A Cost-based Approach for Fast Intrusion Detection

Jiarui Gao[1]
[1]School of Computer Science, Shanghai Key Laboratory of Intelligent Information Processing,
Fudan University, China
jrgao14@fudan.edu.cn

ABSTRACT

The aim of real time Intrusion Detection Systems(IDSs) is to maximize detection accuracy as well as minimize corresponding costs. In this paper, I present a cost-based approach utilizing decision tree model and neural network models with GPU acceleration for fast intrusion detection. In this approach, both computational cost and time cost are considered, and different models are trained with respect to the types of protocols and services. Empirical experiments are carried out on off-line benchmark dataset NSL-KDD.

KEYWORDS

Neural network; Intrusion detection; anomaly detection; cost analysis.

ACM Reference format:
Jiarui Gao[1]. 2017. A Cost-based Approach for Fast Intrusion Detection . In *Proceedings of* , , , 8 pages.
DOI: http://dx.doi.org/xxx

1 INTRODUCTION

With the development of computing and network technology, Internet has became an important part of our daily life. However, this popularity has also brought many security issues. Intrusion Detection Systems(IDSs) are designed to detect attacks. Specifically, they help to discover, determine, and identify unauthorized use, duplication, alteration, and destruction of information systems[27]. Additionally, IDSs are able to monitor the system and alert network administrators if necessary.

With respect to the types of detectable attacks, there are mainly three types of IDSs: misuse-based, anomaly-based, and hybrid[5] IDSs. Misuse-based IDSs, which are also called signature-based IDSs, are designed to detect known attacks by comparing new packets and the predefined signatures stored in the database. Anomaly-based IDSs aims at modeling the normal network, and anomalies can be detected when happened. Hybrid IDSs combine the former two techniques together.

Also with respect to the using environment, IDSs can be divided as the ones used in real-time environment or used in off-line environment. Off-line IDSs have been studied extensively recently especially based on data mining and machine learning techniques, such as K-means[23] and ANN[15]. And excellent detection results with high accuracies and low false alarm rates on the off-line datasets are accomplished.

However, these IDSs always fail to provide real-time detection, which means they can only detect intrusions after the they already happened for a while and are not capable of defending the host from attacks at all. This is because the excellent off-line performance is often achieved thanks to the comprehensive extracted features and long time complicated computing, which are impossible to get in real-time environment. So a balance need to be maintained between accuracy and efficiency in real-time IDSs.

In this paper, the aim is to develop an IDS which can be utilized in real-time environment with high detection accuracy and low costs. For the minimization of computational cost, the traditional machine learning models are replaced by neural network models, which enable the utilizing of GPU acceleration. Also, the time cost is considered, and similar with the time cost theory in [20], low level features are utilized with high priority. The benchmark dataset NSL-KDD is used as testbed for evaluation, which is a new version of KDD Cup 1999 dataset[10] and significant improvement has been made compared with original KDD Cup 1999 dataset[34].

The rest of the paper is organized as follows. Sec 2 gives a detailed review of current related work about intrusion detection. In Sec 3, the new approach for intrusion detection is proposed. The experimental results are reported in Sec 4. More future work is presented in Sec 5. Finally, a conclusion is drew in Sec 6.

2 RELATED WORK

2.1 Intrusion Types

There are many types of intrusions. According to KDD Cup 1999 dataset[34], all the simulated attacks are classified into four categories:

- **Denial of Service Attack (DoS):** This type of attacks aim at making the services of target computer unavailable to its legitimate users by increasing the number of resource demanding processes to exhaust the target's network bandwidth , CPU load, or memory etc[1]. However, with the development of detection and IP tracking techniques, old single source attacks can be easily detected and the source can be rebuffed or shut down. In recent years, many kinds of distributed Dos(DDos) attack techniques have been developed. One of the effective approaches to detect DDos attacks is to capture and recognize patterns of the DDos affected traffic[31]. This survey[38] is recommended for more information.
- **User to Root Attack (U2R)** This type of attacks refer to the behavior that the attacker tries to gain root access of the system. Usually the attacker enters the system as a

normal user first and tries to exploit some vulnerabilities in the system to gain illegal root access.

- **Remote to Local Attack (R2L):** This type of attacks occur when an attacker tries to gain local access as a legitimate user on a remote server where actually he or she has no legitimate accounts. R2L attacks are often carried out by sending packets over network to the remote server[28]. Usually, this illegal accesses are gained by taking advantages of vulnerabilities on the server.
- **Probing Attack:** This type of attacks are basically an information collection behavior. The attacker scans the computers in a specific network to gather information or to exploit vulnerabilities. For more information about IDSs specially designed for probing attacks, this work [37] is recommended.

2.2 Detection Methodology

Basically, there are three types of intrusion detection techniques: misuse-based, anomaly-based, and hybrid[5]. In the following parts, a brief introduction is presented.

2.2.1 Misuse-based Intrusion Detection. The misuse-based intrusion detection is also known as signature-based or knowledge-based intrusion detection. A signature is a pattern or string that corresponds to a known attack or threat[21]. So in the misuse-based IDSs, the detection procedure for new coming packets is to compare them with the known signatures to recognize possible intrusions.

The main advantage of misuse-based methods is that the method is effective for detecting known types of attacks and the detection results are free from false alarms. However, there are also some disadvantages. Firstly, it's usually hard to keep the signature database up to date. Secondly, this type of IDSs are incapable in detecting zero-day attacks or any attacks whose signatures are not included in the database, so their abilities are quite limited.

2.2.2 Anomaly-based Intrusion Detection. An anomaly is a deviation from a known behavior[21]. The anomaly-based IDSs aim at modeling normal status including regular activities, normal network connections, hosts and users etc. And new packets different from normal status can be detected, the anomalies are considered to be an attack.

One of the most attracting advantages of anomaly-based methods is that they have the abilities to recognize zero-day attacks. Another advantage is that unlike the misuse-based methods, the profiles of normal activities are customized for every system, application, or network, thereby making it difficult for attackers to know which activities they can carry out undetected[5]. The main disadvantage of anomaly-based techniques is the they may present high false alarm rates, since every behavior that has not happened before has the potential risk to be considered as an intrusion by this type of IDSs.

2.2.3 Hybrid Intrusion Detection. Hybrid techniques combine the former two types of techniques. They are implemented aiming at decreasing the false alarm rate as well as detecting zero-day attacks. Actually, pure anomaly-based IDSs are rarely used.

2.3 Feature Selection

The reduction of features and feature selection are very important issues in increasing the efficiency of intrusion detection.

This work[6] aims at reducing the features used for misuse-based IDSs, since the comparison between the whole packet and all the signatures can be time demanding and actually redundant. Also, useful feature selection algorithms are proposed in the paper. Furthermore, the features on one specific dataset KDD Cup 1999 dataset are analyzed by comparing the relevance between each features and information gain is employed to find the most discriminating features for each class[16].

Unsupervised machine learning techniques such as PCA[26][19] and K-means[18] are employed for feature selection process. And all of these methods demonstrate good performance on reducing the dimension of features and reserving only the most representative ones.

However, since nearly all of these feature selection methods are used in off-line environment, they ignore the time cost brought by the feature extraction process. So the proposed selected features may bring severe time delay. In the cost-based approach presented in this paper, the time cost factors are considerer, and a balance between efficiency and accuracy is maintained.

2.4 Machine Learning Approaches

In this section, I present a brief introduce of the utilization of machine learning techniques in intrusion detection. Specific machine learning algorithms are beyond the scope of this paper, and this book[2] is highly recommended for more knowledge on pattern recognition and machine learning.

2.4.1 Supervised Learning. Supervised learning can be further divided into two classes: generative models and deterministic models. Both classes contain plenty of proposed models. Some of the popular supervised learning methods are listed as follows: Artificial Neural Network, Bayesian Network, Decision Tree, Genetic Algorithms, Hidden Markov Models, Naive Bayes, Support Vector Machine etc.

Nearly all the mentioned techniques in supervised learning are used in intrusion detection, only a few papers are listed here[15][33][30]. Comprehensive computational complexity analysis and concrete introduction results can be found in [5] and [35].

2.4.2 Unsupervised Learning. One of the most frequently used unsupervised learning techniques is **clustering**. This method aims at clustering the data with similar properties together. The main advantage of clustering is that it makes the labeling work not necessary. Specifically, for intrusion detection task, this type of IDS can learn from audit data without labeling or requiring the network administrator to provide concrete information about the attacks.

There are many clustering methods that demonstrate excellent performance during evaluation[14][23][18][36]. K-means, K-NN are the most frequently used unsupervised learning approaches. Also, some work combine K-means and K-NN together and better results are achieved[23]. However, as mentioned in Sec 2.3, feature selection and reduction are urgently needed in methods like K-NN, because computing the similarities between high dimensional features are really expensive.

2.4.3 *Ensemble Learning.*

Actually, the ensemble learning approaches do not belong to supervised or unsupervised learning. Basically, the idea of ensemble learning is to use multiple weak models to build a strong model. This is because trying to solve a complicated task with a single model may lead the model to be extremely complex. And it is known that complex models with lots of parameters often tend to over fit on the training data. The aggregation of many weak models can avoid over fitting to some extent. Bagging and boosting are frequently used approaches in ensemble learning, this work[11] gives a good example for utilizing ensemble learning methods in decision tree models.

Bagging[3] is a method of generating multiple versions of a single model and using them to get an aggregated model. The aggregation averages over all the versions of models when predicting a result by majority voting. **Boosting**[12] is based on the idea to concentrate more the mis-classified data of every epoch in the training process. Adaptive Boosting (AdaBoost)[29] is one of the most frequently used algorithms in boosting. The **Random Forest**[4] is a method that combines the decision trees and ensemble learning. The so-called forest is composed of many decision trees. Also, bagging techniques are employed to avoid over fitting. The random forest has been proved to be effective in many tasks.

Ensemble learning methods are extensively utilized in intrusion detection[39][28]. The type of single models used for the aggregation can be nearly all kinds of previous mentioned machine learning models. For example, random forest is used both in mis-use based IDS[13] and anomaly-based IDS[39].

In the approach proposed in this paper, both the supervised learning and ensemble learning methods are employed. The combination of decision tree model and neural networks models demonstrates a great improvement compared with using single model for prediction on NSL-KDD dataset.

2.5 Public Datasets

2.5.1 *DARPA 1998 and DARPA 1999 Datasets.*

One of the most commonly used datasets are the Defense Advanced Research Projects Agency (DARPA) 1998 and DARPA 1999 datasets[25][24]. Both of the datasets are created by the Cyber Systems and Technology Group of the Massachusetts Institute of Technology Lincoln Laboratory (MIT/LL). The datasets contain both network and operating system data. Totally, seven weeks of training data and two weeks of test data are collected, and attack simulations are organized during both of the periods. Notice that for anomaly detection, there are some types of attacks only appear in the testing data.

2.5.2 *KDD Cup 1999 Dataset.*

KDD Cup 1999 dataset[34] is also one of the most widely used intrusion detection datasets, which is created for the KDD Cup challenge in 1999. The dataset is based on DARPA 1998 TCP/IP dataset and some basic features are extracted. Also some more advanced features like some content features within a connection and traffic features computed using a two-second time window are computed. Totally, 41 features are extracted for a single connection.

2.5.3 *NSL-KDD Dataset.*

As suggested in [34], the original KDD Cup 1999 dataset has two important shortcomings which highly affect the performance of evaluated systems, and result in a very poor evaluation of anomaly detection approaches.The NSL-KDD dataset consists of only the selected records of the complete KDD Cup 1999 dataset and free from the mentioned issues in [34].

2.5.4 *ADFA Intrusion Detection Datasets.*

The ADFA Intrusion Detection Datasets contain data both generated from Linux and Windows[8] systems. Different from former introduced datasets, ADFA contains raw system call traces. A new host-based anomaly intrusion detection methodology is evaluated on ADFD[9], which utilizing discontiguous system call patterns to increase detection rates whilst reducing false alarm rates.

2.5.5 *UNB ISCX 2012 Intrusion Detection Evaluation Dataset.*

This dataset is proposed in [32]. According to [32], this labeled dataset consists of realistic network and traffic with complete capture. Also, diverse intrusion scenarios are included. This dataset demonstrate a good performance in the comparison with other existing datasets.

3 METHODOLOGY

As shown in Fig 1, a multi-model detector is presented here. Different neural network models are used for packets with different protocol types and services based on a decision tree model. Additionally, neural network models are trained for each level of features respectively.

3.1 Cost Analysis

3.1.1 *Computational Cost.*

One of the major challenges for real-time IDSs is the computational cost attached with the detection procedure. For example, the well-known K-nearest Neighbor Classifier(KNNC)[22] performs well on off-line datasets, but it is not an appropriate detector in real time. Because the KNNC calculates the similarity between the new packet and every packet in the training set to decide the label of the new packet. So KNNC is computationally expensive when the incoming number of packets is huge, and the system may be overloaded.

Traditionally, reducing the dimension of features and number of parameters in detection models will lead to lower computational cost. However, this may also result in lower detection accuracy or higher false alarm rate. Different from former approaches, in utilizing deep neural networks, the computing tasks can be accelerated on GPU[7]. Also, the ability of neural network models to represent any non-linear function in very few hidden layers and hidden neurons makes the computational expense even less.

3.1.2 *Time Cost.*

It is obvious that the time delay for retrieving features simply from a network packet and extracting information from all former similar connections within a time period are quite different. Different from[20], the connection based features are utilized according to[20] in this paper. The exact levels of features are presented in Tab 1.

- Level 1 features are basic features of individual TCP connections.
- Level 2 features are content features within a connection suggested by domain knowledge.
- Level 3 features are traffic features computed using a two-second time window.

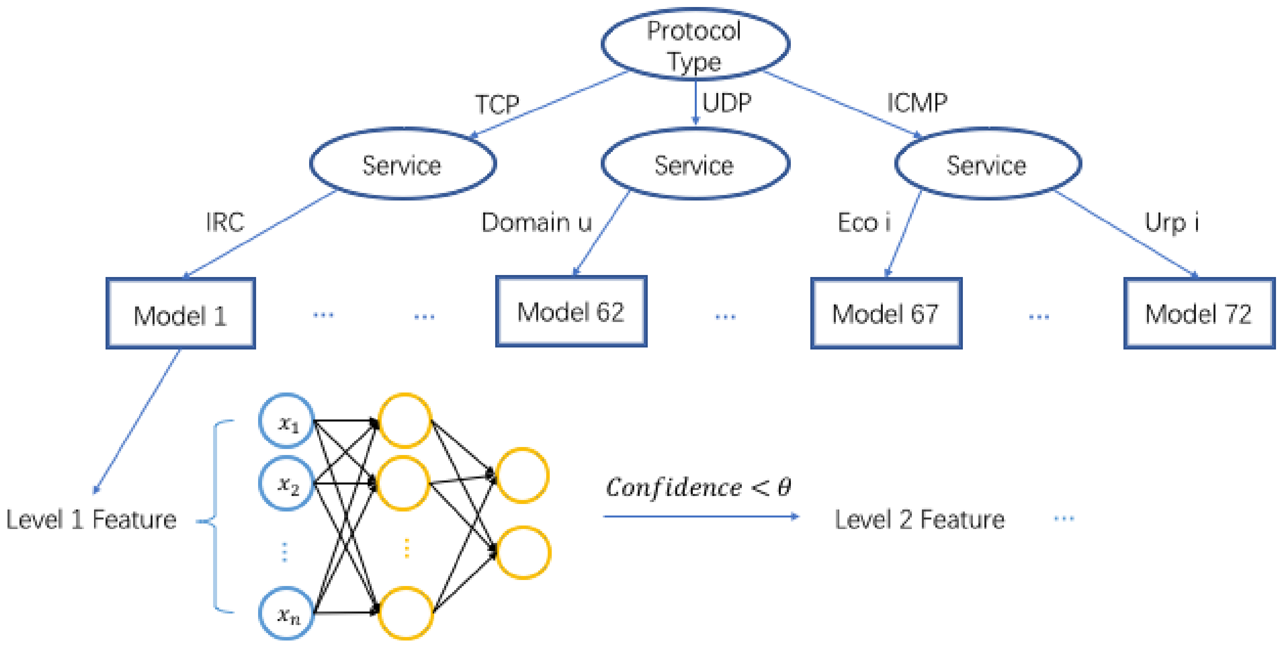

Figure 1: An overview of the detection framework including the decision tree model(top part) and the neural network models(bottom part). The sequence number of models are counted based on NSL-KDD dataset.

Table 1: Features in Different Levels.

Level 1 Features	Level 2 Features	Level 3 Features
Duration	Hot	Count
Protocol type	Num failed logins	Serror rate
Service	Logged in	Rerror rate
Src bytes	Num compromised	Same srv rate
Dst bytes	Root sell	Diff srv rate
Flag	Su attempted	Srv count
Land	Num root	Srv serror rate
Wrong fragment	Num file creations	Srv rerror rate
Urgent	Num shells	Srv diff host rate
	Num access files	
	Num outbound cmds	
	Is hot login	
	Is guest login	

Notice that the Level 3 features require iteration from the entire former two-second packet set, the time cost is extremely high compared with the lower level of features. Also, all the Level 3 features for connections coming at the same time can also be computed at the same time.

The process of utilizing different levels of features is:

- The lower level features are first used;
- If the confidence of the classification is less than a threshold θ, higher level features are extracted and are used combined with other lower level features;
- If the extracted feature level has already reached 3, the eventual detection result is directly made.

For example, both Level 1 and Level 2 features are considered after extracting the Level 2 features.

3.2 Decision Tree Model

Packets with different protocol types and different services tend to have different signatures to be malicious. Thus different detectors should be trained respectively. For each new packet, its protocol type and service decides which detector to use. This model selection framework is be modeled by a decision tree.

As shown in Fig 1, the decision tree has very intuitive architecture in the proposed approach.

3.3 Neural Network Detectors

Neural network models are the leaf nodes for the decision tree, they are trained for each protocol type and service respectively. Actually due to the less complexity for packets detection, neural network models perform nearly the same as traditional classifiers. The main reason of utilizing neural network models in this paper is the excellent GPU acceleration support for deep learning libraries, which will reduce the computational cost to some extent.

Neural network models with only one hidden layer is utilized in this paper, and the following experiments proves their capability for intrusion detection even if they only have such simple architectures.

3.4 Evaluation Methods

Three evaluation metrics are employed in this paper to evaluate the performance of the intrusion detection system, namely, detection accuracy, false alarm rate and total cost:

- Detection accuracy is calculated by comparing the predicted results with the ground truth labels.
- The false alarm rate(FAR) is defined as Eq 1, in which FP, TN represent false positive and true negative samples respectively.

$$FAR = \frac{FP}{FP + TN} \tag{1}$$

- The total cost is the sum of the computational cost and the time cost.

4 EXPERIMENTS

4.1 Dataset

As mentioned in Sec 2.5, NSL-KDD dataset consists of the selected records of the KDD Cup 1999 dataset, which makes it more challenging to be detected compared with the original widely used benchmark KDD Cup 1999 dataset. The NLS-KDD dataset solves the issues of models trained on KDD Cup 1999 dataset, which have high seemingly accuracy but actually poor capability of anomaly detection.

All the features in NSL-KDD dataset are considered in this paper except the additional feature: detection difficulty, which is generalized by statistical analysis for the KDD Cup 1999 dataset[34]. Totally, there are 61 different TCP services, 5 different UDP services and 6 different ICMP services.

4.2 Parameter Setting

The data preprocessing method z-normalization is employed. All the code for training the neural network models is implemented in Python utilizing deep learning framework Tensorflow, and the neural network models are trained on one GPU GeForce GTX TITAN X for 5000 epochs with early stop policy. All the neural network models consist of 1 hidden layer, and the number of hidden neurons are set specifically for every model. The initial learning rate is $\eta = 0.0001$ and dropout ratio is set to be 0.75 to avoid over fitting. Eventually, a softmax layer is utilized to calculate loss and Adam[17] is employed to accelerate learning. The final prediction results of the models are the values of the two output neurons.

4.3 Baseline

Support Vector Machine(SVM) is employed as a baseline method for each model respectively. The code are implemented in Python utilizing Scikit-learn Library. And the following baseline results are evaluated on the same server with the proposed approach without GPU support for SVM method. Also, empirically, the SVM models are also combined with the same decision tree model in the proposed approach to accomplish a comparative result.

4.4 Detection Results

Due to the coming deadline issue, I have only finished part of the experiments.

4.4.1 Detection Accuracy. Tab 2 demonstrates the detection accuracy for each level of features. And apart from the over all accuracy, I take three representative protocol types and services(TCP finger/UDP domain_u/ICMP eco_i) as examples.

Observed from Tab 2, we can see that SVM models and neural network models perform nearly the same on most of the detections. But neural network models demonstrate a good property on the three examples that as the level of features increases, the accuracies never decay; while the SVM method present less stability. Actually, this property is quite important in the proposed approach, because the accuracy decay on higher feature levels will result in some eventual detection results to be wrong. Since the results for packets that cannot be classified confidently by the lower feature level models are all finally decided by the Level 3 feature models. Also, lots of time cost will be wasted for no improvement in detection accuracy.

Also, we can observe that the idea of utilizing decision tree models to detect different protocol types and service with different models is quite reasonable. In Tab 2, huge accuracy improvement has been made thanks to this separation.

4.4.2 False Alarm Rates. Tab 3 demonstrates the false alarm rates for each level of features. As mentioned before, three representative protocol type and service are presented.

As observed, the false alarm rates tends to decay as the increasing level of features. Especially, for the ICMP eco_i models, the false alarm rates decay significantly for Level 3 features. For these type of situations, the extraction of high level features is necessary.

4.4.3 Cost Analysis. Tab 4 demonstrates the **computational cost** for each level of features. The same as former experiments, three representative protocol type and service are selected. Here I report the average time for computing single one packet feature item.

Thanks to GPU acceleration, it can be noticed that significant improvement has been made on computational cost utilizing neural network models, which is quite important in real time detection for the IDSs. Also practically, the number of neurons in the hidden layer is reduced as small as possible for each model, which also save the computational cost.

Maybe you could notice that in the SVM method, the computational cost may even decay as the number of feature dimension

Table 2: Detection Accuracy Results.

Methods/Features	Over all	TCP finger	UDP domain_u	ICMP eco_i
Level 1 Features				
SVM	78.20%	99.26%	99.33%	94.27%
Ours	79.16%	99.26%	99.33%	94.27%
Level 2 Features				
SVM	81.39%	97.06%	99.55%	94.27%
Ours	80.67%	99.26%	99.78%	94.27%
Level 3 Features				
SVM	80.57%	99.26%	99.78%	98.09%
Ours	76.24%	99.26%	99.78%	98.89%

Table 3: Detection False Alarm Rates

Methods/Features	Over all	TCP finger	UDP domain_u	ICMP eco_i
Level 1 Features				
SVM	3.86%	0.00%	0.67%	57.69%
Ours	4.40%	0.00%	0.67%	57.69%
Level 2 Features				
SVM	1.81%	0.00%	0.45%	57.69%
Ours	3.49%	0.00%	0.45%	57.69%
Level 3 Features				
SVM	2.19%	0.00%	0.00%	19.23%
Ours	6.64%	0.00%	0.00%	7.69%

increase. This is because in the SVM model, only the training data on the decision boundary is used in the calculation for prediction, so the actual computational cost does not only depend on the dimension of features. For more theoretical knowledge of SVM, this book[2] is recommended.

Also, **time cost** is considered. Following the process of utilizing different levels of features, new packet is first detected according to its Level 1 features. If the confidence of the classification, which is the larger value of the two output neurons is less than $\theta = 0.8$, than the higher features are extracted and higher level models are used. In the worst case, the packet will eventually get its label through Level 3 models. And more additional computational cost will also be spent at the same time.

The total cost is the sum of computational cost and time cost. The proposed approach is much more efficient compared with simply using Level 3 features for all the packets. Additionally, there is almost no decay of detection accuracies or increment of false alarm rates.

5 FUTURE WORK

5.1 More Experiments

As mentioned in Sec 4, due to the coming deadline, some of the experiments are not completely finished.

The future work for this paper is listed as follows:

- Calculate the average performance for all the models, not for only some examples.
- Measure the specific improvement made by distinguishing different level of features quantitatively, such as calculate the exact cost saved for detecting testing packets.
- Try more specific detection results than just normal and anomaly classification. Actually, the NSL-KDD dataset provides the labels of the specific attack types for each connection.
- The experiments are originally planned to be conducted on the UNB ISCX IDS 2012 dataset, which is the up-to-date dataset for intrusion detection. However, it will take months to get the dataset following the procedures on the official website. But still, more experiments should be conducted on this new dataset before making any conclusions.

5.2 More Issues

There are also some practical issues need to be solved for further implementation:

- There are some protocol types and services with very few training data, such as TCP pm_dump, UDP tftp_u and ICMP urh_i etc. Such a small training dataset will make it impossible to train a neural network model. Maybe some signature-based technique are needed for these services.
- It will cost too much time to train and tune parameters for every model, specifically 72 models in NSL-KDD dataset. But currently I just train the example models manually because the number of hidden neurons are set to be customized for every model.
- Utilizing the features in NSL-KDD makes the detection in this paper to be connection based, which means that the intrusion can only be detected after all the packets in one connection have arrived. Actually, as suggested in[20], some intrusion can be detected only after the first

Table 4: Computational Cost Results.

Methods/Features	Over all	TCP finger	UDP domain_u	ICMP eco_i
Level 1 Features				
SVM	$4.2 * 10^{-4}s$	$4.4 * 10^{-6}s$	$9.2 * 10^{-7}s$	$5.7 * 10^{-7}s$
Ours	$5.4 * 10^{-8}s$	$2.7 * 10^{-6}s$	$4.3 * 10^{-7}s$	$1.4 * 10^{-6}s$
Level 2 Features				
SVM	$5.9 * 10^{-4}s$	$7.3 * 10^{-6}s$	$1.3 * 10^{-6}s$	$1.3 * 10^{-6}s$
Ours	$7.0 * 10^{-8}s$	$2.7 * 10^{-6}s$	$4.6 * 10^{-7}s$	$1.4 * 10^{-6}s$
Level 3 Features				
SVM	$2.3 * 10^{-4}s$	$7.2 * 10^{-6}s$	$4.1 * 10^{-6}s$	$6.2 * 10^{-6}s$
Ours	$1.0 * 10^{-7}s$	$2.8 * 10^{-6}s$	$5.2 * 10^{-7}s$	$1.5 * 10^{-6}s$

packet or only after a few packets have been received, which is capable to reduce the time cost even more. So extracting more packet based features for model training can be considered as another improvement.

- The datasets that are extensively used currently are still the simulations two decades ago, which makes the models analyzed in these papers unsuitable for detecting newly appeared attacks in recent years. So more public datasets are urgently needed in the field of intrusion detection.

6 CONCLUSION

In this paper, a new cost-based approach for fast intrusion detection is proposed. Utilizing both decision tree model and neural network models with GPU acceleration, the proposed IDS is capable of maintaining high detection accuracy and low false alarm rate with the minimization of both computational cost and time cost. The experiments are carried out on NSL-KDD dataset, and a good performance is demonstrated in the evaluation.

However, as mentioned in Sec 5, there are still many issues that needed to be solved for developing IDSs, such as how to detect intrusion only after a few packets have been received, or how to accurately detect zero-day attacks. Also, more attack simulations need to be organized to test the IDSs before they are actually employed in the real environment, since any crash in IDSs may leads to severe problems.

After accomplishing the rest experiments, further research will focus on solving the mentioned issues in Sec 5.

REFERENCES

[1] Monowar H Bhuyan, Hirak Jyoti Kashyap, Dhruba Kumar Bhattacharyya, and Jugal K Kalita. 2014. Detecting distributed denial of service attacks: methods, tools and future directions. *Comput. J.* 57, 4 (2014), 537–556.
[2] Christopher M Bishop. 2006. Pattern recognition. *Machine Learning* 128 (2006).
[3] Leo Breiman. 1996. Bagging predictors. *Machine learning* 24, 2 (1996), 123–140.
[4] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.
[5] Anna L Buczak and Erhan Guven. 2016. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials* 18, 2 (2016), 1153–1176.
[6] Srilatha Chebrolu, Ajith Abraham, and Johnson P Thomas. 2005. Feature deduction and ensemble design of intrusion detection systems. *Computers & security* 24, 4 (2005), 295–307.
[7] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. 2014. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759* (2014).
[8] Gideon Creech and Jiankun Hu. 2013. Generation of a new IDS test dataset: Time to retire the KDD collection. In *Wireless Communications and Networking Conference (WCNC), 2013 IEEE.* IEEE, 4487–4492.
[9] Gideon Creech and Jiankun Hu. 2014. A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns. *IEEE Trans. Comput.* 63, 4 (2014), 807–819.
[10] KDD Cup. 2007. Available on: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. (2007).
[11] Thomas G Dietterich. 2000. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning* 40, 2 (2000), 139–157.
[12] Yoav Freund and Robert E Schapire. 1995. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory.* Springer, 23–37.
[13] Farnaz Gharibian and Ali A Ghorbani. 2007. Comparative study of supervised machine learning techniques for intrusion detection. In *Communication Networks and Services Research, 2007. CNSR'07. Fifth Annual Conference on.* IEEE, 350–358.
[14] Gilbert R Hendry and Shanchieh J Yang. 2008. Intrusion signature creation via clustering anomalies. In *SPIE Defense and Security Symposium.* International Society for Optics and Photonics, 69730C–69730C.
[15] Bhupendra Ingre and Anamika Yadav. 2015. Performance analysis of NSL-KDD dataset using ANN. In *Signal Processing And Communication Engineering Systems (SPACES), 2015 International Conference on.* IEEE, 92–96.
[16] H Günes Kayacik, A Nur Zincir-Heywood, and Malcolm I Heywood. 2005. Selecting features for intrusion detection: A feature relevance analysis on KDD 99 intrusion detection datasets. In *Proceedings of the third annual conference on privacy, security and trust.*
[17] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[18] Vipin Kumar, Himadri Chauhan, and Dheeraj Panwar. 2013. K-means clustering approach to analyze NSL-KDD intrusion detection dataset. *International Journal of Soft Computing and Engineering* 3 (2013).
[19] Shilpa Lakhina, Sini Joseph, and Bhupendra Verma. 2010. Feature reduction using principal component analysis for effective anomaly–based intrusion detection on NSL-KDD. (2010).
[20] Wenke Lee, Salvatore J Stolfo, Philip K Chan, Eleazar Eskin, Wei Fan, Matthew Miller, Shlomo Hershkop, and Junxin Zhang. 2001. Real time data mining-based intrusion detection. In *DARPA Information Survivability Conference & Exposition II, 2001. DISCEX'01. Proceedings*, Vol. 1. IEEE, 89–100.
[21] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. 2013. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications* 36, 1 (2013), 16–24.
[22] Yihua Liao and V Rao Vemuri. 2002. Use of k-nearest neighbor classifier for intrusion detection. *Computers & security* 21, 5 (2002), 439–448.
[23] Wei-Chao Lin, Shih-Wen Ke, and Chih-Fong Tsai. 2015. CANN: An intrusion detection system based on combining cluster centers and nearest neighbors. *Knowledge-based systems* 78 (2015), 13–21.
[24] Richard Lippmann, Joshua W Haines, David J Fried, Jonathan Korba, and Kumar Das. 2000. The 1999 DARPA off-line intrusion detection evaluation. *Computer networks* 34, 4 (2000), 579–595.
[25] Richard P Lippmann, David J Fried, Isaac Graf, Joshua W Haines, Kristopher R Kendall, David McClung, Dan Weber, Seth E Webster, Dan Wyschogrod, Robert K Cunningham, and others. 2000. Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation. In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings*, Vol. 2. IEEE, 12–26.
[26] Guisong Liu, Zhang Yi, and Shangming Yang. 2007. A hierarchical intrusion detection model based on the PCA neural networks. *Neurocomputing* 70, 7 (2007), 1561–1568.
[27] Srinivas Mukkamala, Andrew Sung, and Ajith Abraham. 2005. Cyber security challenges: Designing efficient intrusion detection systems and antivirus tools. *Vemuri, V. Rao, Enhancing Computer Security with Smart Technology.(Auerbach,*

2006) (2005), 125–163.

[28] Srinivas Mukkamala, Andrew H Sung, and Ajith Abraham. 2005. Intrusion detection using an ensemble of intelligent paradigms. *Journal of network and computer applications* 28, 2 (2005), 167–182.

[29] Robi Polikar. 2006. Ensemble based systems in decision making. *IEEE Circuits and systems magazine* 6, 3 (2006), 21–45.

[30] Yan Qiao, XW Xin, Yang Bin, and S Ge. 2002. Anomaly intrusion detection method based on HMM. *Electronics letters* 38, 13 (2002), 663–664.

[31] RR Rejimol Robinson and Ciza Thomas. 2015. Ranking of machine learning algorithms based on the performance in classifying DDoS attacks. In *Intelligent Computational Systems (RAICS), 2015 IEEE Recent Advances in*. IEEE, 185–190.

[32] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, and Ali A Ghorbani. 2012. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *computers & security* 31, 3 (2012), 357–374.

[33] Robin Sommer and Vern Paxson. 2010. Outside the closed world: On using machine learning for network intrusion detection. In *Security and Privacy (SP), 2010 IEEE Symposium on*. IEEE, 305–316.

[34] 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.

[35] Chih-Fong Tsai, Yu-Feng Hsu, Chia-Ying Lin, and Wei-Yang Lin. 2009. Intrusion detection by machine learning: A review. *Expert Systems with Applications* 36, 10 (2009), 11994–12000.

[36] Chih-Fong Tsai and Chia-Ying Lin. 2010. A triangle area based nearest neighbors approach to intrusion detection. *Pattern recognition* 43, 1 (2010), 222–229.

[37] Gholam Reza Zargar and Peyman Kabiri. 2009. Identification of effective network features for probing attack detection. In *Networked Digital Technologies, 2009. NDT'09. First International Conference on*. IEEE, 392–397.

[38] Saman Taghavi Zargar, James Joshi, and David Tipper. 2013. A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks. *IEEE communications surveys & tutorials* 15, 4 (2013), 2046–2069.

[39] Jiong Zhang, Mohammad Zulkernine, and Anwar Haque. 2008. Random-forests-based network intrusion detection systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38, 5 (2008), 649–659.