EC-GAN)

The Generative Adversarial Network with an External Classifier, abbreviated as EC-GAN, is a semi-supervised model that combines an unsupervised GAN with a supervised classifier to aid in the classification of low sample datasets. In the original paper [Haque, 2021], the authors explain that they propose EC-GAN as a way to improve classification utilizing GAN-generated images in restricted, fully supervised regimes. To achieve this, the authors used a three-network model consisting of a generator, a discriminator, and a classifier.

This three-network GAN architecture is not unique and can be found in models such as Triple-GAN [Li et al., 2017], where authors added a third network to the GAN, a classifier, claiming that overall network performance suffers when the discriminator is given the task of both discrimination and classification. The EC-GAN differs from the Triple-GAN in a fundamental way, author explains, as Triple-GAN uses a classifier to supplement GAN training while EC-GAN uses GAN to supplement classifier training [Haque, 2021].

The author explains their motivation by stating that there are many datasets with an abundance of unlabeled data but there are also datasets where even unlabeled data is difficult to come by. They use the acquisition of chest x-ray data as an example of a slow process that requires multiple trained experts to obtain a single data point, stating that such scenarios result in low-sample datasets that limit the effectiveness of deep-learning algorithms. They claim that datasets like these can benefit from any new data, including artificial data, and states that their method aims to improve classification in these specific tasks [Haque, 2021].

The architecture of the EC-GAN model is depicted in Figure 1. The EC-GAN model works in a way so that in every training iteration, the generator constructs fake images from the random input vector of normalized values, much like in the Vanilla GAN implementation. Generated images are then forwarded to the discriminator which tries to discern whether the image is from the real dataset or was faked by the generator, learning in the process to distinguish false from real images. In the same training iteration, the classifier weights are updated by feeding it both real data from the sparse dataset and fake images from the generator. The fake images have been assigned labels by pseudo-labeling process which is necessary as artificially generated images have no labels, and labels are required for the training of a supervised classifier. The generator and the discriminator have their weights updated the same way they would have in the conventional GAN implementation.

Given how the generator improves at creating fake images over time, it is natural for it to produce lower-quality images during the initial training stages. This could pose a problem for classifier training, seeing how it is trained on artificial data form the start of the training process. To control the influence of pseudo-labeled data on the effectiveness of the model the author introduces additional hyperparameters λ and t . The adversarial weight λ , is a hyperparameter introduced to decrease the weight (influence) of fake data on classifier training. Without this parameter, real and fake data would have an equal impact on classifier weights update in a given training iteration, which is undesirable. The second hyperparameter, confidence threshold (t), represents the minimum level of confidence that the classifier must produce for the pseudo-labeling process to take place. That is, if the largest value from the classifier *Softmax* output is lower than the confidence threshold, then this prediction will not be used in pseudo-labeling process.

The author proposed an intuitive new loss function that takes into account the simultaneous training of the classifier on two sets of data by utilizing newly introduced

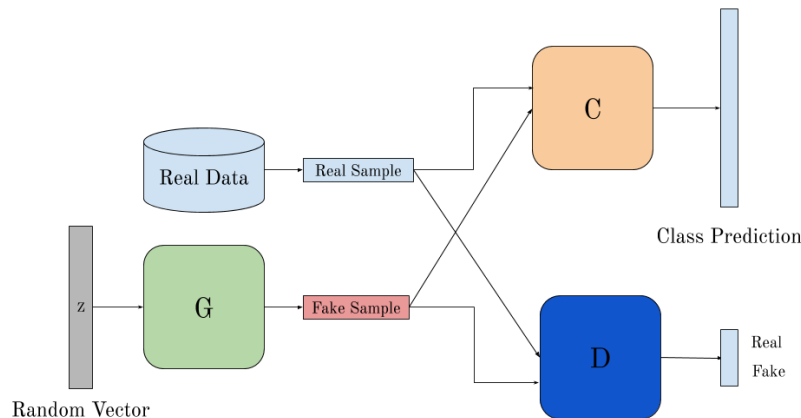


Figure 1: Original External Classifier GAN architecture [Haque, 2021]

hyperparameters. This loss function is what differentiates the EC-GAN from other GAN implementations in addition to the external classifier architecture.

$$L_C(x, y, z) = CE(C(x), y) + \lambda CE(C(G(z)), \operatorname{argmax}(C(G(z))) > t)$$

In the loss function shown above, the x is the real data, y is the real data label, z is the fake data, λ is the adversarial weight, CE is the cross-entropy function, $C(x)$ is the result of the real data classification, $C(G(z))$ is the result of the fake data classification, and finally t represents the confidence threshold. The classifier loss function is actually the sum of the conventional cross-entropy (CE) functions calculated for real and fake data. By combining the two CE functions the classifier weights are updated on both fake and real data in single training iteration.

4 Proposed solution

As mentioned in the previous section, the original EC-GAN implementation focused on its use in image classification. For this purpose, the original author used ResNet18 convolutional neural network (CNN) as the classifier and the Deep Convolutional GAN (DCGAN) to generate artificial images. Both of these algorithms are based on CNNs and specialize in image processing. Seeing how our goal is to aid network traffic classification, some modifications needed to be made to the original EC-GAN. Network traffic data is tabular data, where each row represents a captured data point described by several attributes (columns).

To use the EC-GAN for network traffic classification, we had to replace the original paper's algorithms with ones more suitable to work with tabular data. Our guiding thought here was that, when boiled down, the EC-GAN is just any GAN with an external classifier that uses the new loss function for updating classifier weights. Having said that, we propose the use of a GAN that is better suited for synthetic tabular data generation together with a classifier that is better suited for tabular data. For this purpose, we decided

to use the WCGAN-GP as our GAN and a Deep Neural Network (DNN) with three hidden layers as our classifier.

4.1 WCGAN-GP

The creation of synthetic tabular data is still relatively unexplored area of application for GANs, with few examples of successful implementations, particularly in the field of NIDS systems. However, several existing papers that have succeeded in this task have shown good results. One such paper was presented by Walia, Tierney and McKeever in 2020 [Walia et al., 2020] in which they used WCGAN-GP to create synthetic tabular data. The authors compared their GAN-generated data sets to those generated using traditional oversampling methods like SMOTE. The results show that the data generated by WCGAN-GP are comparable to real-world data and perform significantly better than data generated by the SMOTE technique [Walia et al., 2020]. Based on the successes of Walia, Tierney and McKeever, as well as the authors of the SynGAN paper [Charlier et al., 2019], we decided to use the WCGAN-GP as an algorithm for synthetic tabular data generation in this research. As possible alternative to WCGAN-GP we considered TGAN [Xu and Veeramachaneni, 2018] and [Xu et al., 2019], which are two well-known solutions for tabular data generation.

Conditional Wasserstein GAN with gradient penalty (WCGAN-GP) is an improved version of the original WGAN developed by [Arjovsky et al., 2017]. The Wasserstein GAN with gradient penalty (WGAN-GP) was developed by [Gulrajani et al., 2017] and solves the problem of training stability that the original WGAN was suffering from. Since then, WGAN-GP was further upgraded, made into a conditional WGAN-GP [Yu et al., 2019] and used by other authors for, among other things, creation of synthetic tabular data [Charlier et al., 2019, Walia et al., 2020].

As mentioned, WCGAN-GP falls under a special type of GAN called - conditional GAN (CGAN). The distinctive feature of CGAN is that they allow targeted generation of data by providing the generator with class labels alongside random input vector. This is important because it opens a possibility to command the CGAN generator which data class to create from a random vector instead of it generating data classes at random [Mirza and Osindero, 2014]. Furthermore, Wasserstein in the name of this algorithm denotes Wasserstein's loss function that is used by this model's discriminator instead of the usual Minimax loss function [Arjovsky et al., 2017]. In the original paper, Arjovsky, Chintala, and Bottu [Arjovsky et al., 2017] state that using this algorithm achieves training stability and solves the problem of mode collapse which plagued earlier GAN models, as well as making it easier to find errors and optimal hyperparameters.

The architecture of this algorithm can be seen in Figure 2. It is important to note that in the Wasserstein GAN literature, the model discriminator is referred to as a critic instead of a discriminator, but still performs the same role. Figure 2 shows the building blocks of the WCGAN algorithm, that is fairly similar in structure to the vanilla GAN implementation, with few notable differences.

The training process can also be compared to the one of vanilla GAN, the main differences being that WCGAN-GP generator takes two inputs instead of one, the critic uses Wasserstein loss function instead of the conventional Minimax function and an additional hyperparameter n_critic is introduced.

The two generator inputs are the random noise vector Z and a class label L .

Based on these, the generator creates a batch of synthetic data which is then judged by a critic for its validity. The critic is fed batches of real data and fake data in an alternating fashion giving a binary output as a result. Important thing to mention is that

the WCGAN-GP critic is trained n_critic times more than the generator. Meaning that if n_critic is set to equal 5, the critic will be trained five times in this training iteration and the generator just once. In a revised WGAN paper [Gulrajani et al., 2017], authors state that the algorithm gives better results if the critic is trained several times more than the generator.

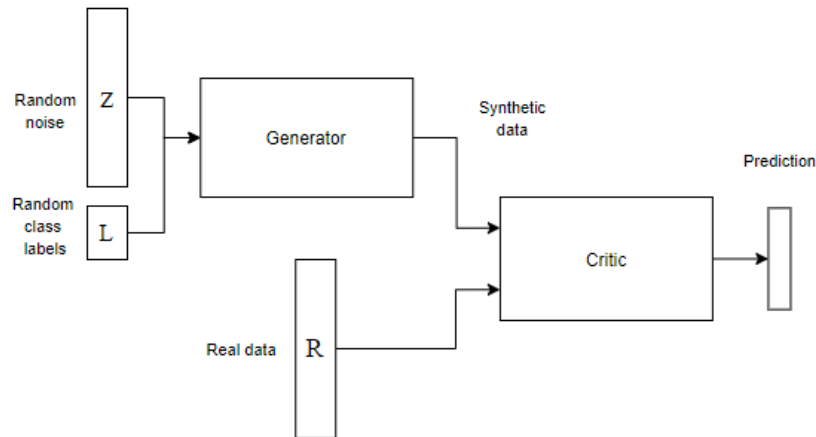


Figure 2: Conditional Wasserstein GAN with gradient penalty (WCGAN-GP) architecture

4.2 Modified EC-GAN

We propose a modification of the original EC-GAN algorithm for the task of aiding classification in low-sample tabular datasets. Our proposed model uses WCGAN-GP for generating believable synthetic tabular data and a simple DNN for the classification task. Proposed models' architecture can be seen in Figure 3. The overall setup is similar as in the original EC-GAN with classifier being fed real data and synthetic data generated by WCGAN-GP.

The training process for the modified EC-GAN starts the same as the training of the WCGAN-GP itself, by feeding the random noise vector and random class label into the generator. After the critic is trained, the batch of synthetic data is passed to the classifier, as well as the batch of real data. The classifier is then trained on both real and fake data using the new EC-GAN loss function to update its weights.

By using a conditional GAN for creating synthetic data we overcame the need for pseudo-labelling process that was used in the original paper to determine the label of newly generated image. Our method uses the same label for WCGAN-GP input and classifier training. Our reasoning is that since the WCGAN-GP is expected to generate data based on a combination of input noise and input label, we can use that label later as ground truth in classifier training. This way when training the classifier on fake data, we pass it synthetic data made by WCGAN-GP and measure its success against the same label that was used as a seed to create that data in the step before.

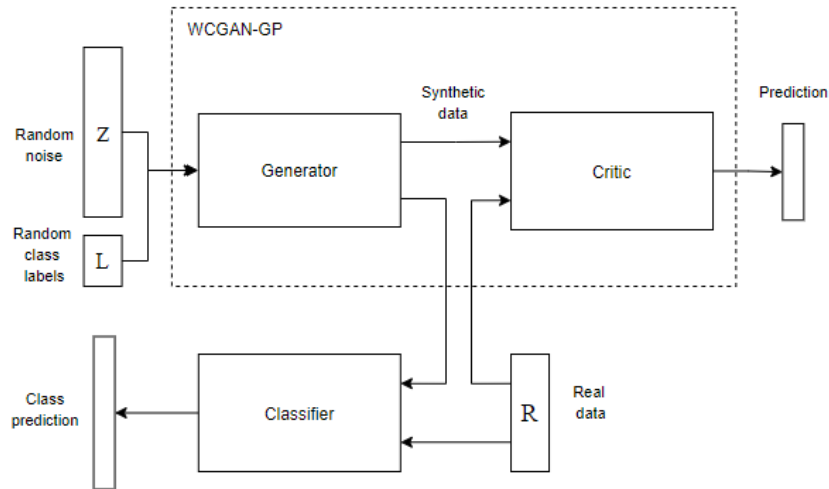


Figure 3: A proposed EC-GAN architecture

Table 1 shows the architecture and hyperparameter values for the WCGAN-GP implementation of this research. The generator and the critic are both deep neural networks made in such a way so that the critic's number of hidden-layer neurons is decreasing, while the generator's is increasing, as it was done by Walia et al. in [Walia et al., 2020]. The n_{critic} hyperparameter mentioned before is set to 5, this value is taken from the improved WGAN paper [Gulrajani et al., 2017].

Architecture of the DNN classifier used in our EC-GAN implementation can be seen in Table 2. There are a few reasons why we decided to use the DNN as the classifier for our EC-GAN model. The first reason is the excellent results obtained by Toupas et al. [Toupas et al., 2019] in their work involving the application of DNN to the NIDS problem. Second reason is that the DNN models are easy to train, especially when they are not too deep. Model's architecture is quite simple, it consists of three hidden layers where the number of neurons is increasing towards the middle and decreasing towards the output. The *ReLU* function was used for hidden layer activation while the *Softmax* function was used as an output layer activation function.

Most of the hyperparameters found in these tables were obtained by extensive experimentation and by trial and error until the optimal value was found. Hyperparameter values specific to the EC-GAN method, the adversarial weight and the confidence threshold remain the same as stated in the original paper [Haque, 2021] (0.1 and 0.2 respectively). During the research phase, different values of these parameters were tested, but the best results were still obtained using the original values.

5 Methodology

In this section, we used our proposed modified EC-GAN¹ to train the classifier on the CIC-IDS-2017 dataset and then evaluated our results against another same-size DNN

¹ Code available at https://github.com/marzekan/EC-GAN_NIDS

Critic	Generator
Input: Dataset dimension	Input: 32 (random noise)
1024, Leaky ReLU (alpha=0.2)	256, Leaky ReLU (alpha=0.2), Dropout(0.3)
512, Leaky ReLU (alpha=0.2)	512, Leaky ReLU (alpha=0.2), Dropout(0.3)
256, Leaky ReLU (alpha=0.2)	1024, Leaky ReLU (alpha=0.2), Dropout(0.3)
Output - 1, Linear activation	Output - Dataset dimension, Linear activation
Other hyperparameters:	
Learning rate: 0.0005	
Optimizer: Adam (beta_1=0.05, beta_2=0.9)	
Batch size: 128	
Epochs: 10	
n_critic: 5	

Table 1: Architecture and hyperparameters of WCGAN-GP

Architecture
Input: Dataset dimension
128, ReLU, Dropout(0.3)
256, ReLU, Dropout(0.3)
128, ReLU, Dropout(0.3)
Output: 15 - class count, Softmax
Optimizer: Adamax

Table 2: Architecture and hyperparameters of DNN model

trained without the aid of synthetic data, as well as against the results obtained by Toupas et al. [Toupas et al., 2019].

We implemented our modified EC-GAN in Python 3.8 using the TensorFlow 2.5 and Keras API and the model was trained on NVIDIA GeForce RTX 2070 graphics card.

5.1 Dataset analysis and pre-processing

We trained our model on the CIC-IDS-2017 dataset [CIC, 2017], which is often used in this field for training supervised classifiers. Beside CIC-IDS-2017 there are other

datasets used for this purpose such as KDD CUP 99 and NSL-KDD but these were mostly replaced by CIC-IDS-2017 in more recent papers. In recent years the newer CIC-IDS-2018 dataset has been developed which is an improvement over the CIC-IDS-2017. Nevertheless, we decided to use the older CIC-IDS-2017 because of its widespread use in existing literature as well as to show that the EC-GAN model can be trained on imbalanced and low-sample tabular data.

CIC-IDS-2017 named after the Canadian Institute for Cybersecurity, was developed with the goal of creating complete, modern set of data in the field of intrusion detection systems [CIC, 2017]. The data set consists of network traffic data aggregated over the period of one working week during which 14 different attacks were simulated. The set also contains a base neutral class called BENIGN, which represents benign, or normal traffic, during which no attack occurs. The malicious traffic and benign traffic together produce 15 different data classes. Seeing how all of the data is assigned into classes, this data set is a labeled data set meant for supervised learning tasks.

From Table 3 it is clear that the largest class - BENIGN, makes as much as 80% of all the data. While the smallest class - Heartbleed, makes up less than one-thousandth of a percent of the entire set. This imbalance in class sizes makes this dataset suitable for testing the EC-GAN model seeing how it was conceived to deal with low-sample, imbalanced, realistic datasets.

	Count	Ratio (%)
BENIGN	2271320	80.31
DoS_Hulk	230124	8.13
PortScan	158804	5.61
DDoS	128025	4.52
DoS_GoldenEye	10293	0.36
FTPPatator	7935	0.28
SSHPatator	5896	0.20
DoS_slowloris	5796	0.20
DoS_Slowhttpstest	5499	0.19
Bot	1956	0.06
Web_Attack_Brute_Force	1507	0.05
Web_Attack_XSS	652	0.02
Infiltration	36	0.001
Web_Attack_Sql_Injection	21	0.0007
Heartbleed	11	0.0003

Table 3: Class sizes and their ratio in CIC-IDS-2017

To make our data suitable for training we needed to perform some pre-processing steps, the flowchart of which can be found in Figure 4. Our process begins with data

cleaning, where we removed rows containing missing values, as well as NULL and infinite values. When put together, these instances account for 1358 rows which makes up for less than 0.05% of total data. This figure being so small, we decided that it was acceptable to remove these rows from the data, rather than dealing with them in some other way.

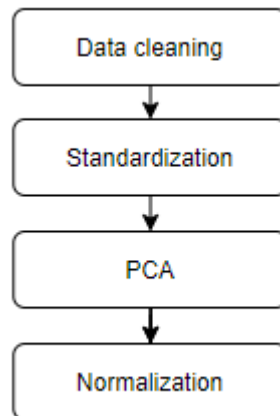


Figure 4: Data preprocessing with PCA-based feature reduction workflow

After the data was cleaned, the next step was to pre-process the data further in preparation for neural network training. Second step after data cleaning was to standardize the data, so that that it would have a mean of 0 and a standard deviation from the mean of 1. This step is done in preparation for the next step which is Principal Component Analysis (PCA). By performing PCA, the feature count was reduced from the original 78, not counting the label, to 31. This reduction in data dimensionality should greatly decrease the training complexity. Final step of pre-processing was data normalization which scales the data to a fixed range usually from 0 to 1. Data normalization was necessary because PCA produces data which is not on the scale of 0 to 1 and such data is beneficial for neural network training. These specific training steps were selected after a trial-and-error period where different combinations of pre-processing steps were tested until satisfactory results were obtained. Data processed this way was used for training both WCGAN-GP and DNN.

5.2 Experiment Design

This section details the process we used to obtain our results, the flowchart of which can be seen in Figure 5. First step was data pre-processing which was discussed in the previous section, after that the data needed to further be prepared for conducting the experiment by creating smaller versions of the original dataset. The idea behind this experiment is to train EC-GAN and a standalone classifier on same sets of progressively smaller version of CIC-IDS-2017 datasets and compare the evaluation results in the end. By doing this we hoped to show how a conventional standalone classifier acts when

trained on smaller and smaller datasets, versus how the same classifier trained with the help of synthetic data acts in the same situation. Synthetic data, in this case, was created as the part of the EC-GAN model.

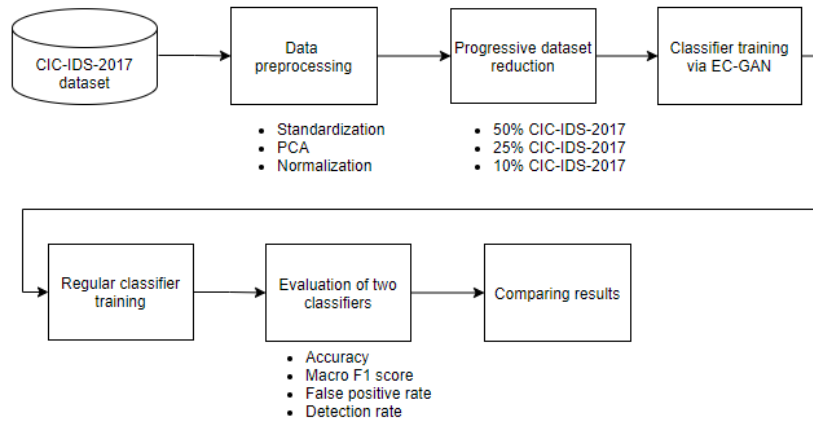


Figure 5: Flow chart of the conducted experiment on downsizing the CIC-IDS-2017 dataset

Downsizing of the original dataset was done as shown in Figure 6. The goal was to create four versions of the CIC-IDS-2017 dataset, each being progressively smaller than the previous one. It was decided these versions would contain 100%, 50%, 25% and 10% of the original dataset. The downsizing method we used was to take a ration of each class and downsize it by randomly deleting the percentage of data point from it. This means that in the dataset version that represents 50% of the dataset, each class was downsized to 50% of the original size, so that the ratio between labels would stay the same. This method ensures that each version of the dataset contains all data classes. There was one exception that we had to make in our downsizing method and that is to not downsize the classes that have less than 2000 members. This accounts for five classes (*Bot*, *Web Attack Brute Force*, *Web Attack XSS*, *Infiltration*, *Web Attack SQL Injection*, *Heartbleed*) that were not downsized because they already had a very low sample count. The number of 2000 was chosen arbitrarily as the limit for the downsizing method and was based on subjective number of samples that each class in the dataset had. By not downsizing the low-member classes we preserved all of their members for training, but we also disturbed the class ratios which could affect the classification results. We believe that this action would not affect the results, seeing how low-member classes still have a minuscule ratio when compared to other more sizeable classes in the dataset.

Next step in our experiment was to train the EC-GAN classifier and the standalone classifier on all four versions of the dataset. When all models completed training, we made an evaluation of each one of them and compared their results across four important metrics: accuracy, detection rate, false positive rate and F1 measure. These metrics are, among others, the standard for testing hypotheses in statistics and are often used to evaluate classification algorithms [Tharwat, 2021].

The evaluation of each one of the models was performed on test data (*testing set*),

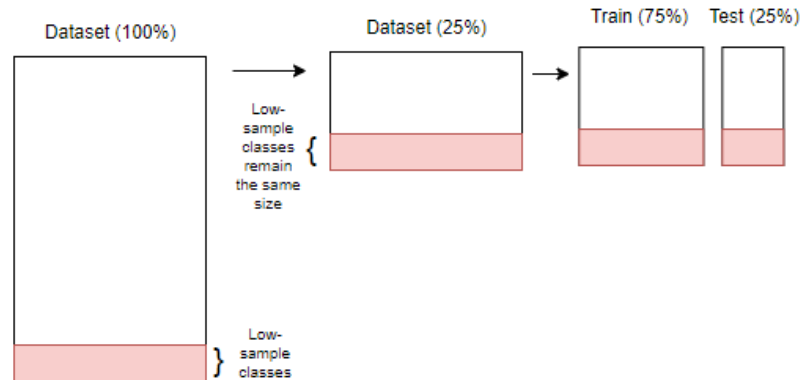


Figure 6: Data set reduction workflow (downsizing of the original dataset)

which makes up 25% of the total set used for training and testing. It is important to emphasize that the test data are also supplemented with synthetic data, but only after the training itself. The reason for this was the very low number of samples in some classes of the training set. Unlike the training process, where synthetic data is generated in parallel with classifier training, synthetic data for model evaluation was generated before testing. To create synthetic data we used the same WCGAN-GP generator that was used in EC-GAN model training.

For testing purposes, ten thousand instances of each class except the BENIGN class were created, because it already makes up over 80% of the data set. The synthetically generated data were then mixed with the real test set and an evaluation was performed. Figure 7 provides an illustration for the better understanding of how synthetic data was used in models' evaluation.

5.3 Results

Because of the imbalanced nature of this dataset, it was important to evaluate the model's performance by additional metrics alongside accuracy, such as the F1 score. Table 4 depicts the evaluation results for each class through the metrics for classifier evaluation - precision, recall and F1-score.

Table 5 presents the evaluation results of the EC-GAN classifier and a standalone classifier as well as the results presented by Toupas et al. [Toupas et al., 2019]. The EC-GAN classifier achieved better results across all metrics than the conventionally trained classifier without the aid of synthetic data. Best results were achieved by EC-GAN classifier model that was trained on 25% of the original dataset. This fact proves the benefit of the EC-GAN model for classification in situations where not much data is available. The model with highest accuracy of all is the one presented by Toupas et al., reaching 99.95% accuracy. Their results also show a remarkably low false alarm rate, that the EC-GAN classifier also managed to reach. The metrics in which the EC-GAN classifier leads are F1 measure and detection rate. The results of this work are successful and show that using the EC-GAN method, a supervised classifier can be trained even when the original data set is very small.

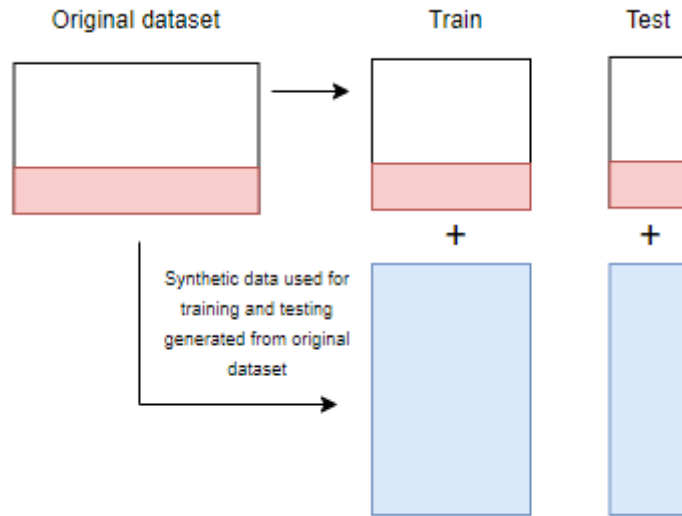


Figure 7: Supplementing training and testing sets with synthetic data

	Precision	Recall	F1-score
BENIGN	.9903	.9868	.9885
Bot	.9935	.9823	.9879
DDoS	.9987	.9981	.9984
DoS_GoldenEye	.9971	.9971	.9971
DoS_Hulk	.9759	.9937	.9847
DoS_Slowhttpstest	.9912	.9988	.9950
DoS_slowloris	.9971	.9973	.9972
FTPPatator	.9986	.9993	.9990
Heartbleed	1.000	1.000	1.000
Infiltration	1.000	.9988	.9994
PortScan	.9487	.9750	.9617
SSHPatator	.9947	.9986	.9966
Web_Attack_Brute_Force	.9993	.9666	.9827
Web_Attack_Sql_Injection	1.000	.9993	.9996
Web_Attack_XSS	1.000	.9851	.9920

Table 4: Evaluation results showing precision, recall and F1 score for each dataset class

The evaluation of the work was performed on a set of test data (testing set), which makes up 25% of the total set used for training and testing of models. It is important to emphasize that the test data are also supplemented with synthetic data, but only after the training itself. The reason for this is the same as with model training - some classes simply number too few members to be viable for training, let alone training and testing purposes. If we take into account that some classes have only a dozen members, most of whom (75%) are used to train models, it is clear why it is necessary to supplement

the testing set. Unlike the training process, where synthetic data is generated in parallel with classifier training, in the testing process synthetic data is generated before testing. To create synthetic data, same generator from the WCGAN-GP model that was used in EC-GAN model training was used.

	DNN				DNN			DNN
	(usual method)				(EC-GAN)			[Toupas et al., 2019]
(%) CIC-IDS-2017	10%	25%	50%	100%	10%	25%	50%	100%
Accuracy	.9771	.9806	.9844	.9852	.9897	.9895	.9876	.9995
Macro F1 score	.7266	.7247	.6351	.6289	.9924	.9995	.9901	.9410
False positive rate	.0004	.0001	.0006	.0002	.0007	.0005	.0010	.0005
Detection rate	.9255	.9414	.8985	.9218	.9933	.9897	.9852	.9562

Table 5: Evaluation results of two classifiers (EC-GAN and usual method) compared with Toupas et. al. [Toupas et al., 2019]

6 Conclusion

In this paper we presented our modified version of the novel semi-supervised EC-GAN model that is used for aiding classification of low-sample supervised tabular datasets. The EC-GAN model developed in this paper was made with the goal of improving network traffic in Network Intrusion Detection Systems (NIDS). Our main contribution is the presented modified EC-GAN that uses WCGAN-GP for creating synthetic tabular data that is then used by a simple DNN for supervised classification. We have chosen GAN as our starting point since similar methods like SMOTE [Chawla et al., 2002] have been shown to be less effective (see [Lee and Park, 2021] for example).

We conducted an experiment which involves four versions of the original CIC-IDS-2017 dataset, each being progressively smaller than the other. We then trained the EC-GAN model and a standalone DNN classifier on all four versions of the dataset to see how each of them would act when trained on smaller and smaller datasets. In the end we presented our results by comparing the two classifiers with results obtained by another paper using a DNN classifier for NIDS classification. The results have shown that the EC-GAN classifier trained on the 25% of the original dataset outperforms other classifiers in the F1 metric with the value of 0.9995, while matching the best false positive rate of .0005. We have shown that the EC-GAN classifier achieves better results when trained on small datasets than the conventional DNN classifier, and that it matches the results obtained by other authors in this field. Like any other DL model, our proposed methodology is still affected by the input data, meaning that it is limited by the same issues that it is trying to solve. Another limitation is that the introduction of GAN to classifier training has increased the training complexity, more so since GANs are harder and more expensive to train than the average standalone classifier model. In future work we believe it would be beneficial to try out other GANs for creating synthetic tabular data in the same EC-GAN scenario described in this paper, as well as to try and use other classifier architectures.

7 Acknowledgment

This work has been supported in full by the Croatian Science Foundation under the project number IP-2019-04-5824.

References

- [Ahmad et al., 2021] Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., and Ahmad, F. (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, 32(1):e4150. Available: Wiley Online Library, <https://onlinelibrary.wiley.com/>. [accessed: 13.06.2021.].
- [Alharbi et al., 2021] Alharbi, A., Alosaimi, W., Alyami, H., Rauf, H. T., and Damaševičius, R. (2021). Botnet attack detection using local global best bat algorithm for industrial internet of things. *Electronics*, 10(11):1341.
- [Alzaqebah et al., 2022] Alzaqebah, A., Aljarah, I., Al-Kadi, O., and Damaševičius, R. (2022). A modified grey wolf optimization algorithm for an intrusion detection system. *Mathematics*, 10(6):999.
- [Arjovsky et al., 2017] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein gan. Available: arXiv, <https://arxiv.org/>. [accessed: 16.07.2021.].
- [Azeez et al., 2020] Azeez, N., Bada, T., Misra, S., Adewumi, A., Vyver, C., and Ahuja, R. (2020). *Intrusion Detection and Prevention Systems: An Updated Review*, pages 685–696. Available: ResearchGate, <https://www.researchgate.net/>. [accessed: 06.09.2021.].
- [Charlier et al., 2019] Charlier, J., Singh, A., Ormazabal, G., State, R., and Schulzrinne, H. (2019). Syngan: Towards generating synthetic network attacks using gans. *CoRR*, abs/1908.09899.
- [Chawla et al., 2002] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.
- [CIC, 2017] CIC (2017). Intrusion detection evaluation dataset (cic-ids2017).
- [Gulrajani et al., 2017] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. *CoRR*, abs/1704.00028. Available: arXiv, <https://arxiv.org/>. [accessed: 16.07.2021.].
- [Haque, 2021] Haque, A. (2021). Ec-gan: Low-sample classification using semi-supervised algorithms and gans. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(18):15797–15798. Available: arXiv, <https://arxiv.org/>. [accessed: 10.06.2021.].
- [Ho et al., 2012] Ho, C.-Y., Lin, Y., Lai, Y.-C., Chen, I.-W., Wang, F.-Y., and Tai, W.-H. (2012). False positives and negatives from real traffic with intrusion detection/prevention systems. *International Journal of Future Computer and Communication*, 1:87–90. Available: ResearchGate, <https://www.researchgate.net/>. [accessed: 09.09.2021.].
- [Lee and Park, 2019] Lee, J. and Park, K. (2019). AE-CGAN Model based High Performance Network Intrusion Detection System. *Applied Sciences*, 9(20):4221.
- [Lee and Park, 2021] Lee, J. and Park, K. (2021). Gan-based imbalanced data intrusion detection system. *Personal and Ubiquitous Computing*, 25(1):121–128.
- [Li et al., 2017] Li, C., Xu, K., Zhu, J., and Zhang, B. (2017). Triple generative adversarial nets.
- [Liu and Lang, 2019] Liu, H. and Lang, B. (2019). Machine learning and deep learning methods for intrusion detection systems: A survey. *Applied Sciences*, 9(20). Available: MDPI, <https://www.mdpi.com/>. [accessed: 13.06.2021.].

- [Mirza and Osindero, 2014] Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *CoRR*, abs/1411.1784. Available: IEEE Xplore, <https://ieeexplore.ieee.org/>. [accessed: 13.07.2021.].
- [Tharwat, 2021] Tharwat, A. (2021). Classification assessment methods. *Applied Computing and Informatics*, 17(1):168–192. Available: Emerald Insight, <https://www.emerald.com/insight>. [accessed: 09.09.2021.].
- [Toldinas et al., 2021] Toldinas, J., Venčkauskas, A., Damaševičius, R., Grigaliūnas, Š., Morkevičius, N., and Baranauskas, E. (2021). A novel approach for network intrusion detection using multistage deep learning image recognition. *Electronics*, 10(15):1854.
- [Toupas et al., 2019] Toupas, P., Chamou, D., Giannoutakis, K. M., Drosou, A., and Tzovaras, D. (2019). An intrusion detection system for multi-class classification based on deep neural networks. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 1253–1258.
- [Walia et al., 2020] Walia, M., Tierney, B., and McKeever, S. (2020). Synthesising tabular data using wasserstein conditional gans with gradient penalty (wrgan-gp). Available: ResearchGate, <https://www.researchgate.net/>. [accessed: 16.07.2021.].
- [Woźniak et al., 2020] Woźniak, M., Siłka, J., Wiczorek, M., and Alrashoud, M. (2020). Recurrent neural network model for iot and networking malware threat detection. *IEEE Transactions on Industrial Informatics*, 17(8):5583–5594.
- [Xu et al., 2019] Xu, L., Skoularidou, M., Cuesta-Infante, A., and Veeramachaneni, K. (2019). Modeling tabular data using conditional GAN. *CoRR*, abs/1907.00503. Available: arXiv, <https://arxiv.org/>. [accessed: 09.07.2021.].
- [Xu and Veeramachaneni, 2018] Xu, L. and Veeramachaneni, K. (2018). Synthesizing tabular data using generative adversarial networks. *CoRR*, abs/1811.11264. Available: arXiv, <https://arxiv.org/>. [accessed: 23.06.2021.].
- [Yu et al., 2019] Yu, Y., Tang, B., Lin, R., Han, S., Tang, T., and Chen, M. (2019). Cwgan: Conditional wasserstein generative adversarial nets for fault data generation. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2713–2718. Available: IEEE Xplore, <https://ieeexplore.ieee.org/>. [accessed: 10.09.2021.].