Prospects for Generative - Adversarial Networks in Network Traffic Classification Tasks

G D Asyaev¹

¹Lecturer, Department of system programming, FSAEIHE SUSU (NRU), 76, Lenin prospect 454080, Chelyabinsk, Russia

E-mail: asiaevgd@susu.ru

Abstract. The paper presents an approach that allows increasing the training sample and reducing class imbalance for traffic classification problems. The basic principles and architecture of generative adversarial networks are considered. The mathematical model of network traffic classification is described. The training sample taken to solve the problem has been analyzed. The data preprocessing is carried out and justified. An architecture of the generative-adversarial network is constructed and an algorithm for generating new features is developed. Machine learning models for traffic classification problem were considered and built: Logistic regression, k Nearest Neighbors, Decision tree, Random forest. A comparative analysis of the results of machine learning models without and with the generation of new features is conducted. The obtained results can be applied both in the tasks of network traffic classification, and in general cases of multiclass classification and exclusion of unbalanced features.

1. Introduction

Currently, there are many models for solving traffic classification problems [17]. These can be both conventional machine learning models (random forest, catboost, lightGBM), cluster models, and the use of deep learning models - neural networks. However, most tasks in the field of network traffic classification are trained on marked-up data [18]. It is worth noting that in order to get a good, diverse volumetric dataset requires enough time and resources, which in most cases is quite problematic [19]. The main idea of this study is to use generative-adversarial networks to generate a training sample. This will significantly reduce the time for data acquisition, its partitioning and preprocessing, and improve the quality metric for a sufficiently pronounced unbalanced sample.

Article I. General principles of generative-adversarial networks

1.1. Generative modeling

Let there be some finite number of features $X$ and target variables $Y$ in the training dataset. Let $z$ have some distribution:

$$z \sim N(0,1)$$

Then, in the generalized case, there exists a function

$$f: Z \rightarrow X \text{that } f(z) \sim P(X).$$
Without knowing $P(X)$ explicitly, the model obtains some general bases implicitly. In this way one can recover $f(z)$, by which entries of $P(X)$ can be obtained as:

$$f(z), z \sim N(0,1).$$ (3)

This fact served the development of generative modeling, which is a method of learning without a teacher, where there is an automated detection of various patterns and dependencies, the signs of which can be used to generate new data [7]. In this case, it is believed that the new features will be as similar as possible in their distribution to the existing data set.

A key feature of generative-adversarial networks (GAN) is the formation of a quality criterion in the form of a separate neural network [3].

1.2. Generative-adversarial network

It was with the development of generative modeling that generative-adversarial networks (fig. 1) emerged, which consist of two neural networks:

- **Generator.** This neural network is necessary for learning the generation of new features $X$. The generator must form the features in such a way, that the discriminator could not classify it as a fake feature [2]. The discriminator is a certain criterion of quality of the generator.

- **Discriminator.** Its main task is to determine the real signs $X$, from the fake $X\sim$. That is, when the discriminator model will input a true sign, its output will be some number (in the range from 0 to 1) labeled real, and when the model will input a generated sign, the output, respectively, will be some number labeled "fake" [1]. These numerical values can be interpreted as the degree of confidence of the discriminator that the input feature belongs to either a real class or a generated one (fig. 2). So, for example, when the input will be a real sign, the output of the discriminator class "real" should tend to 1.

\[loss_{dis_{real}} = -t_R \ast \log \log (real) - (1 - t_R) \ast \log (1 - real) \]
\[loss_{dis_{fake}} = -t_F \ast \log \log (fake) - (1 - t_F) \ast \log (1 - fake)\]

where $loss_{dis_{real}}$ - loss at the real sign, where $t_R=1$, a $loss_{dis_{fake}}$ - losses at the generated sign, where $t_F=0$. 

![Figure 1. General architecture of GAN.](image)
Substituting these values, we can write a generalized loss function [7] for the case in question:

\[ \text{loss}_{\text{dis}} = - \log \log (\text{real}) - \log (1 - \text{fake}) \]  

(4)

Thus, the quality index must be minimal for real features and maximal for generated ones [11].

\[ \text{loss}_{\text{dis}} = - \log \log (\text{fake}) \]  

(5)

Thus the generator and the discriminator are trained independently of each other [12].

**Article II. Creating a model**

To build the model, we took a dataset from the Kaggle website, "UNSW_NB15 by the IXIA PerfectStorm tool. Australian Centre for CyberSecurity (ACCS). Unprocessed network packets from the UNSW-NB 15 dataset were created using the IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) to create a hybrid of real contemporary normal activity and synthetic contemporary attack behavior. The training sample size is 82332 records (fig. 3).

![Figure 3. Set of traits of the training sample.](image)

The training dataset contains 44 features and 1 target variable attack_cat. It takes on the following values (fig. 4):

- 'Normal',
- 'Reconnaissance',
- 'Backdoor',
- 'DoS',
- 'Exploits',
- 'Analysis',
- 'Fuzzers',
- 'Worms',
- 'Generated attribute',
- 'Output',
- 'Input Signal',
- 'Generator',
- 'Discriminator',
- 'Figure 2. Generator learning mode.'
Based on the distribution of the target variable, the sample is quite unbalanced. Attacks like 'Analysis', 'Fuzzers', 'Worms', 'Shellcode', 'Generic' have less than 700 entries. It is for these types of attacks that it is reasonable to apply a generative adversarial network to generate new records in order to eliminate class imbalance [19].

It is worth noting that records where the number of missing values exceeded 40% were replaced with median values. Since there were categorical features in the training set, and obviously they carry useful information, they were converted to numerical ones using the One Hot Encoding method. The choice of the method was due to the fact that it does not as sign weight to any of the features when training the model:

![Distribution of the target variable](image)

**Figure 4.** Distribution of the target variable.

The data have been normalized using StadartScaler, and the unique number of each row has been eliminated due to un informativeness.

The data were divided into a test sample and a training sample in a 70/30 ratio. It is worth noting that each class of the target variable fell into the training sample.

The following models were chosen for training:
- Logistic Regression
- K-nearest Neighbors
- Decision tree
- Random Forest (with gini criterion)
- Random Forest (with entropy criterion).
1.3. Logistic regression
The first model chosen for training was a simple logistic regression to determine how well the data had been preprocessed and cleaned, and to look at the importance ranking table of the features. The regularization coefficient was 0.1, the number of epochs was chosen to be 40, and the cross validation value was 10.

| Feature    | Importance |
|------------|------------|
| 7          | 7.0        |
| 181        | 3.0        |
| 190        | 3.0        |
| 159        | 2.0        |
| 24         | 2.0        |

**Figure 5.** Quality indicators of the model.

It is worth noting that even without fine-tuning, the simple model was about 90% accurate (fig. 5). And the average precision-recall score was 82%. A table of the importance of the signs was constructed. Figure 6 shows the top 5 indicators.

1.4. K-nearest neighbors
This method refers to the method of learning without a teacher. The number of nearest neighbors was 3.

| Accuracy: 76.74 |
|-----------------|
| Accuracy of CV: 66.94 |
| Execution time: 35.958560943603516 |

**Figure 7.** Indicators of model quality.

It is worth noting that the quality of the model on cross-validation was about 67%, which is slightly better than the naive Bayesian classifier (fig. 7).

1.5. Decision tree
The main goal of this algorithm is to create a model that predicts the value of the target variable by learning simple decision rules derived from data functions [14]. The tree can be viewed as a piecewise constant approximation.

| Accuracy: 93.74 |
|-----------------|
| Accuracy of CV: 92.94 |
| Execution time: 44.524121293516 |

**Figure 8.** Quality indicators for the model.

Fig. 9 shows a section of the constructed decision tree. You can clearly see that the built model has a small number of leaves.
Figure 9. A section of the constructed decision tree.

1.6. Random forest
Random Forest represents an ensemble of decision tree classifiers for different subsamples of a dataset and uses averaging to improve prediction accuracy and control over fitting [16]. The figure below (fig. 10) shows a comparative table of the results of the random forest model under different finetuning and data preprocessing parameters.

| PreProcessing               | Parameters                                      | Train Acc | Test Acc |
|-----------------------------|-------------------------------------------------|-----------|---------|
| OneHot encoding, StandardScaler | n_estimators 10, max_depth 10, max_features 10, class_weight (0:2, 1:1)  | 94.10     | 91.05   |
|                             | n_estimators 10, max_depth 10, max_features 20, class_weight (0:2, 1:1) | 94.16     | 91.61   |
|                             | n_estimators 10, max_depth 20, max_features 20, class_weight (0:2, 1:1) | 95.84     | 89.16   |
|                             | n_estimators 30, max_depth 10, max_features 30, class_weight (0:2, 1:1) | 94.19     | 91.66   |
|                             | n_estimators 10                              | 95.66     | 87.90   |
|                             | n_estimators 10, max_depth 20                | 95.73     | 86.91   |
|                             | n_estimators 10, max_features 20             | 95.76     | 87.57   |
|                             | n_estimators 50                              | 96.08     | 87.31   |
| OneHot encoding, MinMaxScaler | n_estimators 10, max_depth 10, max_features 30, class_weight (0:2, 1:1) | 93.88     | 91.49   |
|                             | n_estimators 10, max_depth 10, max_features 10, class_weight (0:2, 1:1) | 94.20     | 91.32   |
|                             | n_estimators 10, max_depth 20, max_features 30, class_weight (0:2, 1:1) | 94.08     | 91.90   |
|                             | n_estimators 10, max_depth 10, max_features 10, class_weight (0:2, 1:1) | 94.21     | 91.68   |
|                             | n_estimators 20, max_depth 10, max_features 10, class_weight (0:2, 1:1) | 94.21     | 91.36   |
|                             | n_estimators 10, max_depth 10, max_features 10, class_weight (0:2, 1:1) | 94.20     | 91.02   |

Figure 10. Comparative table of model parameters and its quality.

Separately, it should be noted that none of the models exceeded the accuracy value of 95%.

In order to increase such quality criteria as precision, recall, accuracy it was decided to use a generatively adversarial network for such target variables as 'Analysis', 'Fuzzers', 'Worms', 'Shellcode', 'Generic'. The choice of variables was due to the small number of records in the training sample [20].

Below is the algorithm for generating the generated records:

1. A sample of one of the target variables was generated one by one.
2. The record data were fed to the input of convolutional neural networks without additional grouping into batches (vector dimension one) [13].
3. The generated sequence was generated at the output of the neural network.
4. The random module was used to select either a real sequence of features from the dataset or a generated sequence from the generator (CNN).
5. The set of features selected at step 4 [2] was fed to the input of the discriminator.
6. A "real" / "fake" label for the sequence was formed at the output of the discriminator.
7. π. 1 - 6 was repeated until the discriminator began to confuse the real and generated set of features with a probability of 75% [4].

We used convolutional neural networks (CNN) as a generator [13]. Its architecture is shown in Figure 11.

![Figure 11. Generator architecture.](image)

The full-link neural network acted as a discriminator. It consists of 8 dense layers, lasso regularization (0.3) to reduce the probability of overtraining, and an output of dimension 1 (label real/fake).

![Figure 12. Discriminator architecture.](image)

The generated records were added to the dataset and the models considered were re-trained. The table below presents a comparative characterization of the models without the use of data preprocessing and with the use of artificially generated features for labels with less than 700 records using GAN.
Table 2. Comparative table of models.

| Model                  | ACC | Precision with GAN | Recall with GAN |
|------------------------|-----|---------------------|-----------------|
| Logistic Regression    | 90.74 | 93.3 | 86.4 | 89.4 | 80.4 | 90.1 |
| K Nearest Neighbours   | 76.74 | 82.2 | 70.1 | 77.3 | 65.2 | 80.1 |
| Decision Tree          | 93.74 | 97.7 | 88.6 | 92.1 | 83.1 | 95.3 |
| Random Forest          | 95.74 | 99.21 | 90.2 | 98.8 | 88.4 | 99.1 |

It is clearly seen that the best model was Random Forest, and the use of GAN to reduce the imbalance of classes allowed to determine the types of attacks with the value of the metric of quality precision was 99%. In addition, all models have increased the completeness of correct answers, which proves the applicability of artificial feature generation methods in practice in the task of traffic classification.

2. References

[1] Chen H and Jiang L 2019 Efficient GAN-based method for cyber-intrusion detection (ArXiv)
[2] Vinchurkar D P A R 2012 A Review of Intrusion Detection System Using Neural Network and Machine Learning Technique *International Journal of Engineering Science and Innovative Technology (IJESET)* Vol 1, Issue 2
[3] Yu F, Wang L, Fang X and Zhang Y 2020 The Defense of Adversarial Example with Conditional Generative Adversarial Networks *Security and Communication Networks* pp 1-12
[4] Tang W, Tan S, Li B and Huang J 2017 Automatic Steganographic Distortion Learning Using a Generative Adversarial Network *IEEE Signal Processing Letters* vol 24 10 pp 1547-1551
[5] Chhetri S R, Lopez A B, Wan J and Al Faruque M A 2019 GAN-Sec: Generative Adversarial Network Modeling for the Security Analysis of Cyber-Physical Production Systems in Proceedings of the 2019 Design, Automation and Test in Europe Conference and Exhibition, DATE 2019 *Institute of Electrical and Electronics Engineers Inc.*
[6] Mao X, Li Q, Xie H, Lau R Y, Wang Z and Smolley S P 2017 Least Squares Generative Adversarial Networks *Proceedings of the IEEE International Conference on Computer Vision* pp 2813-2821
[7] Yan Q, Wang M, Huang W, Luo X and Yu F R 2019 Automatically synthesizing DoS attack traces using generative adversarial networks *International Journal of Machine Learning and Cybernetics* vol 10 12 pp 3387-3396
[8] 2019 Development Obscuring and Analyzing Sensitive Information with Generative Adversarial Networks *World Wide Technology, Tech. Rep.*
[9] Wu C, Ju B, Wu Y, Xiong N N and Zhang S 2020 WGAN-E: A generative adversarial network for facial feature security *Electronics* (Switzerland) vol 9 3
[10] Dey S, Ye Q and Sampalli S 2019 A machine learning based intrusion detection scheme for data fusion in mobile clouds involving heterogeneous client networks *Information Fusion* 49 205-215
[11] Wang Y, Wang Y, Liu J, Huang Z and Xie P 2016 A survey of game theoretic methods for cybersecurity *In 2016 IEEE First International Conference on Data Science in Cyberspace (DSC)* pp 631-636
[12] Wu M, Song Z and Moon Y B 2019 Detecting cyber-physical attacks in CyberManufacturing systems with machine learning methods *Journal of Intelligent Manufacturing* 30(3) 1111-1123
[13] Abu-Nimeh S, Nappa D, Wang X and Nair S 2007 A Comparison of Machine Learning Techniques for Phishing Detection *APWG eCrime Researchers Summit* (Pittsburg)
[14] Revett K et al 2007 A machine learning approach to keystroke dynamics based user authentication 
International Journal of Electronic Security and Digital Forensics Vol 1 1

[15] Ford V & Siraj A 2014 Applications of Machine Learning in Cyber Security 27th International 
Conference on Computer Applications in Industry and Engineering (CAINE)

[16] Olaschinde Olayemi & Alese Boniface & Adetunmbi Adebayo 2019 Machine learning approach 
for information security International Journal of Information and Computer Security 16 91-101

[17] Ksenia N and Minbaleev A 2020 Legal Support of Cybersecurity in the Field of Application of 
Artificial Intelligence Technology 2020 International Conference Quality Management, 
Transport and Information Security, Information Technologies (IT&QM&IS) pp 59-62

[18] Nikolskaia K and Naumov V 2020 Ethical and Legal Principles of Publishing Open Source Dual- 
Purpose Machine Learning Algorithms 2020 International Conference Quality Management, 
Transport and Information Security, Information Technologies (IT&QM&IS) pp 56-58

[19] Nikolskaia K and Minbaleev A 2020 Legal Regulation of Incidents Related to DDoS Attacks 
2020 International Conference Quality Management, Transport and Information Security, 
Information Technologies (IT&QM&IS) pp 53-55

[20] Nikolskaia K and Minbaleev A 2020 New Perspectives on Ethics and the Laws of Artificial 
Intelligence in the Investigation of Incidents Related to DDoS Attacks 2020 International Multi- 
Conference on Industrial Engineering and Modern Technologies (FarEastCon) pp 1-5