https://doi.org/10.7840/kics.2021.46.12.1

불균형 CIC-IDS-2018 데이터 세트에 대한 CatBoost 기반 네트워크 침입 탐지

알리카노브 주마벡◆, 양 승 삼*, 노 영 태°

CatBoost-Based Network Intrusion Detection on Imbalanced CIC-IDS-2018 Dataset

Alikhanov Jumabek◆, SeungSam Yang*, YoungTae Noh°

요 약

증가하는 인터넷 트래픽의 양과 속도는 악의적인 공격자에게 전례 없는 공격 기회를Tempo 제공한다. 예를 들어 악의적인 침입 네트워크 트래픽을 감지하기 위한 네트워크 침입 감지 시스템(NIDS)에 과부하가 발생할 수 있다. 현재 NIDS 솔루션의 대부분은 flow 정보(예: 패킷 수, 평균 도착 간 시간)가 포함된 flow 기록을 활용하며, 이러한 flow 정보를 기반으로 트리 기반 ML 모델을 활용한다. 그러나 최근 CatBoost와 같은 Gradient Boosting Machine 방법은 Kaggle 대회와 같은 표 형식 데이터 셋에서 기존의 트리 기반 솔루션보다 우수한 성능을 보여주었다. 이 논문에서 우리는 네트워크 침입 탐지 작업을 위한 CatBoost의 적용 가능성을 탐구한다. 또한 데이터 불균형을 해결하여 얻은 성능 향상을 시연한다. 또한 다양한 유형의 최근 실제 사이버 공격이 포함된 최신 CIC-IDS-2018 데이터 세트를 활용한다. 우리의 실험에 따르면 단순한 오버샘플링 기술로 데이터 불균형을 해결하면 CatBoost의 경우 88.84%에서 92.41%로 정확도가 크게 향상되었으며, 의사 결정 트리(88.34%) 및 랜덤 포레스트(89.88%)의 정확도를 능가했다.

Key Words : network intrusion detection, gradient boosting machine, machine learning, data imbalance

ABSTRACT

Increasing volume and speed of internet traffic fosters unprecedented opportunity for malicious attackers. This in turn creates challenges for network intrusion detection systems (NIDSs) whose job is to detect intrusive (i.e., malicious) network traffic. Majority of current solutions exploit flow records which contain information regarding the flow (e.g., number of packets, avg. inter-arrival time). Hence, most of the NIDS solutions exploit tree-based ML models such as Decision Tree and Random Forest due to the tabular form of a flow record. However, recently Gradient Boosting Machine methods such as CatBoost has shown their superior performance over traditional tree-based solutions on tabular datasets such as in Kaggle competitions. In this work we explore the applicability of CatBoost for network intrusion detection task. Further, we

※ **This research was supported in part by ICT Innovative Company Technology Development Support Program through the Institute for Information & Communication Technology Planning & Evaluation (IITP) funded by Ministry of Science and ICT (2020-0-01965), and by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2020R1A4A1018774)**
◆ **First Author : Inha University Computer Science Engineering, jumabek4044@gmail.com, 학생회원**
° **Corresponding Author : Inha University Computer Science Engineering, ytnoh@inha.ac.kr, 종신회원**
* **Inha University Computer Science Engineering, seungsam2357@gmail.com,**
논문번호 : 202109-239-B-RU, Received December 22, 2014; Revised December 22, 2014; Accepted December 22, 2014.

demonstrate the performance gain achieved by addressing data imbalance. Our experimental comparisons show that addressing data imbalance with simple over-sampling technique can provide significant performance boost - from 88.84% to 92.41% accuracy improvement in the case of CatBoost. Results also suggest CatBoost classifier (92.41%) outperforms Decision Tree and Random Forest (88.34% and 89.88%) in term of balanced accuracy on CIC-IDS-2018 dataset.

Ⅰ. Introduction

Continuous increase in volume and speed of the network traffic is creating new opportunity for malicious parties. This creates new challenges for Network Intrusion Detection Systems (NIDSs) whose task is to discover and prevent malicious activities. Increased network protocol diversity such as different TCP variants[3] further adds to the challenge. Large body of works have been proposed to mitigate the problem by improving the robustness and efficiency of NIDS[1,2]. Majority of recently proposed NIDSs rely on Machine learning (ML) or Deep Learning (DL) based methods[4-11]. Meanwhile, gradient boosting machines (GBMs) have shown their empirical success on tabular datasets in data science competitions such as Kaggle[13-15] and other domains such as ad click predictions[21]. However, to the best of our knowledge there is no study on NIDSs which investigated the performance of GBMs. This inspired us to explore and compare the performance of candidate GBM – CatBoost[13] classifier against other traditional tree based methods such as Decision Tree and Random Forest in imbalanced network intrusion detection task. Another problem we investigated was the data imbalance which naturally occurs in NIDS datasets where samples for malicious traffic are rare compared to normal traffic. However, this problem is ignored in most of the studies. Hence, we also demonstrate how a simple random-over-sampling based data balancing approach can improve the performance of ML-based models.

Our main contributions can be listed in four parts as follows:

1. CatBoost-based NIDS model is proposed. Experiments show its superiority to other tree based methods in terms of balanced accuracy metric in multi-class classification setting.

2. We demonstrated the performance gain achieved after addressing data imbalance.

3. Performance evaluation of classifiers on recent dataset that captured realistic cyber-attack scenarios is provided.

4. Our jupyter notebook code for experiments will be open sourced after the acceptance of this manuscript. We hope our code can facilitate further research on gradient boosting based methods for network intrusion detection task and provide practical guide on tackling data imbalance.

Ⅱ. Literature Study

Depending on the granularity of the inspected data, NIDSs can be categorized into packet based and flow based methods. Deep Packet Inspection (DPI) is computationally expensive as it examines every network packet closely in detail. Furthermore, its effectiveness is limited when the content of the packet is encrypted which is becoming a de facto in modern day network traffic.

Flow-based NIDS is favoured for its effectiveness and being applicable even when the traffic is encrypted. Our focus in this study is a flow-based NIDSs. Flow-level NIDS approaches themselves could be further divided into two main groups: Anomaly Detection (AD) based and Misuse Detection (MD) based[1,17,18]. AD approaches capture activities that deviate from normal profile. A main advantage of AD-based approaches is the ability to detect unseen attacks. However, they often suffer from high false alarm rate due to previously unseen yet legitimate traffics. In contrast, MD-based approaches aimed at identifying previously known attacks based on their signature and patterns. They are effective for detecting the known types of

attacks but fail to recognize previously unseen attacks.

Mirksky et al.[17] developed Kitsune - AD based NIDS that is comprised of ensemble of autoencoders. Kitsune can efficiently detect attacks in an online manner while having a comparable performance to offline trained anomaly detectors. Bovenzi et al.[18] proposed a lightweight anomaly detection by designing a novel multi-modal deep autoencoder. Their proposed autoencoder treats each categoricalfeature as a separate modality and achieves good performance while having a lightweight design for IoT scenarios.

This study focused on MD-based NIDS that operates on flow information (records). In the following we highlight some of the recent works on ML-based NIDSs that use misuse detection approach. Shone et al.[9] adopted a non-symmetric autoencoder to learn features in an unsupervised manner from unlabeled data. Once autoencoder based feature extractor is learned, features are extracted and RF classifier is trained. Similarly, Javaid et al.[8] proposed to learn feature representation using sparse autoencoder on unlabeled data. Then, softmax classifier is trained on labeled data where features are extracted from pre-trained autoencoder. Yin et al.[19] proposed a Recurrent Neural Networks (RNNs) by considering both binary and multi-class classification settings. Authors also investigated the impact of the number neurons in the hidden layers of RNN and learning rate on models performance. Their results outperformed other ML-based classifiers such as J48, ANN, RF and SVM on NSL-KDD[22] benchmark.

A major limitation of previous studies is the usage of outdated network traces (i.e. dataset) such as KDD or NSL-KDD[22]. Another knowledge gap is the lack of research work that studied the application of boosting algorithms such as CatBoost[13] on network intrusion detection task. Our work departs from previous studies in two main ways. First, we propose state-of-the-art boosting algorithm CatBoost-based NIDS model. Second, we validate the performance of our proposed model on recent CIC-IDS-2018 dataset where we compare our model with well known machine learning and deep learning based models.

Unless otherwise mentioned, throughout the paper we refer to flow-based NIDS that is misuse detection based approach.

Ⅲ. Catboost-Based Nids

In this section we elaborate on deployment scenario of proposed CatBoost-based NIDS. A schematic view of the components involved in the NIDS is depicted in Fig. 1. We composed the system by following NetFlow – a flow export technology design where flow monitoring involves three stages: 1) flow metering and exporting, 2) collection, and 3) analysis. The flow metering and exporting process is performed on a metering device (e.g., switch). Flow information (e.g., number of packets, average inter-arrival time) is accumulated in the flow cache and exported to the collector. Collector stores the received flow records for further usage. Additionally, it sends flow records to the monitoring applications such as NIDS for real-time analysis. It is noteworthy to mention that benchmark network intrusion datasets do not consider the presence of sampling. Hence, flow records are extracted from full traffic and not sampled set of packets.

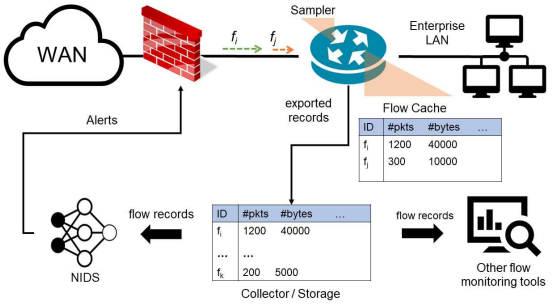

Fig. 1. Schematic view of flow monitoring architecture which is located between WAN and enterprise LAN. While packets are passing through, flow information is accumulated in the switch from sampled set of packets. Once flow finishes or timer expires, accumulated record of the flow is exported to Collector. Collector can store the records for future and/or pass it to real-time monitoring systems, in our case CatBoost-based NIDS. NIDS analyzes the flow record and informs firewall or network administrator if a record for that particular flow is found malicious.

Ⅳ. Experimental Setup

4.1 Data Preparation

To conduct our benchmarks we used CIC-IDS-2018 dataset[20] - that is relatively recent, large scale and captured realistic cyber attack scenarios during data collection. Dataset provides both full PCAP traces and preprocessed CSV files which contain flow records extracted by CICFlowMeter. In this work, we exploit only the CSV files that is located in "Processed Traffic Data for ML Algorithms" folder of CIC-IDS-2018 dataset. Specifically, we obtained labeled flow records from AWS cloud with the following command: ''aws s3 sync -no-sign-request -region ap-northeast-2 's3://cse-cic-ids2018/Processed Traffic Data for ML Algorithms' <dest-dir>''. Table 1 shows the number of flow records present in each of ten CSV files.

We found and eliminated 59 rows that contain meaningless string names (e.g., 'Label' for Label columns) that are likely to be erroneous. More specifically, Friday-16-02, Wednesday-28-02 and Thursday-01-03 had 1, 33 and 25 noisy records/rows records in corresponding order. Each flow in the dataset contains 78 flow properties measured by CICFLowMeter[2].

Flow identifying columns such as Flow-ID, Src-IP, Src-Port, Dst-IP are also excluded from analysis as well. In order to be able to process big datawith 64G RAM memory, we downsized the datasize 10x by excluding excluding the portion of the normal traffic. Class distribution of the downsized dataset is shown in Fig. 1. Note, here class refers to the type of malicious attack that is being carried for the given flow. Benign label represents the flow of normal traffic. More details can be found in our code that we are planning to release after the acceptance.

Table 1. Number of Flow records provided in each file of CIC-IDS-2018 dataset.

No.	Filename	#records
1	Wednesday-14-02-2018_TrafficForML_CICFlowMeter.csv	1,048,576
2	Thursday-15-02-2018_TrafficForML_CICFlowMeter.csv	1,048,576
3	Friday-16-02-2018_TrafficForML_CICFlowMeter.csv	1,048,576
4	Tuesday-20-02-2018_TrafficForML_CICFlowMeter.csv	7,948,749
5	Wednesday-21-02-2018_TrafficForML_CICFlowMeter.csv	1,048,576
6	Thursday-22-02-2018_TrafficForML_CICFlowMeter.csv	1,048,576
7	Friday-23-02-2018_TrafficForML_CICFlowMeter.csv	1,048,576
8	Wednesday-28-02-2018_TrafficForML_CICFlowMeter.csv	613,105
9	Thursday-01-03-2018_TrafficForML_CICFlowMeter.csv	331,126
10	Friday-02-03-2018_TrafficForML_CICFlowMeter.csv	1,048,576

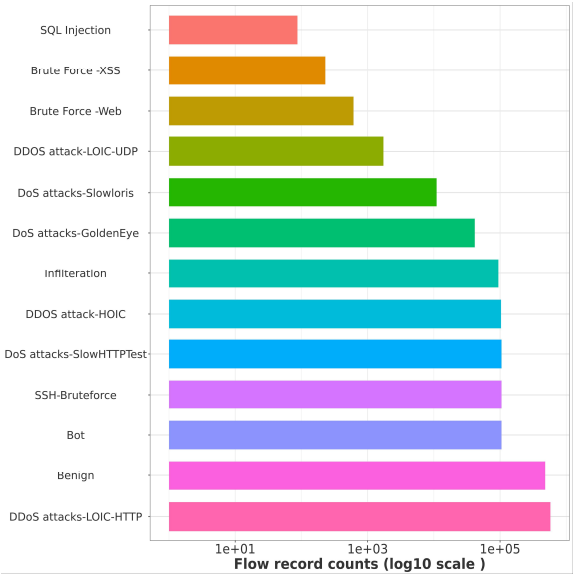

Fig. 2. Distribution of flow records in CIC-IDS-2018 dataset. y-axis corresponds to number of records that belong to a given category of attacks or benign category in the case of normal traffic.

4.2 Cross Validation

CIC-IDS-2018 dataset has 10^7 flow records, performing K-fold cross validation would be too costly. For efficiency purposes we resort to three-way holdout cross validation where data is split into train/test set in 80:20 ratio. To ensure the presence of rare classes in test set, we use StratifiedShuffleSplit interface from Scikit-Learn[12]. In case of CatBoost, we furter split train data into what we call dev and val sets where model is trained on dev set and periodically validated on val

set to prevent overfitting. Note, test set is never used during model training or validation.

4.3 Categorical Variables

Dataset contains two categorical variables Dst-Port and Protocol, we encode them using Scikit-Learn's OrdinalEncoder. Note, in the case of CatBoost, we do not use any encoding as CatBoost naturally supports categorical variables.

4.4 Data Balancing

There are various techniques that address the data imbalance. One way to handle data imbalance is to oversample minority classes until they have same number of samples with the majority class. Among them random under- and over- sampling are the most commonly used. We chose random over-sampling based data balancing instead of under-sampling based balancing in order to preserve as much as intrusive (i.e., malicious flow records) as possible.

It is noteworthy to mention that data balancing is only applied on train set. In the case of CatBoost, train set is further divided into dev:val set on 80:20 ratio. Hence, for CatBoost we consider additional setting where dev and val set are balanced separately to ensure oversampled (i.e., duplicate) data do not end up in both dev and val set after splitting.

Although, more sophisticated over-sampling based techniques such as SMOTE (Synthetic Minority Over-sampling Technique) are available, for the purpose of clarity we only used over- sampling based data balancing.

4.5 ML Model Parameters and Experimental Environment

For Decision Tree and Random Forest classifiers we used the default hyper parameters provided in scikit-learn[12] version 0.24.1. For CatBoost we used the default parameters of catboost[13] library where 1000 iterations are used to build the model. To speed up the training phase, our CatBoost model is trained on GPU.

All the experiments were conducted on the same machine with 64G RAM and Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz. For CatBoost training a Titan V GPU of 11G memory is used. Since our comparisons do not include training time of classifiers, training classifiers in different environment in terms of training time is acceptable.

Table 2. Parameters used for each classifier for building NIDS models.

No.	Decision Tree	Random Forest	CatBoost
Node split strategy	Best split is chosen	1,048,576	NA
Tree depth	unconstrained	unconstrained	NA
Examined #features at each split	√(n_features)	√(n_features)	NA
number of estimators	NA	100	1000
optimization function/criterion	Gini impurity	Gini impurity	Cross Entropy
early stopping	NA	NA	20 iterations
Random Seed	42	42	42

Ⅴ. Experimental Results

In this section we compare our proposed CatBoost based network intrusion detection model to other tree based methods such as RandomForest and Decision Tree. To prevent biased evaluation, we use balanced accuracy instead of (plain) accuracy as an evaluation metric. This is because (plain) accuracy metric is biased in favor of majority classes/categories.

When working with imbalanced datasets, addressing the data imbalance on train set should be performed. Otherwise, it can cause learning issues for the model and result in biased evaluation in favor of frequent categories. To demonstrate the effect of addressing data imbalance, in Table 2 we

Table 3. Balanced accuracy of tree based classifiers on CIC-IDS-2018 dataset (in %).

Classifier	Imbalanced	Balanced
Decision Tree	87.36	88.34
Random Forest	88.13	89.88
CatBoost	88.84	91.95
CatBoost (dev/val separately balanced)	NA	92.41

report the results for both balanced and imbalanced settings.

First insight derived from the experiments is the significant effect of addressing data imbalance, especially in the case of CatBoost (88.84% to 91.95%). We can observe the remarkable performance gains for all three classifiers. This observation is inline with previous study on imbalanced data[16] where addressing data imbalance improved the performance. Second insight is the superior performance of CatBoost classifier in both imbalanced and balanced settings. It is noteworthy to mention that our dataset imbalance severity was reduced when we elimintated 90% of the dataset by removing a portion of benign cases during pre-processing stage. Thus, when the imbalance severity is high we expect the effect of addressing data imbalance to be high as well.

Ⅵ. Conclusion

In this paperwe showed the applicability of CatBoost classifier for the network intrusion detection task. We compared our results to other tree based classifiers such as Decision Tree and Random Forest. We also demonstrated the performance gains achieved by addressing data imbalance. Results show that addressing data imbalance has major performance benefits while CatBoost classifier showed promising results over other tree based methods on both balanced and imbalanced settings. Similarly to previous work, this study relied on the preprocessed flow records provided by the CIC-IDS-2018 dataset authors. However, in such setting same flow could have multiple sub-records and those sub-records could appear in both train and test set after data split. Such a scenario creates evaluation bias. Therefore, in future flow-level evaluation should be designed.

References

[1] M. H. Umer, M. Sher, and Y. Bi, "Flow-based intrusion detection: Techniques and challenges," *Comput. & Secur.*, vol. 70, pp. 238-254, 2017.

[2] A. H. Lashkari, Github, *CICFlowMeter*, [Online]. Accessed: Sep. 2021. Available: https://github.com/ahlashkari/CICFlowMeter

[3] M. Salman, T. J. Chaudhery, and Y. Noh, "Study on performance of AQM schemes over TCP variants in different network environments," *IET Commun.*, 2020.

[4] V. K. Github, "*Network intrusion detection on UNSW-NB15*," Accessed: Sep. 2021. [Online]. Available: https://github.com/vinayakumar/Network-IntrusionDetection/blob/master/UNSW-NB15/CNN/multiclass/cnn2.py.

[5] R. Vinayakumar, et al., "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525-41550, 2019.

[6] J. Kevric, J. Samed, and S. Abdulhamit, "An effective combining classifier approach using tree algorithms for network intrusion detection," *Neural Comput. Appl.*, vol. 28, no. 1, pp. 1051-1058, 2017.

[7] N. Sultana, et al., "Survey on SDN based network intrusion detection system using machine learning approaches," *Peer-to-Peer Netw. and Appl.*, vol. 12, no. 2, pp. 493-501, 2019.

[8] A. Javaid, et al., "A deep learning approach for network intrusion detection system," *Eai Endorsed Trans. Secur. Safety*, vol. 3, no. 9, Art. no. e2, 2016.

[9] N. Shone, et al., "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41-50, 2018.

[10] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," *ICISSp*, vol. 1, pp. 108-116, 2018.

[11] M. A. Ferrag, et al., "Deep learning techniques for cyber security intrusion detection: A detailed analysis," *6th Int. Symp. ICS & SCADA Cyber Secur Res.*, pp. 126-136, 2019.

[12] F. Pedregosa, et al., "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825-2830, 2011.

[13] A. V. Dorogush, V. Ershov, and A. Gulin, "CatBoost: Gradient boosting with categorical features support," arXiv preprint arXiv: 1810.11363, 2018.

[14] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, pp. 1189-1232, 2001.

[15] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery and Data Mining, in KDD*, pp. 785-794, 2016.

[16] A. Meliboev, J. Alikhanov, and W. Kim, "1D CNN based network intrusion detection with normalization on imbalanced data," *ICAIIC*, pp. 218-224, 2020.

[17] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," arXiv preprint arXiv:1802.09089, 2018.

[18] G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico and A. Pescape, "A hierarchical hybrid intrusion detection approach in IoT scenarios," in *GLOBECOM*, Taipei, Taiwan, Dec. 2020.

[19] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," in *IEEE Access*, vol. 5, pp. 21954-21961, 2017.

[20] A. H. Lashkari, "*CSE-CIC-IDS 2018 on AWS*," distributed by www.unb.ca, https:// www.unb.ca/cic/datasets/ids-2018.html, 2019.

[21] Y.-J. Han and I.-W. Joe, "Prediction of ad clicks using early stop based on XGBoost," *J. KICS*, vol. 46, no. 6, pp. 993-1000, 2021.

[22] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," *IEEE Symp. Comput. Intell. Secur. Defense Appl.*, pp. 1-6, 2009.

알르카노브 주마벡 (Alikhanov Jumabek)

2014년 8월 : 타슈켄트 정보 기술 대학 기계 공학과 졸업
2017년 2월 : 인하대학교 컴퓨터 석사 졸업
2019년 9월~현재 : 인하대학교 컴퓨터공학과 박사과정
<관심분야> 컴퓨터 시각 인식, 네트워크 보안, 모바일 데이터 과학, 머신러닝 및 딥러닝
[ORCID:0000-0003-3103-6033]

양 승 삼 (SeungSam Yang)

2019년 2월 : 인하대학교 수학과 학사, 컴퓨터 공학과 복수전공
2021년 8월 : 인하대학교 전기컴퓨터공학과 석사
<관심분야> 네트워크 시스템 및 보안, 암호이론
[ORCID:0000-0002-0938-1570]

노 영 태 (YoungTae Noh)

2004년 8월 : 조선대학교 전계산학과 학사
2007년 2월 : 광주과학기술원정보기술공학부 석사
2007년 9월 : 광주과학기술정보기술원 정보통신공학과 연구원
2012년 6월 : University of California 컴퓨터 과학과박사
2012년 7월~2014년 11월 : Cisco Systems Software Engineer
2015년 2월~2015년 8월 : Purdue University Post-doc
2015년 9월~현재 : 인하대학교 컴퓨터 공학과 교수
<관심분야> 데이터 센터 네트워킹, 무선 네트워킹, HCI(Human-computer interaction), 머신러닝
[ORCID:0000-0002-9173-1575]