UDC 004

A. Dremov, A. Volokyta

METHOD FOR MALICIOUS NETWORK TRAFFIC CATEGORISATION

Abstract. This paper aims to provide a solution for malicious network traffic detection and categorisation. In this paper we propose a semi-supervised GAN to train a discriminator model to categorise malicious traffic, as well as identify malicious and non-malicious traffic. The main goal is to achieve accurate categorisation of malicious traffic with few labelled examples.

Keywords: cybersecurity, network security, malicious traffic identification, machine learning, generational adversarial networks, semi-supervised learning.

Introduction/Relevance of the research topic

Computer networks are a key part of modern digital communications. However, these networks can be susceptible to malicious network traffic and various attacks. These attacks can be categorised by specific packet information used in these attacks. As such, network intrusion and attack detection play an important part in identifying an attack and counteracting it and are a relevant area of research.

Additionally, modern machine learning methods and algorithms can be used to categorise data or objects with great precision, provided a large enough training sample. However, rapid developments in security penetration create a problem, where new penetration methods appear frequently and gathering enough packet samples for model training becomes a difficult task. Therefore, the problem of training models with few initial samples remains relevant today. A combination of these areas is the main research area of this paper.

Problem Definition

The core problem that the research focuses on is the problem of malicious traffic identification and categorisation. First part of the problem is the identification of whether or not traffic is malicious in nature. Malicious traffic is one that can be used to attack the computer network and individual devices in the network and include malware, DoS attacks, network scanning, data exfiltration, R2L etc. Second part of this problem is categorisation of malicious traffic.

Actual scientific research and issue analysis

A number of researchers have tackled the problem of network attack classification [1][2] and the effect of malicious traffic on computer networks [3]. Of particular interest to this paper is the general approach to performing a network attack described in [1], as well as classification and effects described in [2] and [3] respectively.

Additionally, research into the intrusion detection and, more importantly, an analysis of malicious traffic packet contents [4][5][6] help connect network attacks to packet contents. This allows to define features used by the machine learning algorithm.

Lastly, research in the area of applying machine learning to solve network intrusion detection problem was performed [7], where a variety of models and algorithms are used. The research describes the architecture of semi-supervised GAN networks [8].

Uninvestigated parts of general matters defining

In author's opinion, the problem of intrusion detection using machine learning algorithms when there is insufficient data remains understudied. Additionally, proposed solutions may encounter difficulty with generalisation when being applied in different scenarios. A GAN based model could be used to achieve greater degree of generalisation.

Research objective

The purpose of this work is to research methods and models of malicious network traffic detection and categorisation with the usage of artificial intelligence models. Additionally, the purpose of the work is to create an AI model that can be used to detect classify malicious traffic with packet information.

Presentation of the main material

The dataset used in this research is NSL-KDD (https://www.kaggle.com/datasets/hassan06/nslkdd), which contains 125000 examples of network traffic packet data, as well as 22 categories based on attack type. Packets labelled "normal" indicate no attack. The features used in the classification include internet protocol used, service used, login status, login attempts, attempts to take root status, file and script creation, error rate, and other, for a total of 41 features. A total of 67000 records are labelled as non-malicious traffic and 58000 are labelled as malicious (fig. 1), (fig. 2), (fig. 3).

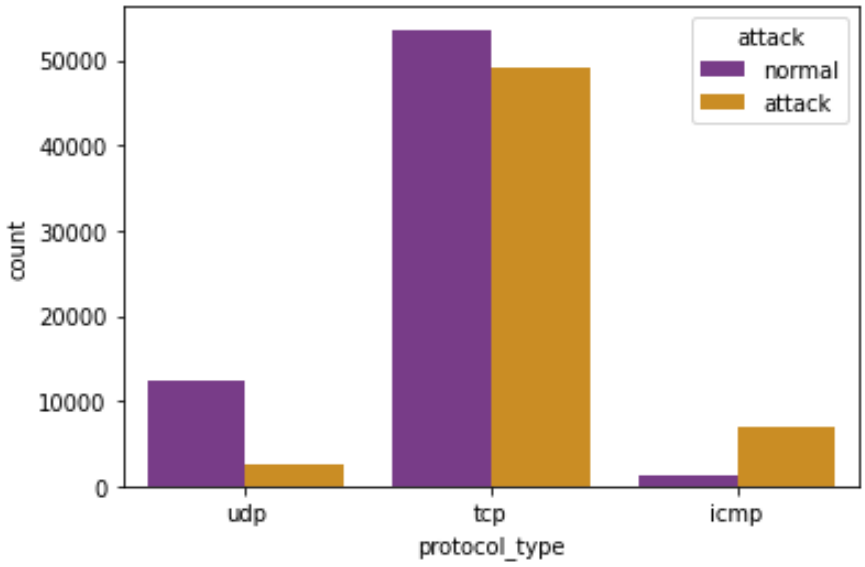

Fig. 1. Distribution of malicious and non-malicious traffic with regards to protocol used

```
RangeIndex: 125973 entries, 0 to 125972
Data columns (total 43 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   duration                     125973 non-null  int64
 1   protocol_type                125973 non-null  int64
 2   service                      125973 non-null  int64
 3   flag                         125973 non-null  int64
 4   src_bytes                    125973 non-null  int64
 5   dst_bytes                    125973 non-null  int64
 6   land                         125973 non-null  int64
 7   wrong_fragment               125973 non-null  int64
 8   urgent                       125973 non-null  int64
 9   hot                          125973 non-null  int64
 10  num_failed_logins            125973 non-null  int64
 11  logged_in                    125973 non-null  int64
 12  num_compromised              125973 non-null  int64
 13  root_shell                   125973 non-null  int64
 14  su_attempted                 125973 non-null  int64
 15  num_root                     125973 non-null  int64
 16  num_file_creations           125973 non-null  int64
 17  num_shells                   125973 non-null  int64
 18  num_access_files             125973 non-null  int64
 19  num_outbound_cmds            125973 non-null  int64
 20  is_host_login                125973 non-null  int64
 21  is_guest_login               125973 non-null  int64
 22  count                        125973 non-null  int64
 23  srv_count                    125973 non-null  int64
 24  serror_rate                  125973 non-null  float64
 25  srv_serror_rate              125973 non-null  float64
 26  rerror_rate                  125973 non-null  float64
 27  srv_rerror_rate              125973 non-null  float64
 28  same_srv_rate                125973 non-null  float64
 29  diff_srv_rate                125973 non-null  float64
 30  srv_diff_host_rate           125973 non-null  float64
 31  dst_host_count               125973 non-null  int64
 32  dst_host_srv_count           125973 non-null  int64
 33  dst_host_same_srv_rate       125973 non-null  float64
 34  dst_host_diff_srv_rate       125973 non-null  float64
 35  dst_host_same_src_port_rate  125973 non-null  float64
 36  dst_host_srv_diff_host_rate  125973 non-null  float64
 37  dst_host_serror_rate         125973 non-null  float64
 38  dst_host_srv_serror_rate     125973 non-null  float64
 39  dst_host_rerror_rate         125973 non-null  float64
 40  dst_host_srv_rerror_rate     125973 non-null  float64
 41  attack                       125973 non-null  int64
 42  level                        125973 non-null  int64
dtypes: float64(15), int64(28)
memory usage: 41.3 MB
```

Fig. 2. Dataset information

:

	duration	protocol_type	service	flag	src_bytes	dst_bytes
0	0	1	20	9	491	0
1	0	2	44	9	146	0
2	0	1	49	5	0	0
3	0	1	24	9	232	8153
4	0	1	24	9	199	420
...	...	...	...	...	...	...
125968	0	1	49	5	0	0
125969	8	2	49	9	105	145
125970	0	1	54	9	2231	384
125971	0	1	30	5	0	0
125972	0	1	20	9	151	0

Fig. 3. Example of values in dataset

The following data pre-processing was performed. The categorical values were converted to numerical values. The dataset was scaled using standard scaling, equation (1).

$$X' = \frac{x - \bar{x}}{\sigma} \tag{1}$$

Where x is the original feature vector, $\bar{x}$ is the mean of the feature vector, $\sigma$is standard deviation.

Lastly the labels were one-hot encoded for categorical classification.

For training we make use of a 70:30 split of training to test data.

As a baseline classifier, a simple deep network was implemented using tensorflow keras with two fully connected layers with 32 and 16 neurons, activation function is “relu”, batch normalisation layers and dropout layers to prevent overfitting (fig. 4). Final layer is a dense layer with “softmax” activation for categorical classification. Model metrics are “categorical_crossentropy” for loss function and “categorical_accuracy” for accuracy. The model was trained for 50 epochs on the dataset and achieved 99% accuracy, indicating possible overfitting (fig. 5). This classifier will be used to evaluate performance of the GAN-based classifier.

Second model is based on a generative adversarial network (GAN). These networks consist of a generator model and a classifier model. The generator uses gaussian distribution noise to generate fake information, equation (2).

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \quad (2)$$

Where μ is mean of the distribution, σ is standard deviation.

The classifier model of GAN is used to classify generator output as real or fake. For this a DNN with sigmoid activation is used. The result of the classification is used to calculate generator loss and discriminator loss (fig. 6). This allows to train the generator to create more believable fake data.

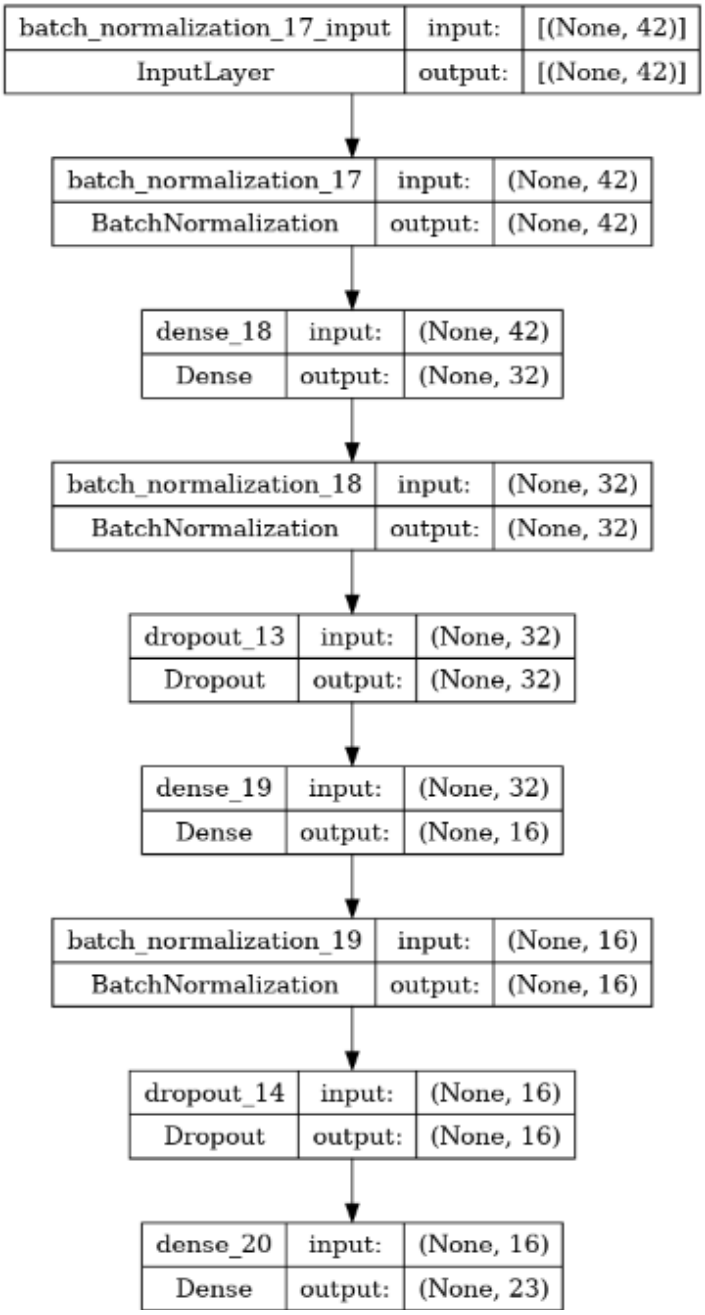

Fig. 4. Baseline classifier DNN

A subtype of GAN networks is a semi-supervised GAN. These are often used when trying to create a generator with little real samples available. In this case the discriminator predicts N+1 classes, with additional label being used for fake data classification. Of particular interest to this research is the efficiency of the categorical discriminator, not the generator model.

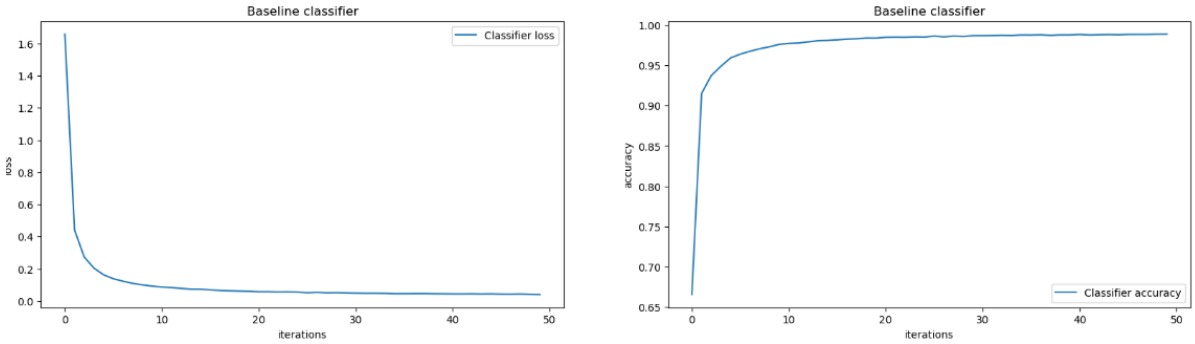

Fig. 5. Baseline classifier metrics

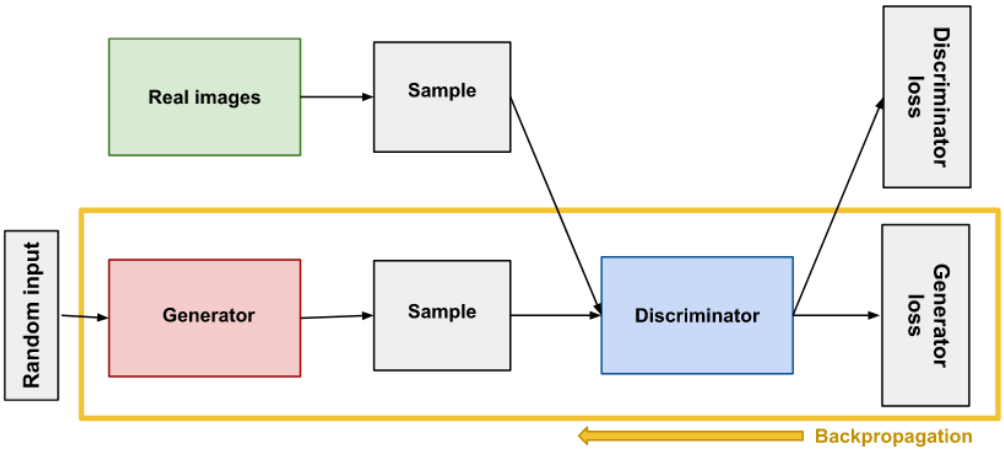

Fig. 6. General GAN architecture

In our implementation, we use two discriminator models, one for real/fake categorisation and another for attack categorisation. Target of the research is the attack categorisation model. The models share weights to ensure correct categorisation for real/fake as well as attack class. We use two dense layers size 256 and “relu” activation, as well as batch normalisation and dropout layers. Output layers are “softmax” for categorical classification model and “sigmoid” for binary classification. Loss functions and metrics are “categorical_crossentropy”, “binary_crossentropy”, “categorical_accuracy”, “binary_accuracy” for categorical discriminator and binary discriminator respectfully (fig. 7) (fig. 8). Since all of our input data is labelled, we only use a small sample of labelled entries, between 100 and 500 samples, as initial

input for categorical classifier model. For generator a model with three dense layers was used with 128, 256 and 512 nodes and “relu” activation. Additionally, batch normalisation and dropout layers were used. Output layer is dense layer with nodes equal to number of features and “tanh” activation (fig. 9). For model training 10 epochs were used. With final training categorical accuracy around 99% and binary accuracy around 78%. Final validation categorical accuracy around 89%. This indicates possible model overfitting (fig. 10-14).

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 42)]              0

 dense_3 (Dense)             (None, 256)               11008

 batch_normalization_3 (Batc  (None, 256)              1024
 hNormalization)

 dropout_2 (Dropout)         (None, 256)               0

 dense_4 (Dense)             (None, 256)               65792

 batch_normalization_4 (Batc  (None, 256)              1024
 hNormalization)

 dropout_3 (Dropout)         (None, 256)               0

 output_cat (Dense)          (None, 23)                5911

=================================================================
Total params: 84,759
Trainable params: 5,911
Non-trainable params: 78,848
```

Fig. 7. Categorical discriminator model

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 42)]              0

 dense_3 (Dense)             (None, 256)               11008

 batch_normalization_3 (Batc  (None, 256)              1024
 hNormalization)

 dropout_2 (Dropout)         (None, 256)               0

 dense_4 (Dense)             (None, 256)               65792

 batch_normalization_4 (Batc  (None, 256)              1024
 hNormalization)

 dropout_3 (Dropout)         (None, 256)               0

 output_bin (Dense)          (None, 1)                 257

=================================================================
Total params: 79,105
Trainable params: 0
Non-trainable params: 79,105
_________________________________________________________________
```

Fig. 8. Binary discriminator model

```
Model: "model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 64)]              0

 dense_5 (Dense)             (None, 128)               8320

 dropout_4 (Dropout)         (None, 128)               0

 batch_normalization_5 (Batc  (None, 128)              512
 hNormalization)

 dense_6 (Dense)             (None, 256)               33024

 dropout_5 (Dropout)         (None, 256)               0

 batch_normalization_6 (Batc  (None, 256)              1024
 hNormalization)

 dense_7 (Dense)             (None, 512)               131584

 dropout_6 (Dropout)         (None, 512)               0

 batch_normalization_7 (Batc  (None, 512)              2048
 hNormalization)

 dense_8 (Dense)             (None, 42)                21546

 model_1 (Functional)        (None, 1)                 79105

=================================================================
Total params: 277,163
Trainable params: 196,266
Non-trainable params: 80,897
_________________________________________________________________
```

Fig. 9. GAN model (generator and discriminator model)

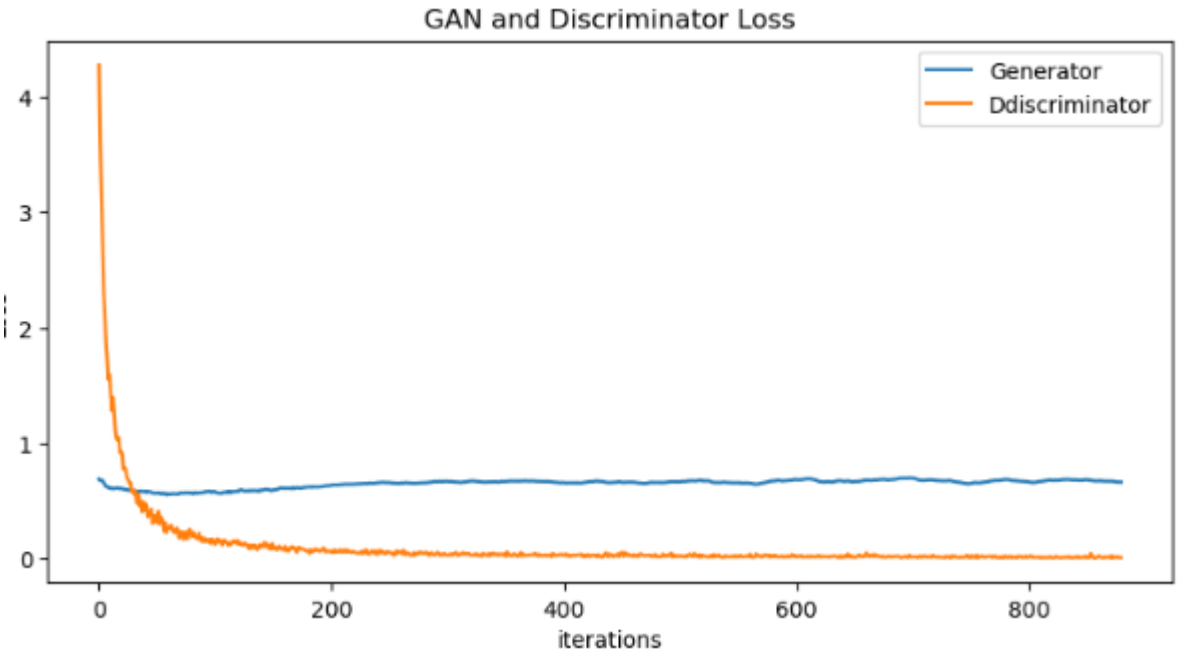

Fig. 10. GAN and discriminator(categorical) losses

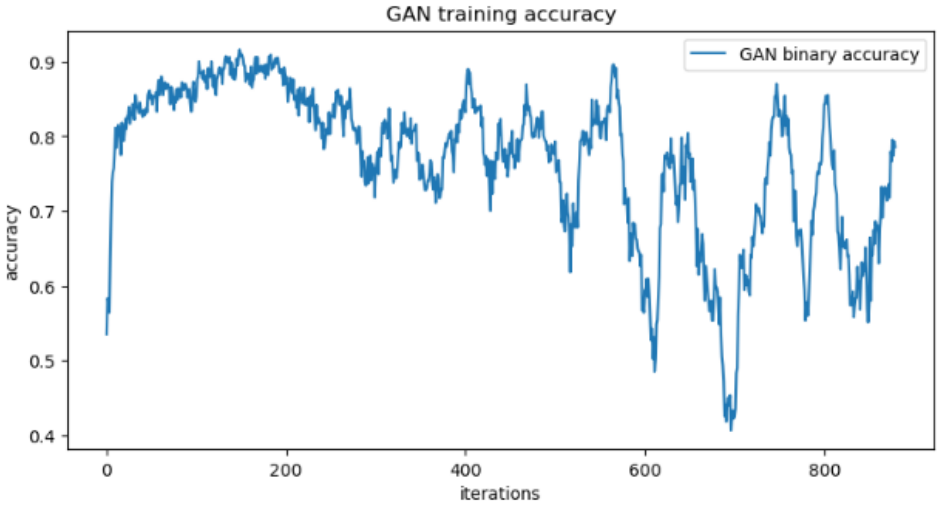

Fig. 11. GAN training accuracy

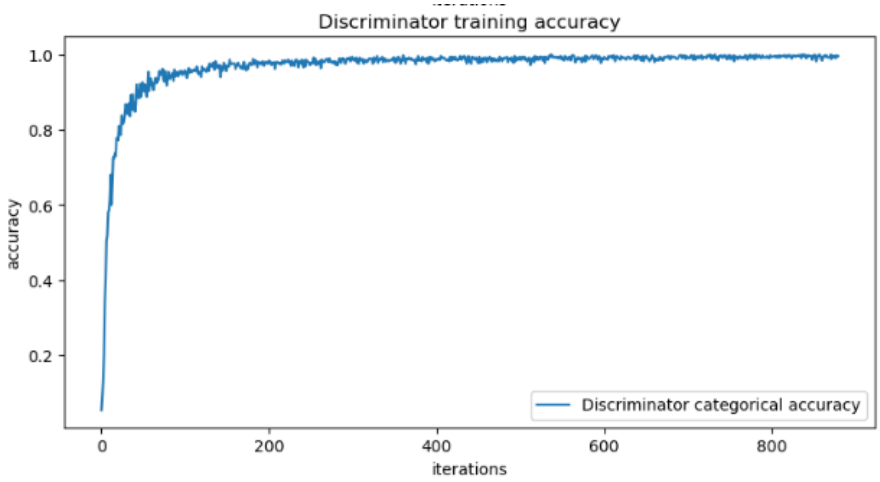

Fig. 12. Discriminator training accuracy

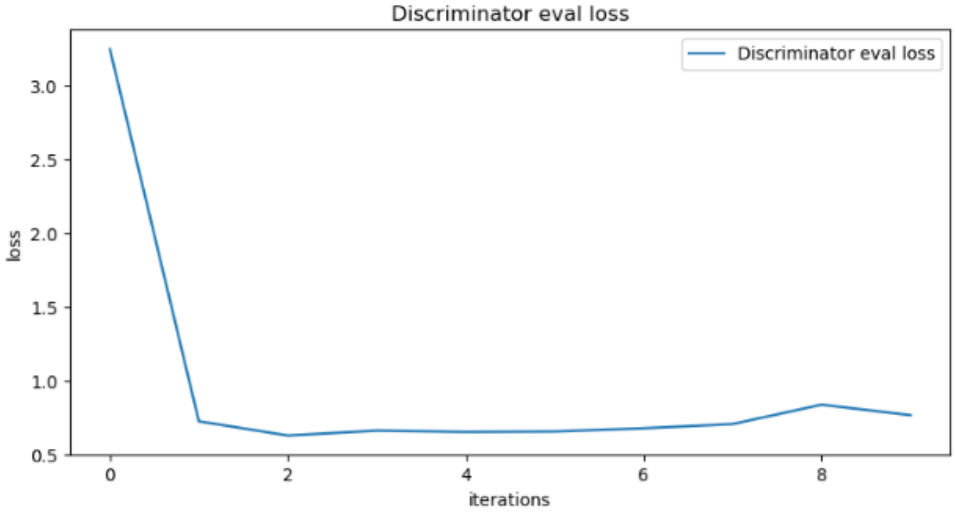

Fig. 13. Discriminator validation data loss (every epoch)

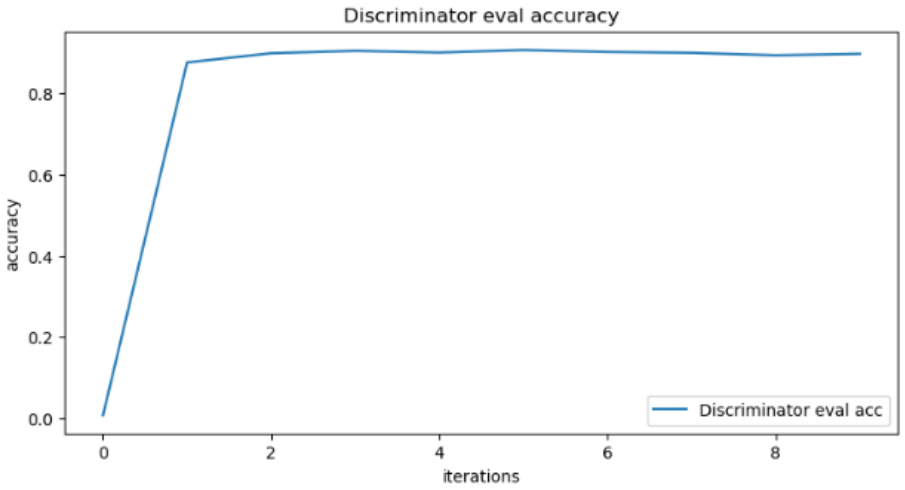

Fig. 14. Discriminator validation data accuracy (every epoch)

Conclusions

This research proposes the use of semi-supervised GAN model to train a classifier network for categorising malicious network traffic with limited number of labelled entries. For comparison we also used a baseline classifier DNN with a full dataset. The baseline classifier managed to achieve a validation accuracy of 99%, whereas SGAN discriminator only achieved 88%. The SGAN discriminator shows signs of overfitting with training accuracy of 99%. While the results are subpar compared to a full dataset classifier, it is worth noting that SGAN model only received a small portion of the dataset labels, between 100 to 500 samples, in different tests, while still achieving a relatively high accuracy score. It should also be pointed out, that GAN networks generally have trouble generating entirely new information, instead it creates slight variations of existing data. As such it may not be able to be used to train a network to predict entirely unknown threats.

Overall, SGAN networks may not be an effective solution to training network attack classifiers, however, additional research may be conducted. In particular, the question of network hyperparameter tuning remains open, as it may allow us to prevent overfitting and improve model accuracy. Additionally, the research was conducted only on a single dataset, it is worth exploring additional datasets to further evaluate the proposed solution.

References

1. Chasaki D., Wu Q., Wolf T. Attacks on Network Infrastructure // 2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN) / HI, USA. Lahaina: IEEE, 2011. C.1-1. URL: https://ieeexplore.ieee.org/document/6005919

2. Anderson R. Security Engineering: A Guide to Building Dependable Distributed Systems. Hoboken: Wiley Data and Cybersecurity, 2020. 1232 c. URL: https://ieeexplore.ieee.org/book/9820859

3. Lan K., Hussain A., Dutta D. The Effect of Malicious Traffic on the Network // Proc. Passive and Active Measurement Wksp. (PAM). 2009. № 10. C.1-1 URL: https://www.researchgate.net/publication/228611227_The_Effect_of_Malicious_Traffic_on_the_Network

4. John W., Olovsson T. Detection of malicious traffic on back-bone links via packet header analysis // Campus-Wide Information Systems. 2008. № 25 (5). C.342-358 URL: https://www.emerald.com/insight/content/doi/10.1108/10650740810921484/full/html

5. Network Traffic Analysis and Intrusion Detection Using Packet Sniffer / M. Qadeer та ін. // 2010 Second International Conference on Communication Software and Networks. Singapore, 2010. C.313-317. URL: https://ieeexplore.ieee.org/document/5437681

6. Wang W., Gombault S., Guyet T. Towards Fast Detecting Intrusions: Using Key Attributes of Network Traffic // 2008 The Third International Conference on Internet Monitoring and Protection. Bucharest, Romania, 2008. C.86-91. URL: https://ieeexplore.ieee.org/document/4561331

7. Network Intrusion Detection System: A Machine Learning Approach' / M. Panda та ін. // Intelligent Decision Technologies (IDT) Journal, IOS Press. 2011. № 4 (5). C.347-356 URL: https://content.iospress.com/articles/intelligent-decision-technologies/idt00117

8. Augustus O. Semi-supervised learning with generative adversarial networks // arXiv 1606.01583. 2016. № 0. C.1-3 URL: https://arxiv.org/abs/1606.01583

9. Pasupa, K., Tungjitnob, S., & Vatathanavaro, S. (2020). Semi-supervised learning with deep convolutional generative adversarial networks for canine red blood cells morphology classification. *Multimedia Tools and Applications*, 1, 1-1. Retrieved from https://doi.org/10.1007/s11042-020-08767-z

AUTHORS

Volokyta Artem – Ph. D., Associate Professor, Department of Computer Engineering NTUU “KPI”.

Scientific interests: real time systems, information security, distributed computations

Dremov Artem – student of the Department of Computer Engineering NTUU “KPI”.

Scientific interests: real time systems, cybersecurity, machine learning.