POSTER: A Re-evaluation of Intrusion Detection Accuracy: an Alternative Evaluation Strategy

Said Al-Riyami
Liverpool University
Liverpool, UK
alriyami@liverpool.ac.uk

Frans Coenen
Liverpool University
Liverpool, UK
coenen@liverpool.ac.uk

Alexei Lisitsa
Liverpool University
Liverpool, UK
lisitsa@liverpool.ac.uk

ABSTRACT

This work tries to evaluate the existing approaches used to benchmark the performance of machine learning models applied to network-based intrusion detection systems (NIDS). First, we demonstrate that we can reach a very high accuracy with most of the traditional machine learning and deep learning models by using the existing performance evaluation strategy. It just requires the right hyper-parameter tuning to outperform the existing reported accuracy results in deep learning models. We further question the value of the existing evaluation methods in which the same datasets are used for training and testing the models. We are proposing the use of an alternative strategy that aims to evaluate the practicality and the performance of the models and datasets as well. In this approach, different datasets with compatible sets of features are used for training and testing. When we evaluate the models that we created with the proposed strategy, we demonstrate that the performance is very bad. Thus, models have no practical usage, and it performs based on a pure randomness. This research is important for security-based machine learning applications to re-think about the datasets and the model's quality.

KEYWORDS

Intrusion detection system, Network Security, Security and Privacy, Domain Adaptation, Machine Learning, Deep Learning

1 INTRODUCTION

In recent years, the number of malware and intrusion attacks has escalated dramatically. Symantec claims that it detected more than 357 millions new variants of malware in 2016 [12]. Kaspersky report that it detected 360,000 new malicious files a day in 2017 [2]. The McAfee Labs Threats Report of 2017 state that it detected 57.6 million new malware or about 157,808 per day [3].

One of the most commonly used intrusion detection methods is *signature-based* detection, which uses knowledge of attacks in the form of signatures, which can be easily checked against monitored traffic. The major issue with this method is the availability of signatures in the context of the constantly growing number of attacks; the set of signatures has to be constantly updated and this requires expert knowledge. This issue can be alleviated in part by the automation of the detection procedures and the use of machine learning methods, in particular, supervised machine learning.

For years, traditional machine learning algorithms have been used for the network intrusion detection system (NIDS). For example, Decision tree learning, Random forest, Logistic regression, Naive Bayes, k-nearest neighbours algorithm (k-NN) and other. But recently, the use of Deep Learning models has become popular in areas of cybersecurity. There are several architectures that can be for deep learning includes Article Neural Network (ANN), Autoencoders (AE), Convolution Neural Networks (CNN) and Recurrent Neural Networks (RNN) with different cells, like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU).

There are several labelled datasets used for the purpose of evaluating network intrusion detection system. For example, NSL-KDD which is a refined version of the popular KDD Cup 1999 datasets. The dataset can be used in term of binary classification (normal, attack), or multi-class classification (5 attack categories, or the name of the exact attack) [13]. Most of the research on NIDS has focused on creating machine learning models that perform better than previously reported in term of accuracy. Other measures are also used as the F1 score. The evaluation strategy used is by train and test in the same dataset. The results based on this method of evaluating are very high. You can get 97% and more accurate detection rate with most of the datasets available to NIDS. We argue that this way of evaluation is far from the main goal of creating an intrusion detection system. The model should build a real understanding of the attack behaviour. But the results we are getting is more towards overfitting on the dataset itself that have been created in a particular computer network setting with a particle methodology to introduce intrusion's ground truth.

Section 2 focus on the current strategy of evaluation. We will demonstrate that we can get very high accuracy in the different datasets by using traditional machine learning algorithms, as well as deep learning models after a good hyperparameters tuning. Our result shows that we can match the reported result or exceed them in some cases. Section 3 we propose training and testing are done in two different datasets. By doing so, we show that the models perform poorly. This result is very important to re-think the models and datasets available in this area.

2 GETTING ACCURACY HIGH

Most of the time when we use traditional machine learning algorithms for NIDS, we are getting a higher result comparing with deep learning algorithms. In this section, we want to demonstrate that we can use most of the deep learning models and try to reach to the same performance of classical machine learning or even more. The aim of this part is to show that getting a high-performance

CCS '18, October 15–19, 2018, Toronto, ON, Canada

ACM ISBN 978-1-4503-5693-0/18/10.
https://doi.org/10.1145/3243734.3278490

Table 1: Evaluation with the current strategy - F1 Score

Algorithm	Kyoto+	NSL-KDD	gureKDD	NSL-KDD (multi)
DT	97.64%	99.46%	99.92%	99.36%
RF	98.19%	99.56%	99.94%	99.30%
logit	93.57%	93.69%	97.34%	95.27%
k-NN	97.16%	99.15%	99.92%	98.07%
ANN	98.13%	99.24%	99.46%	98.25%
LSTM	97.91%	99.21%	99.42%	98.45%
GRU	98.03%	99.17%	99.31%	98.40%

model is not hard based on the strategy method of evaluating the models.

2.1 Methodology

We will use three different datasets: Kyoto2006+, NSL-KDD, and gureKDD. First, we will check the accuracy that has been reported in those datasets. Then, we will run different traditional machine learning algorithms and report the result. Finally, we will try different deep learning algorithms. Even though we getting low accuracy at the beginning, comparing with traditional machine learning, but we will show with the right hyperparameters tunning we can match or bypass the result of the traditional machine learning.

The training and testing are done within the same dataset. The split is around 80% for training and 20% for testing. Then, we will calculate the performance only based on the result of the testing part. The performance can be measured by the ratio of the total number of correct predictions by the total number of predictions. But to avoid any Accuracy Paradox that might be there because of the distribution imbalance of the data, we use the F1 score calculated using Equation 1. The F1 will balance between Precision and Recall.

The models that we used are: Decision Tree (DT), Random forest (RF), Logistic Regression (logit), K-NN, Artificial Neural Network (ANN), and Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) and with Gated Recurrent Unit (GRU).

$$F1 = 2\frac{Precision \times Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN} \quad (1)$$

2.2 Kyoto+2006 Dataset Experiments

Kyoto 2006+ is a dataset that has been generated for the purpose of NIDS [10][11]. They used honeypot traffic as a source of ground truth about the malicious activities. The dataset contains 22 features. There are 3 class of labels for this dataset: normal, known attack, unknown attack. We will convert the problem to be binary where 0 means normal traffic and 1 means malicious traffic. For comparison reason, we will follow the same split and preprocessing of this dataset to the one reported in [1]. The best accuracy results in that paper are ≈84.15%. And after we calculated their F1 score we got ≈87.56%. Another reported work with the same datasets can be found here [9]. They used a stack of Artificial Neural Network (ANN) layers. Their best result shows when they used 1000 hidden neuron with F1 Score of ≈97.50%.

After we experiments with different models, the result we obtained is shown in Table 1. We reach to an F1 score of ≈98.13% by using normal Artificial Neural Network (ANN).

2.3 KDD Variation Datasets Experiments

KDD cup 1999 dataset is one of the most known datasets for NIDS. Different variations created based on this dataset because of reported problem [5][4]. We will use NSL-KDD and gureKDD datasets. NSL-KDD is a refined version of KDD [13]. The gureKDD dataset is generated from a scratch with the same purpose of KDD [6].

Many studies were done by using NSL-KDD dataset. For example, this paper [7] reported an accuracy of 99% for both binary and multiclass classification by using SVM. This paper [8] reached 99.6% with K-NN. This paper [14] reported an accuracy of 97.09% by using a recurrent neural network (RNN).

After we experiment with different models, the result we obtained is shown in Table 1. For binary classification of both NSL-KDD and gureKDD, most the models give ≈99% of the F1 score. We also show that this result can be obtained for the multiclass problem. To demonstrate that, we used 40 classes of different attacks including normal traffic label. The result is shown in table 1. You can notice that the result are around ≈98%-99%.

As you can notice from Table 1, our result matches the current reported results or, sometimes, exceeded them by using the same evaluation strategy.

3 A DIFFERENT EVALUATION STRATEGY

As we can see from the from previous, we can reach high accuracy results. Many models created reach to 99% accuracy. Based on the current way to assess the NIDS model, does it mean that we can stop here and used the learned model in NIDS? Does the model we created represent attack understanding learning? Or is it just an over-fitting based on the dataset which represents a certain type of computer network with a particular setting? In order to address these questions, we propose another evaluation strategy. We propose to use two different datasets that have the same domain but a different distribution of traffic. Both of them should share a same purpose and features, but generated from two different computer networks. In this way, we will make sure that the model not learning just the distribution of traffic in the network, but should generalized the understating of the thing we want to find. In our problem, we want to learn the abstract behaviour of the attack that allows us to find the attack in a different network or find a new zero-day attack with the same understanding.

3.1 Methodology

We will use NSL-KDD and gureKDD. Both datasets represent different computer networks, which will give us different distribution in the traffic and the same set of features. Also, both datasets are available for public download. The training will be done in one dataset and the testing in the other, and vice versa. We take one dataset as a whole in the training and another as a whole in the testing. We will follow the same process as before where will use traditional machine learning and different deep learning models. The problem will be binary, but the idea can be applied to multiclass problems as well. The importance of F1 score will show in the result. So, consider only F1 score as a performance indicator.

Table 2: New Evaluation Strategy - NSL-KDD(training), gureKDD(testing)

Algorithm	Accuracy	F1 Score
Random Forest	97.65%	36.08%
ANN	80.85%	9.59%
LSTM	87.21%	17.5%

Table 3: New Evaluation Strategy - gureKDD(training), NSL-KDD(testing)

Algorithm	Accuracy	F1 Score
Random Forest	52.71%	3.62%
ANN	52.16%	5.19%
LSTM	52.78%	7.40%

3.2 Results And Discussion

During the training of the model, all of them reach the accuracy of ≈99%. But we will only report the accuracy during the testing phase. Table 2 shows the result when NSL-KDD used for training and gureKDD used for testing. Table 3 shows the result when we use gureKDD for training and NSL-KDD for testing.

As you can notice from the result obtained, all models give a very low F1 score. From the current way to evaluate the performance, the model is perfect. But when we test it with a different network, the results are just random guesses. It means that the model didn't carry out the real learning of intrusions. This way to measure the performance of NIDS models will not only benefit the way we create our model but also about the datasets that we have and the way we created them. From this angle, the datasets should have those qualities. It should represent attack/malicious understanding learning and can adapt to the change in the network traffic. The general way to use the method is to train in certain dataset and test in another dataset that has been generated from different network traffic. But we can also go for more specific scenarios to evaluate based on this method as follows:

- When we have two datasets that have the same type of attacks. We want to evaluate if we can find all the attacks or not. This similar to the problem that we demonstrate before.
- When we have a categories of attack shared and we want to evaluate the goodness of finding each category of attack.
- Evaluate the quality of the datasets. A high-quality dataset should facilitate the good learnability of intrusion that can be transferred to different datasets.
- Evaluating the finding of zero-day attack. Can the model created in one dataset detect unseen attack from another dataset?
- Multiple dataset learning: we can also make the model train from multiple datasets and test in one or more datasets.

We should avoid train and test our model from traffic or information on the same computer network. Even if they have different names. For example, we cannot train in NSL-KDD and test in KDD99. Because they are basically the same dataset. If your result is below 50% of the F1 score, this means your model can not adapt to the new dataset. If above 50%, it means that your model starts to learn something about the intrusion.

4 CONCLUSIONS

For a long time, we treated the problem of network intrusion detection system (NIDS) without the concern with the practicality of the result in the real world. The researches are done in such a way we train and test our model in the same dataset and just calculate the accuracy. But, we argue that this way of thinking is not practical. So, we spent the first section where we use different datasets with different models. We showed that it's not difficult to reach to very high accuracy. It's just required some hyperparameters tunning, which lead up to 99% accuracy. But we argue that those results do not represent the actual world. So, we spent out the second part to use another evaluation strategy to measure the performance of the model. We set up the experiments where the training is done in one dataset and tested in another dataset and vice versa. And the result shows that all models perform bad, with less than 40% F1 score and most of them less than 10%. This paper is a call to rethink the current way to measure the quality of the models and datasets we have in NIDS.

REFERENCES

[1] Abien Fred M Agarap. 2018. A Neural Network Architecture Combining Gated Recurrent Unit (GRU) and Support Vector Machine (SVM) for Intrusion Detection in Network Traffic Data. In *Proceedings of the 2018 10th International Conference on Machine Learning and Computing*. ACM, 26–30.

[2] Kaspersky Lab. 2017. Number of the Year: 360,000 Malicious Files Detected Daily in 2017. (2017).

[3] McAfee Lab. 2017. Threat Report. (2017).

[4] Matthew V Mahoney and Philip K Chan. 2003. An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection. In *International Workshop on Recent Advances in Intrusion Detection*. Springer, 220–237.

[5] John McHugh. 2000. Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Transactions on Information and System Security (TISSEC)* 3, 4 (2000), 262–294.

[6] Iñigo Perona, Ibai Gurrutxaga, Olatz Arbelaitz, José I Martín, Javier Muguerza, and Jesús Ma Pérez. 2008. Service-independent payload analysis to improve intrusion detection in network traffic. In *Proceedings of the 7th Australasian Data Mining Conference-Volume 87*. Australian Computer Society, Inc., 171–178.

[7] Muhammad Shakil Pervez and Dewan Md Farid. 2014. Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs. In *Software, Knowledge, Information Management and Applications (SKIMA), 2014 8th International Conference on*. IEEE, 1–6.

[8] B Basaveswara Rao and K Swathi. 2017. Fast kNN Classifiers for Network Intrusion Detection System. *Indian Journal of Science and Technology* 10, 14 (2017).

[9] Raman Singh, Harish Kumar, and RK Singla. 2015. An intrusion detection system using network traffic profiling and online sequential extreme learning machine. *Expert Systems with Applications* 42, 22 (2015), 8609–8624.

[10] Jungsuk Song, Hiroki Takakura, and Yasuo Okabe. 2006. Description of kyoto university benchmark data. *Available at link: http://www. takakura. com/Kyoto_data/BenchmarkData-Description-v5. pdf [Accessed on 15 March 2016]* (2006).

[11] Jungsuk Song, Hiroki Takakura, Yasuo Okabe, Masashi Eto, Daisuke Inoue, and Koji Nakao. 2011. Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation. In *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*. ACM, 29–36.

[12] Symantec. 2017. Internet Security Threat Report. (2017).

[13] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. 2009. A detailed analysis of the KDD CUP 99 data set. In *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*. IEEE, 1–6.

[14] Chuanlong Yin, Yuefei Zhu, Jinlong Fei, and Xinzheng He. 2017. A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access* 5 (2017), 21954–21961.