# 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 Intrusion Detection System(IDS) is to maximize detection accuracy as well as minimize corresponding costs. In this paper, I present a cost-based approach utilizing neural network with GPU acceleration for fast intrusion detection. In this approach, both computational cost and time cost are considered, and models are trained with respect to their different protocol types 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 , , ,* 6 pages.
DOI: http://dx.doi.org/xxx

## 1 INTRODUCTION

With the development of computing and Internet technology, Internet has became an important part of our daily life. However, this popularity has also brought security issues. Intrusion Detection Systems(IDSs) are designed to detect attacks, which help discover, determine, and identify unauthorized use, duplication, alteration, and destruction of information systems[25]. Specifically, IDSs are able to monitor intrusions and alert network administrators if necessary.

According to the detectable attacks, there are mainly three types of IDSs: misuse-based, anomaly-based, and hybrid[4]. Misuse-based IDSs, which are also called signature-based IDSs are designed to detect known attacks by comparing new packets and the signatures in the database. Anomaly-based IDSs aims at modeling the normal network, and anomalies can be detected when happened. An attractive property of the anomaly-based IDSs is that they have the ability for detecting zero-day attacks.

Also with respect for using environment, IDSs can be used in real-time environment or off-line environment. Off-line IDSs have been studied extensively recently especially with the support of data mining and machine learning techniques, such as K-means[21] and ANN[13] etc. And excellent detection results with high accuracies and low false alarm rates are demonstrated. For detailed introduction for these machine learning techniques, the survey[4] is recommended.

However, these systems fail to provide real time information, which means they can only detect intrusions after they already happened for a while. However, it's obvious that as the complexity of features increase, the detection accuracy will also make an improvement. So a balance need to be maintained between accuracy and efficiency.

In this paper, the aim is to develop an IDS which can be utilized in real-time environment with high detection accuracy and low cost. I present a cost-based approach for anomaly intrusion detection. The traditional machine learning models are replaced by neural networks, which enable the computational cost to be minimized utilizing GPU acceleration. Also, the time cost is considered, and according to the time cost theory in [18], low level features are utilized with high priority. The benchmark dataset NSL-KDD is used for evaluation, which is a new version of dataset KDD Cup 1999[9] and has significant improvements compared with KDD Cup 1999[30].

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 introduced. 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 Detection Methodology

Basically, there are three types of intrusion detection techniques: misuse-based, anomaly-based, and hybrid[4]. Hybrid techniques are a combination for the two former techniques.

*2.1.1 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[19]. So in the misuse-based IDS, the process for coming packets is to compare them with the known signatures for recognizing possible intrusions.

The advantages of misuse-based method is that the misuse-based method is effective for detecting known type of attacks free from false alarms. However, the disadvantages are that it's usually hard to keep the signature database up to date. Also, the technique is incapable in detecting zero-day attacks or any attacks whose signatures are not included in the database.

*2.1.2 Anomaly-based Intrusion Detection.* An anomaly is a deviation to a known behavior, and profiles represent the normal or expected behaviors derived from monitoring regular activities,



,
© 2017 ACM. ACM ISBN xxx...$15.00
DOI: http://dx.doi.org/xxx

network connections, hosts or users over a period of time[19]. The anomaly-based techniques aims at modeling normal network status, and new packets are compared with the normal normal status to distinguish an attack.

The advantages of anomaly-based method is that they have the ability to recognize zero-day attacks. Another advantage is that unlike the misuse-based method, the profiles of normal activity are customized for every system, application, or network, thereby making it difficult for attackers to know which activities they can carry out undetected[4]. The main disadvantage of anomaly-based techniques is the it may present high false alarm rates, since every behavior that has not happened before has the potential risk to be considered as a intrusion.

*2.1.3 Hybrid Intrusion Detection.* Hybrid techniques combine the former two types of methods. They are implemented aiming at decrease the false alarm rate as well as detecting zero-day attacks.

## 2.2 Feature Selection

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

This work aims at reduce the features used for misuse-based techniques[5], since the comparison between the whole packet and all the signatures can be redundant. And useful feature selection algorithms are proposed. Also, the features on one specific dataset KDD Cup 1999 are analyzed by comparing the relevance between each features and information gain is employed to find the most discriminating features for each class[14].

Unsupervised machine learning techniques such as PCA[24][17] and K-means[16] are employed for feature selection. And all of these methods demonstrates good performance in feature selection.

However, these feature selection method ignore the time cost brought by the extraction process, so the selected features may bring severe time delay. In the cost-based approach presented in this paper, the cost factors are considerer, and a balance between cost and accuracy is maintained.

## 2.3 Machine Learning Approaches

In this section present a brief introduce of the utilization of machine learning techniques in intrusion detection. Specific machine learning algorithm is beyond the scope of this paper, and this book[1] is highly recommended for more information.

*2.3.1 Supervised Learning.* Supervised learning can be divided into generative models and deterministic models, and both 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 former techniques in supervised learning are used in intrusion detection[13][29][28]. Both the computational complexity and concrete introduction of the former techniques can be found in this work[4].

*2.3.2 Unsupervised Learning.* One of the most frequently used techniques is **clustering**. It is an unsupervised learning aims at clustering the data with some same property together. The main advantage of clustering is that it makes the labeling work not necessary. Specifically, for intrusion detection, the system can learn from audit data without requiring the network administrator to provide the attacks information.

There are many clustering method that demonstrates an excellent performance in detection intrusion experiments[12][21][16][31]. K-means, K-NN and etc are popular unsupervised machine learning approaches. Also, some work combine them together and better results are demonstrated[21]. However, as mentioned in the mentioned work, feature selection and reduction are urgently needed in methods like K-NN, because computing the similarities between high dimensions of features are expensive.

*2.3.3 Ensemble Learning.* Actually, the ensemble learning approaches do not belong to supervised or unsupervised learning. Usually, the idea of ensemble learning is to use multiple weak models to build a strong model, since some classification tasks may be difficult for one classifier. And complex models with lots of parameters always tend to over fit on the training data. Bagging and boosting are frequently used approaches in ensemble learning, [10] gives a good example for utilizing ensemble learning methods in decision tree models.

**Bagging**[2] predictors is a method for generating multiple versions of a single model and using these to get an aggregated predictor. The aggregation averages over all the versions of models when predicting a result by majority voting. **Boosting**[11] basically concentrates more the mis-classified data in the modeling process. Adaptive Boosting (AdaBoost)[27] is one of the most popular algorithms in boosting. The **Random Forest**[3] another method that combines the decision trees and ensemble learning. The so-called forest is composed of many decision trees.

Ensemble learning is extensively used in intrusion detection[32][26]. The single model can be nearly all kinds of previous mentioned machine learning models.

## 2.4 Public Datasets

*2.4.1 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[23][22]. 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.

*2.4.2 KDD Cup 1999 Dataset.* KDD Cup 1999 dataset[30] is 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, as well as some more advanced features like some content features within a connection and traffic features computed using a two-second time window. Totally, 41 features are extracted for a single connection.

*2.4.3 NSL-KDD Dataset.* As suggested in [30], the original KDD Cup 1999 dataset has two important issues which highly affects the performance of evaluated systems, and results in a very poor evaluation of anomaly detection approaches. The NSL-KDD

dataset consists of selected records of the complete KDD data set and free from the mentioned shortcomings in [30].

*2.4.4 ADFA Intrusion Detection Datasets.* The ADFA Intrusion Detection Datasets cover both Linux and Windows[7]. Different from former introduced datasets, ADFA contains raw system call traces. A new host-based anomaly intrusion detection methodology is evaluated on ADFD[8], which utilizing discontiguous system call patterns to increase detection rates whilst reducing false alarm rates. Also, excellent results are demonstrated.

# 3 METHODOLOGY

As shown in Fig 1, a multi-model detector is presented here. Based on a decision tree structure, different models are used for every kind of packets with different protocol types and services. And 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 in real-time IDSs is the computational cost attached with the system. For example, the well-known K-nearest Neighbor Classifier[20] 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 each 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, causing the system to be overloaded.

Traditionally, less features numbers and models with less parameters will lead to low computational cost. However, this may also result in lower detection accuracy or higher false alarm rate. Different from former approaches, utilizing deep neural networks, the computing tasks can be accelerated on GPU[6]. Also, the ability to represent any non-linear function in very few hidden layers and hidden neurons make 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[18], the connection based features are utilized according to[18] in this paper.

- 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.

Notice that the Level 3 features require iteration from the entire former two-second packet set, the time cost is extremely high compares 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 $\theta$, higher level features are extracted and are used combined with other lower level features;
- If the extracted feature level is already 3, then the eventual detection result is made without confidence calculating.

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, and for each new packet, its protocol type and service decides which detector to use. This model selection framework can be modeled by a decision tree.

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

## 3.3 Neural Network Detectors

Neural network models are at the bottom layer for the detection process, they 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 CPU acceleration support for deep learning libraries, which will reduce the computational cost in 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 with such simple architectures.

# 4 EXPERIMENTS

## 4.1 Dataset

As mentioned in Sec 2.4, NSL-KDD dataset consists of the selected records of the KDD Cup 1999 dataset, which is more challenging to detect 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[30]. 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 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 networks 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 0.75 to avoid over fitting. Finally, a softmax layer is utilized to calculate loss and Adam[15] is employed to accelerate learning. The prediction results of the models are the values of the two output neurons.

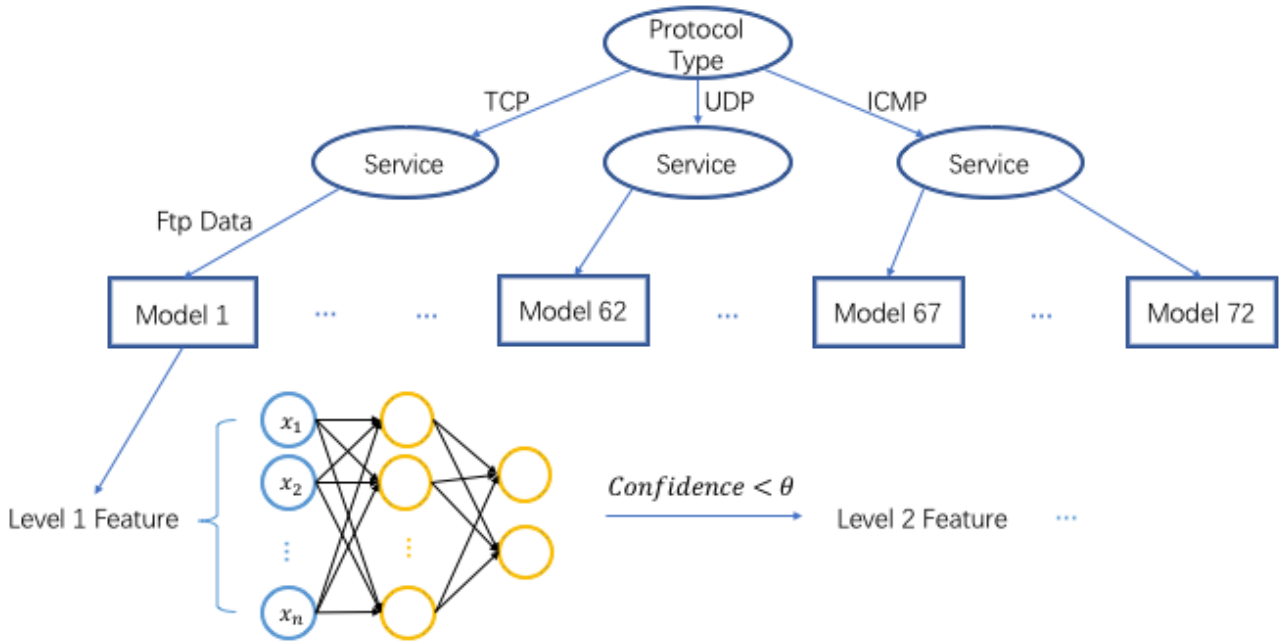


**Figure 1: An overview of the detection framework including the decision tree(top part) and the neural networks(bottom part). The total number of models are calculated on NSL-KDD dataset.**

## 4.3 Baseline

Support Vector Machine(SVM) is employed as a baseline method for each model respectively. The code are implemented in Python on Scikit-learn Library. And the following baseline results are evaluated on the same server with the proposed approach without GPU support for SVM method.

## 4.4 Detection Results

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

*4.4.1 Detection Accuracy.* Tab 1 demonstrates the detection accuracy for each level of features. And apart from the over all accuracy, I take three representative protocol type and service(TCP finger/UDP domain_u/ICMP eco_i) as examples.

Observed from Tab 1, 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 get higher, 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 level 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 decided finally by the higher feature level models. Also, lots of time cost will be wasted for no improvement in detection accuracy.

Also, we can observe that the decision tree idea that to detect different protocol types and service with different models is quite reasonable. In Tab 1, huge accuracy improvement has been made thanks to this separation.

*4.4.2 False Alarm Rates.* Tab 2 demonstrates the false alarm rates for each level of features. As mentioned before, three examples are selected.

*4.4.3 Cost Analysis.* Tab 3 demonstrates the **computational cost** for each level of features. The same as former experiments, three representative protocol type and service are selected. Notice here I report the average time for computing single one packet feature item.

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

Also, **time cost** is considered. Every new packet are first detected according to their 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.

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, and there is almost no decay of detection accuracy.

**Table 1: 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 | 77.26% | 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 2: Detection False Alarm Rates.**

| Methods/Features | Over all | TCP finger | UDP domain_u | ICMP eco_i |
|---|---|---|---|---|
| **Level 1 Features** | | | | |
| SVM | 78.20% | 0.00% | 0.67% | 57.69% |
| Ours | 77.26% | 99.26% | 99.33% | 94.27% |
| **Level 2 Features** | | | | |
| SVM | 81.39% | 0.00% | 0.45% | 57.69% |
| Ours | 80.67% | 99.26% | 99.78% | 94.27% |
| **Level 3 Features** | | | | |
| SVM | 2.19% | 0.00% | 0.00% | 19.23% |
| Ours | 76.24% | 99.26% | 99.78% | 98.89% |

**Table 3: 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 | $1.0 * 10^{-7}s$ | $2.8 * 10^{-6}s$ | $5.2 * 10^{-7}s$ | $1.5 * 10^{-6}s$ |

# 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 some 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.

## 5.2 More Issues

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

- There are some protocol type and service with very few training data, such as TCP pm_dump, UDP tftp_u and ICMP urh_i etc. This issue will make it impossible to train a neural network model. Maybe some signature-based technique are needed here.
- 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.
- 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 packages in one connection have arrived. Actually, as suggested in[18], some intrusion can be detected only after the first package

or a few packages are sent, which is capable to reduce the time cost even more. So extracting more package based features for model training can be considered as another improvement.

## 6 CONCLUSION

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

However, there still remains many issues to be solved for IDSs, such as how to detect intrusion only after a few packages have been received, and how to accurately detect zero-day attacks. Further research will investigate to solve the mentioned issues.

## REFERENCES


[1] Christopher M Bishop. 2006. Pattern recognition. *Machine Learning* 128 (2006).
[2] Leo Breiman. 1996. Bagging predictors. *Machine learning* 24, 2 (1996), 123–140.
[3] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.
[4] 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.
[5] 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.
[6] 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).
[7] 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.
[8] 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.
[9] KDD Cup. 2007. Available on: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. (2007).
[10] 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.
[11] 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.
[12] 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.
[13] 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.
[14] 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*.
[15] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[16] 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).
[17] Shilpa Lakhina, Sini Joseph, and Bhupendra Verma. 2010. Feature reduction using principal component analysis for effective anomaly–based intrusion detection on NSL-KDD. (2010).
[18] 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.
[19] 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.
[20] Yihua Liao and V Rao Vemuri. 2002. Use of k-nearest neighbor classifier for intrusion detection. *Computers & security* 21, 5 (2002), 439–448.
[21] 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.
[22] 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.
[23] 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.
[24] 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.
[25] 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.
[26] 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.
[27] Robi Polikar. 2006. Ensemble based systems in decision making. *IEEE Circuits and systems magazine* 6, 3 (2006), 21–45.
[28] Yan Qiao, XW Xin, Yang Bin, and S Ge. 2002. Anomaly intrusion detection method based on HMM. *Electronics letters* 38, 13 (2002), 663–664.
[29] 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.
[30] 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.
[31] 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.
[32] 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.