GPU-based Classification for Wireless Intrusion Detection

Alina Lazar
alazar@ysu.edu
Youngstown State University
Youngstown, OH

Alex Sim
Kesheng Wu
asim@lbl.gov,kwu@lbl.gov
Lawrence Berkeley
National Laboratory
Berkeley, CA

ABSTRACT

Automated network intrusion detection systems (NIDS) continuously monitor the network traffic to detect attacks or/and anomalies. These systems need to be able to detect attacks and alert network engineers in real-time. Therefore, modern NIDS are built using complex machine learning algorithms that require large training datasets and are time-consuming to train. The proposed work shows that machine learning algorithms from the RAPIDS cuML library on Graphics Processing Units (GPUs) can speed-up the training process on large scale datasets. This approach is able to reduce the training time while providing high accuracy and performance. We demonstrate the proposed approach on a large subset of data extracted from the Aegean Wi-Fi Intrusion Dataset (AWID). Multiple classification experiments were performed on both CPU and GPU. We achieve up to 65x acceleration of training several machine learning methods by moving most of the pipeline computations to the GPU and leveraging the new cuML library as well as the GPU version of the CatBoost library.

CCS CONCEPTS

• **Security and privacy → Intrusion detection systems**; • **Networks** → *Network security*; • **Computing methodologies → Ensemble methods**.

KEYWORDS

Network intrusion detection, classification, GPU, Wi-Fi networks

ACM Reference Format:
Alina Lazar, Alex Sim, and Kesheng Wu. 2021. GPU-based Classification for Wireless Intrusion Detection. In *Proceedings of the 2021 Systems and Network Telemetry and Analytics (SNTA '21), June 21, 2021, Virtual Event, Sweden.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3452411.3464445

1 INTRODUCTION

In today's interconnected world, when Internet is becoming an important tool for all communication aspects of almost everyone, cyberattacks are extending and becoming more diverse and sophisticated. Cyberattacks affect not only big businesses and organizations, but also target individual people and the digital services they use including social interactions, internet banking, online education, shopping, and e-health [18]. Malicious activities including illegal data access, stolen credential, impersonation, data alteration are spreading all over the cyber world at an alarming rate.

Due to rapid spreading of cyberattacks, automated NIDSs are integrated in most computer network systems. Security analysts are using machine learning to automatically monitor their organizations' systems in order to prevent cyberattacks. Correctly identifying malicious network transfers is just the first step of providing reliable, high quality, uninterrupted network services. As cyberattacks are becoming more sophisticated and cyber criminals are developing powerful machine learning based malware, improving the NIDS is an urgent necessity.

Even if machine learning models have been proven successful to automate the attack detection [4, 10, 11, 14, 20, 21] the size, high-dimensionality and non-linearity characteristics made the process harder. Training efficient and performant machine learning models requires large, labeled datasets [13, 22]. Furthermore, intrusion detection models are usually evaluated using labeled datasets that contain normal and attack network traffic patterns. Old intrusion datasets quickly become outdated because of evolving attack tactics and models have to be updated very often. The datasets used for training must contain instances of the mosr recent types of attacks as well as genuine network traffic to enhance the detection accuracy of NIDS. The adaptability of ML techniques to a new environment is a useful characteristic for security applications.

NIDS systems monitor the passive measurements extracted from existing network traffic. Captured network packets or information collected using networking software systems such as Netflow [9], Wireshark [19] or *tstat* [15] could all be used to build intrusion detection datasets. These datasets are usually noisy, contain missing values, include features that are highly correlated and are highly imbalanced.

The volume size has a direct impact on the training time, meaning that very large datasets may become unfeasible to train if we do not find and apply scalable strategies. For this reason, we present in this work a study exploring the large-scale implementation of multiple classification algorithms available in the open GPU data science library named RAPIDS AI. This package provides efficient implementations of classification algorithms that run on NVIDIA GPUs. Using the RAPIDS AI implementations, we are able to train classification algorithms over large intrusion detection datasets up to 65x faster than conventional CPU versions.

Data scientists build machine learning pipelines using sets of different tools. Usually, one set of tools is employed for data extraction, cleaning and preprocessing and another one for training models,

SNTA '21, June 21, 2021, Virtual Event, Sweden

ACM ISBN 978-1-4503-8386-8/21/06…$15.00
https://doi.org/10.1145/3452411.3464445

visualizations and post-processing. Scikit-learn [16] is one of the open source libraries that integrates tools for all the steps in a machine learning pipeline. The newer RAPIDS AI cuML open source library implements many of the machine learning algorithms part of the Scikit-learn library as accelerated versions for the GPUs. This library provides the tools to extend the GPU's computing power to the traditional machine learning tasks by using massive parallelization. Now, complete pipelines, including datasets integration, preprocessing, feature engineering, machine learning modeling, inference, and visualization can be moved and ran faster on GPUs.

This paper includes the following sections. Section 2 presents a short literature review on NIDS research approaches using machine learning and compares our approach with previous work. Section 3 provides an overview of the AWID datasets. Section 4 describes the classification methodology for the GPU. Section 5 thoroughly discusses and compares the experiments performed on CPU using Scikit-learn with the experiments performed on GPU using RAPIDS cuML. Last section 6 draws conclusions and highlights possible future developments of this work on NIDS.

2 BACKGROUND

The Aegean WiFi Intrusion Dataset (AWID), an intrusion detection datasets, comprises of WLAN traffic in flow format including both malicious and normal data records, collected from a real network environment. In a small office simulated setting, Kolias et al. [12] performed sixteen popular types of network attacks to collect a series of large datasets. In the AWID datasets, each instance includes 155 features plus the class label, and the attributes have mostly numerical values. The AWID datasets are described in details in the paper [12] that also reports the accuracy results of classifying attacks using several machine learning algorithms. Manually performing feature selection and reducing the number of attributes from 155 to 20, resulted in an increased accuracy and precision for most classifiers. Their experiments show "impersonation" and "flooding" attacks are harder to classify in comparison to the "injection" attack instances.

An intrusion detection survey focusing on Random Forest (RF) models was described by Resende in [17]. The RF ensemble type algorithm is a good choice to implement NIDS because of it's ability to overcome overfitting and to perform well on imbalanced datasets, with missing values and with large number of attributes. However, the main limitation of the RF algorithm is the high computational complexity, especially when using a large number of trees.

Using the same AWID dataset, Bhandari et al. [1] showed that by selecting a subset of important features of size 10% of the initial set, the training time can be reduced by 4x, while improving the discriminating ability to identify attack instances. This study employed the SHAP method for feature selection and concluded that by selecting a small subset of the most important features, the algorithm is able to important patterns to discriminate between the "attack" instances and the "normal" instances.

A hybrid method combining principal component analysis (PCA)-firefly based machine learning method with the gradient boosting algorithm XGBoost was proposed in [2] with the goal to classify intrusion detection system (IDS) datasets. The hybrid PCA-firefly algorithm is used to perform dimensionality reduction and select a subset of features. The XGBoost algorithm is trained on the reduced dataset for classification. The method was demonstrated on a dataset with 43 features and 125,973 instances and ran on Google Colab using GPUs, but no training times or comparison with the CPU timings were reported in this paper.

Our approach is similar to other previous approaches using traditional and tree-based machine learning methods. However, in this work we compare the training performance of these algorithms on CPU and GPU. We show that the training on the GPU can provide a speed-up between 2x and 65x depending on the algorithm.

Table 2: Accuracy Metrics for CPU Experiments

	Training	Prediction
Naive Bayes	$O(np)$	$O(p)$
KNN	-	$O(np)$
Linear SVM	$O(p^2n + n^3)$	$O(n_{sv}p)$
RF	$O(n^2pn_{trees})$	$O(pn_{trees})$
Gradient Boost	$O(npn_{trees})$	$O(pn_{trees})$

3 AWID DATASET

The AWID publicly available collection of datasets, contain both normal and attack real network flows [12]. There is a full dataset and a reduced dataset. Both full and reduced sets data are divided into two subsets one called training (AWID-CLS-F-Trn, AWID-CLS-R-Trn) and one called testing (AWID-CLS-F-Tst, AWID-CLS-R-Tst). for this paper we combined two files part of the AWID-CLS-F-Tst set to obtain a new dataset with over 8,000,000 instances. For this new dataset, the distribution of the 4 classes is shown in Table 1. Same as for the reduced AWID dataset, the ratio between the number of "normal" and "attack" instances is 14:1. The high ratio between normal and attack instances signifies the dataset is highly imbalanced, fact that makes it hard to classify especially when using small training sets.

Table 1: Data Distribution in Terms of Type of Attacks

	Normal	Injection	Impers.	Flooding
Large AWID	8,336,139	84,741	461,707	10,134

3.1 Data Preprocessing

The AWID datasets contain 154 features and the class. The majority of features are of numeric data type, with different value ranges. Therefore, the scaling step is one of the most important pre-processing steps and can significantly improve the performance of any kind of machine learning algorithms. Training dataset sizes were set as powers of 2, ranging between 2^{12} and 2^{23}. The test dataset was fixed at 2^{18}. We ran the experiments in a loop based on the training dataset size. Inside the loop, first the data was divided into training and testing, second scaling was applied to both training and testing, third the model was trained and last prediction was done on the testing dataset to evaluate the model and get the accuracy.

4 METHODS

4.1 Multinomial Naive Bayes, K-Nearest Neighbor Classification and Random Forest Methods

Multinomial Naive Bayes is the baseline classification method for our experiments. Naive Bayes classifiers are considered simple "probabilistic classifiers" that assume statistical independence between the features. They are among the simplest models, but can achieve high accuracy levels for certain problems. This algorithm is implemented in both Scikit-Learn and cuML.

K-Nearest Neighbor (KNN) classification is another very simple classification method that involves no training step. An test instance is directly classified by the majority vote of its neighbors, with the instance classified to the class most common among its neighbors.

Support vector classification (SVC) [6] is a set of supervised learning methods, based on a quadratic optimization problem, that only uses a subset of training points to build the decision function. The kernel trick is used to transform the data into a high-dimensional space. The assumption is that in the high-dimensional space the data becomes linearly separable and therefore easier to classify.

The Random Forest RF algorithm [3] is based on combining the predictions of many small decision trees using the bagging method for better performance. The small sub-trees, part of the model, are trained on smaller random sub-sampling of the initial training dataset. Only randomly selected subsets of features are used for calculating the information gain to split the nodes, hence the name of the algorithm. Because RFs are built on many small decision trees, is easy to rank the features in terms of their importance.

4.2 Gradient Boosting Algorithms

The gradient boosting algorithms are improvements over the classical decision trees (DT) based on ensemble methods. They are implemented on the idea of selecting the best feature based on some impurity measure, usually calculated as the information gain, and splitting the node for that feature. The only difference between algorithms consists in the objective function, that is minimized during gradient descend part of the algorithm.

Standard gradient boosting methods solve the over-fitting problem of DT, but not very efficiently. To overcome gradient tree boosting shortcomings, Chen [7] proposed the XGBoost algorithm, an optimized gradient boosting algorithm. XGBoost uses LASSO and Ridge regularization techniques to improve the optimization part. These improvements allow XGBoost to be faster and more robust during training compared to DT and RF algorithms.

CatBoost [8], another gradient boosting implementation, includes a step of processing categorical features using permutation techniques and target-based statistics. CatBoost successfully handles categorical features during training time and eliminates the need of converting them before training. The algorithm uses a new schema for calculating leaf values when selecting the tree structure, that also reduces overfitting.

4.3 Algorithm Complexity

The upper bound complexity of the machine learning algorithms described above, for training and prediction, is shown in Table 2.

Table 3: Accuracy Metrics for CPU Experiments

	Acc.	Prec.	Recall	F1
MultinomialNB	53.67	41.29	26.67	20.87
KNN	-	-	-	-
SVC	100	100	100	100
RF	95.70	35.72	48.9	39.45
XGBoost	100	100	100	100
CatBoost	100	100	100	100

Table 4: Accuracy Metrics for GPU Experiments

	Acc.	Prec.	Recall	F1
MultinomialNB	69.72	45.51	29.15	24.98
KNN	93.38	27.12	48.46	28.13
SVC	100	100	100	100
RF	97.51	42.92	49.30	45.53
XGBoost	100	100	100	100
CatBoost	100	100	100	100

In this table n is the number of training instances, p is the number of features, n_{tress} is the number of trees and n_{sv} is the number of support vectors. It is clear that the SVC method is the has the most computational expensive training step. This can be observed on Figure 1, that shows we could only train SVC on less than 500,000 instances.

5 EXPERIMENTS AND RESULTS

5.1 Hardware and Software Infrastructure

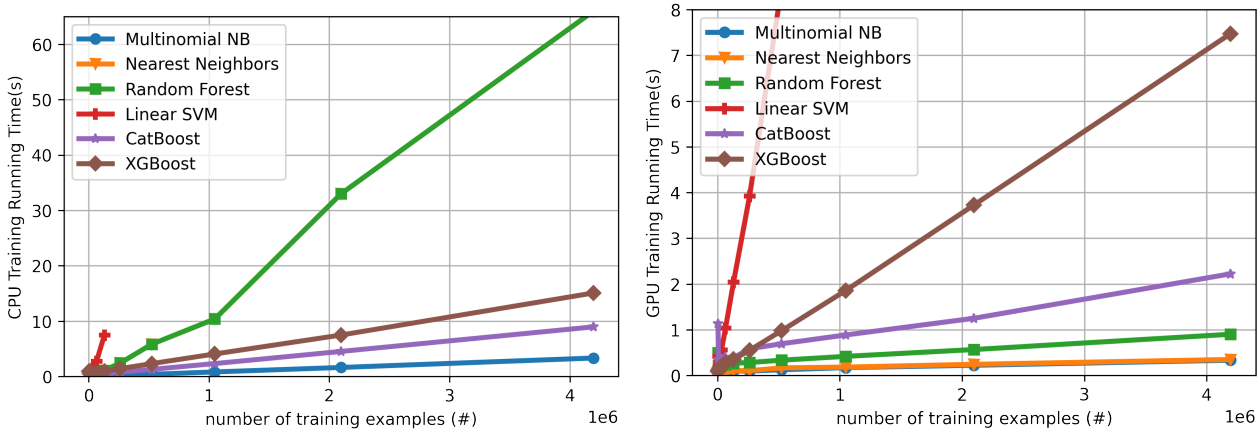
Our experiments were performed on computing nodes at the Ohio Supercomputer Center [5]. These computing nodes have 48 CPU cores, 384 GB of memory, and 2 NVIDIA Volta V100 GPUs. We used a RAPIDS 0.18 conda environment including the cuDF and cuML libraries together with NVIDIA libraries.

5.2 Benchmark Training Step

We applied a variety of classification methods including Multinomial Naive Bayes, K-Nearest Neighbors, Random Forest, Liniar SVM, CatBoost, XGBoost, on the large extracted AWID dataset. Accuracy, precision, recall and training time were calculated on both CPU and GPU runs. As shown in Table 3 and Table 4, the most accurate models was found to be SVC, CatBoost and XGBoost, that had an accuracy of 100% on both CPU and GPU runs.

To illustrate the performance benefits of the cuML library, we compare cuML's algorithms with scikit-learn's algorithms. Scikit-learn only interfaces with the serial version of these machine learning algorithms. We also included experiments with CatBoost, the method that has a CPU and GPU implementation.

Figure 1a shows training performance using Scikit-learn algorithms on CPUs. It is worth notice that even for a large dataset with over 4,000,000 instances, Multinomial Naive Bayes, XGBoost and CatBoost training takes less than 20s. Random Forest takes around 65s or more, that is 3x as much as Naive Bayes, XGBoost and CatBoost. Linear SVC was trained on around 100,000 instances maximum because of the long time it takes to train. No results were



(a) CPU **(b) GPU**

Figure 1: Training Running Time Comparison on CPU and GPU

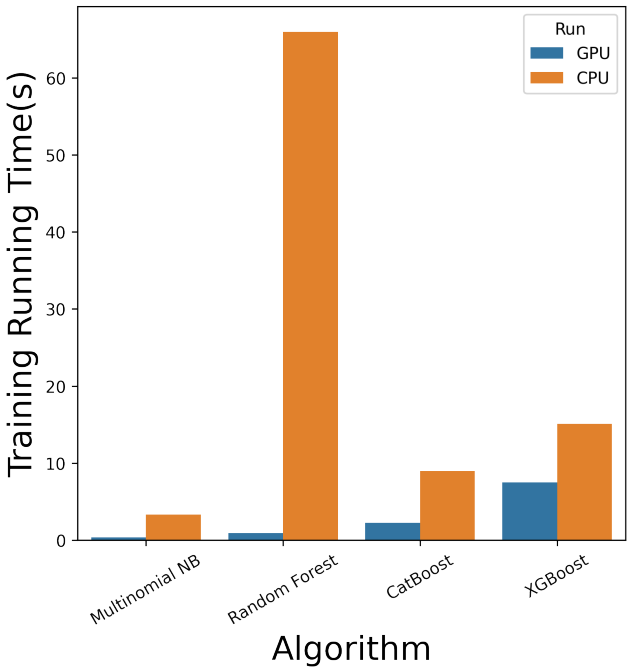


Figure 2: Training running time comparison on CPU and GPU side by side for several machine learning algorithms

obtained on the K-Nearest Neighbors Classifier because the large size of the testing set.

Figure 1b shows training performance obtained using the RAPIDS cuML algorithms and CatBoost, on GPU. In this case, when training on the same large dataset with over 4,000,000 instances, Multinomial Naive Bayes, K-Nearest Neighbors Classifier, Random Forest and CatBoost training took less than 3s. XGBoost trained in around 7s or around 3x more compared to Naive Bayes, Random Forest and CatBoost. Linear SVM performs the worst among all the algorithms, taking at least 8x more than every other algorithm, even for smaller (262,144 instances) training sets. The GPU implementation of K-Nearest Neighbors Classifier performs very well because of the intrinsic parallelism characteristic of this algorithm.

Figure 2 compares the training performance on CPU versus GPU for Multinomial Naive Bayes, Random Forest, CatBoost, XGBoost for a training dataset with over 4,000,000 instances. It is worth noting that the largest performance gap is seen for Random Forest, where the GPU version improves by 65x. For the other algorithms the improvement varies between 2x and 5x.

6 CONCLUSIONS

In this paper we present a scalable machine learning workflow to speed-up network intrusion detection and attacks classification over large 5G datasets. We use the RAPIDS AI cuML library and the CatBoost library to compare these implementations with classical scikit-learn CPU implementations. The results show a speedup up to 65-fold on GPUs. RAPIDS cuDF and cuML libraries offer all necessary tools to easily scale training when GPU resources are available. This pipeline can be adapted to other intrusion detection tasks processing and interpretation tasks, aiming to provide efficient and scalable solutions to many applications.

RAPIDS cuML learning algorithms easily scale training when GPU resources are available. The proposed pipeline may be adapted to other intrusion detection datasets to provide efficient and scalable solutions for these important applications. It is well-known that deep learning training benefits from the GPUs computing characteristics. Therefore, we plan to compare results from tree based classifiers with results from deep learning approaches.

ACKNOWLEDGMENTS

This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. This research used

resources of the National Energy Research Scientific Computing Center and the Ohio Supercomputer Center.

REFERENCES

[1] Shilpa Bhandari, Avinash K Kukreja, Alina Lazar, Alex Sim, and Kesheng Wu. 2020. Feature Selection Improves Tree-based Classification for Wireless Intrusion Detection. In *Proceedings of the 3rd International Workshop on Systems and Network Telemetry and Analytics*. 19–26.

[2] Sweta Bhattacharya, Praveen Kumar Reddy Maddikunta, Rajesh Kaluri, Saurabh Singh, Thippa Reddy Gadekallu, Mamoun Alazab, Usman Tariq, et al. 2020. A novel PCA-firefly based XGBoost classification model for intrusion detection in networks using GPU. *Electronics* 9, 2 (2020), 219.

[3] Leo Breiman. 2001. Random Forests. *Mach. Learn.* 45, 1 (Oct. 2001), 5–32.

[4] Liwei Cao, Xiaoning Jiang, Yumei Zhao, Shouguang Wang, Dan You, and Xianli Xu. 2020. A Survey of Network Attacks on Cyber-Physical Systems. *IEEE Access* 8 (2020), 44219–44227.

[5] Ohio Supercomputer Center. 1987. Ohio Supercomputer Center. http://osc.edu/ark:/19495/f5s1ph73

[6] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)* 2, 3 (2011), 1–27.

[7] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. ACM, New York, NY, USA, 785–794.

[8] Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. 2018. CatBoost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363* (2018).

[9] Cristian Estan, Ken Keys, David Moore, and George Varghese. 2004. Building a better NetFlow. *ACM SIGCOMM Computer Communication Review* 34, 4 (2004), 245–256.

[10] Yogita Hande and Akkalashmi Muddana. 2020. A Survey on Intrusion Detection System for Software Defined Networks (SDN). *International Journal of Business Data* (2020).

[11] Khalid Khan, Amjad Mehmood, Shafiullah Khan, Muhammad Altaf Khan, Zeeshan Iqbal, and Wali Khan Mashwani. 2020. A survey on intrusion detection and prevention in wireless ad-hoc networks. *Int. J. High Perform. Syst. Archit.* 105 (May 2020), 101701.

[12] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Stefanos Gritzalis. 2016. Intrusion Detection in 802.11 Networks: Empirical Evaluation of Threats and a Public Dataset. *IEEE Communications Surveys Tutorials* 18, 1 (2016), 184–208.

[13] Alina Lazar, Kesheng Wu, and Alexander Sim. 2018. Predicting Network Traffic Using TCP Anomalies. *Conference on Big Data (Big Data)* (2018).

[14] Mohammad Masdari and Hemn Khezri. 2020. A survey and taxonomy of the fuzzy signature-based Intrusion Detection Systems. *Appl. Soft Comput.* 92 (July 2020), 106301.

[15] Marco Mellia, Michela Meo, Luca Muscariello, and Dario Rossi. 2008. Passive analysis of TCP anomalies. *Computer Networks* 52, 14 (2008), 2663–2676.

[16] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[17] Paulo Angelo Alves Resende and André Costa Drummond. 2018. A Survey of Random Forest Based Methods for Intrusion Detection Systems. *ACM Comput. Surv.* 51, 3, Article 48 (May 2018), 36 pages. https://doi.org/10.1145/3178582

[18] Gaganjot Kaur Saini, Malka N Halgamuge, Pallavi Sharma, and James Stephen Purkis. 2020. A Review on Cyberattacks: Security Threats and Solution Techniques for Different Applications. In *Cyber Warfare and Terrorism: Concepts, Methodologies, Tools, and Applications*. IGI Global, 98–126.

[19] Chris Sanders. 2017. *Practical packet analysis: Using Wireshark to solve real-world network problems*. No Starch Press.

[20] Geeta Singh and Neelu Khare. 2021. A survey of intrusion detection from the perspective of intrusion datasets and machine learning techniques. *International Journal of Computers and Applications* (2021), 1–11.

[21] Nasrin Sultana, Naveen Chilamkurti, Wei Peng, and Rabei Alhadad. 2019. Survey on SDN based network intrusion detection system using machine learning approaches. *Peer-to-Peer Networking and Applications* 12, 2 (March 2019), 493–501.

[22] Astha Syal, Alina Lazar, Jinoh Kim, Alex Sim, and Kesheng Wu. 2019. Automatic detection of network traffic anomalies and changes. In *Proceedings of the ACM Workshop on Systems and Network Telemetry and Analytics*. 3–10.