# Active Learning for Intrusion Detection Systems

Quang-Vinh Dang

# Active Learning for Intrusion Detection Systems

Quang-Vinh Dang
*Data Innovation Lab*
*Industrial University of Ho Chi Minh City*
Ho Chi Minh, Vietnam
dangquangvinh@iuh.edu.vn

***Abstract*—Intrusion Detection Systems (IDSs) play a vital role in the modern cyber-security system. The main task of an IDS is to distinguish between benign and malicious network flows. Hence, the researchers and practitioners usually utilize the power of machine learning techniques by considering an IDS as a binary-classifier. Recent research works demonstrate that an ensemble learning algorithm like xgboost can achieve almost perfect classification in the offline configuration. On the other hand, the performance of a simple and lightweight classification algorithm like Naive Bayes can be improved significantly if we can select a proper sub-training set. In this paper, we discuss the usage of active learning in online configuration to reduce the labeling cost but maintaining the classification performance. We evaluate our approach using the popular real-world datasets and showed that our approach outperformed state-of-the-art results.**

***Index Terms*—cyber-security, intrusion detection systems, active learning**

## I. Introduction

Intrusion Detection Systems (IDSs) are important components of modern information technology systems [1]. In short, the task of an IDS is to detect and classify any malicious activities that happen in a computer system that allows the quick and efficient reactions from the administrators, either manual or automatic.

By its nature, the core of an IDS is usually a machine learning classifier. The task of the machine learning algorithm is, given the information of a network flow, classify the network flow in one of two classes: *benign* or *malicious*. It is a well-known binary classification problem that has been studied for a long time in machine learning community. Over years, the researchers and practitioners have evaluated different algorithms and techniques to improve the classification performance of IDSs [2]. In recent years, along with the emergence of IoT devices, the requirement for IDSs have been extended to these environments [3], [4].

Another important research work is to create sharing datasets to study and evaluate the IDSs. One of very first efforts is the creation of the dataset KDD'99 [5]. After the release of KDD'99, the dataset has received several critical reviews [6]–[8] such as the unrealistic of the distribution of the network flow. Overtime, the researchers have released other datasets for IDSs evaluation, such as IDS'12 [9] and IDS'17 [10]. Both datasets are collected from real-world networks deployed by the Canadian Institute for Cybersecurity.

Recent research study [11] showed a very interesting result. For the traditional classification task of an IDS, i.e. to classify benign and malicious network flows, ensemble learning algorithms [12] such as xgboost [13] performed almost perfectly and achieved the AUC score around $99\%$. However, the main concern is that ensemble learning like boosting machines require a huge computational power, hence it is not practically to deploy them in IoT devices [3]. The authors of [11] demonstrated that even in general, a lightweight and simple algorithm like Naive Bayes cannot compete with ensemble machine learning algorithms, we still can improve the classification performance by choosing a proper training dataset while keep avoiding over-fitting. However, the method to choose the sub-training set is not discussed comprehensively in the study.

In this paper, we present our approach by using active learning technique to actively select training sample. Different from existing active learning algorithms, we rely on the idea that *rare* events are more important for learning, particularly for the Naive Bayes algorithm. We review state-of-the-art studies in Section II then present our approach in Section III. We evaluate our idea and show the experimental results in Section IV and conclude our paper in Section V.

## II. Literature Review

### A. Intrusion Detection Systems

The researchers and practitioners have deployed many different algorithms and variants as the core of IDSs. In general, they can be grouped into supervised and unsupervised settings [11].

Before the era of deep learning, the researchers [14]–[17] have proposed to use traditional machine learning methods such as decision-tree or logistic regression for the problem of intrusion detection. Feature generation by genetic programming has been considered by [17]. However, the following studies do not show a significant improvement of automatic feature generation compared to the manually building ones [11]. Other popular classification algorithms such as SVM have been used until recently [18], [19]. However, given the development of the computational systems and machine learning algorithms, ensemble learning algorithms like random forest are preferred [20].

As deep learning attracts a lot of attention since its victory in ImageNet competition in 2012 [21], the power of deep neural networks have been utilized in IDS. The authors of [22], [23] employed multi-layer neural networks and natural language processing techniques to analyze the behavior of computer systems. More complicated neural architectures rather than a

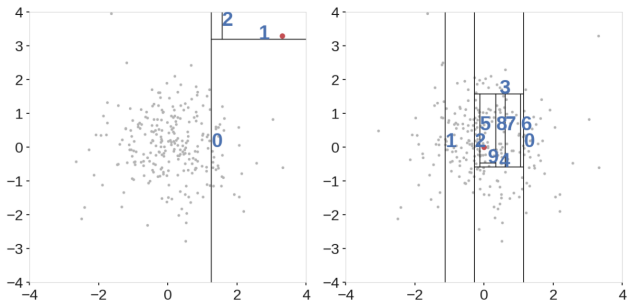


Fig. 1. Isolation Forest tries to isolate an instance from others [29]. Harder an instance to be isolated, more *normal* it is.

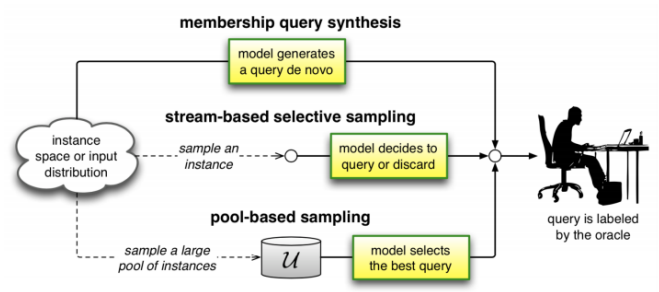


Fig. 2. Active Learning Schema [32].

feed-forward network is used to analyze the time-dependence of the network flows [24].

In the intrusion detection problem, the unsupervised machine learning algorithms are mostly the anomaly-detection algorithms [25], [26]. Anomaly-detector tries to distinguish between normal, or benign flows, with un-normal flows without explicitly defining what *normal* is. One of beginning efforts in using anomaly detection in IDSs belong to the authors of [27] where the researchers used One-class SVM algorithm for the discussing problem. Recent surveys such as [28] surveyed multiple efforts of using anomaly-detection in IDSs.

One of most powerful anomaly-detection algorithms up to date is Isolation Forest [29], [30]. The Isolation Forest is visualized in Figure 1. The main idea of Isolation Forest is that it tries to distinguish a single instance, and more difficult to distinguish an instance, more *normal* it is. Isolation Forest uses decision tree as its base learner.

*1) xgboost:* As xgboost is the state-of-the-art model in IDS research [11], we quickly review the algorithm here. In our model, we use xgboost to predict the performance impact of the algorithm.

*xgboost* stands for eXtreme Gradient Boosting [13] is a ensemble technique that has been introduced as one member of gradient boosting family [31]. The main idea of gradient boosting techniques is to build multiple sequential learners (will be referred as *weak learners*) where the next learner tries to correct the error made by the previous one.

Hence, the final boosting model of xgboost will be:

$$F_m(x) = F_0(x) + \sum_{i=1}^{M} \rho_i h_i(x, a_i) \tag{1}$$

$F_0(x)$ is the initial model: $F_0(x) = argmin_\rho \sum_{i=1}^{N} l(y_i, \rho)$, i.e. $F_0(x)$ can be initialized as constant. $\rho_i$ is the weight value of the model number $i$, $h_i$ is the base model (decision tree) at the $i^{th}$ iteration.

xgboost also introduced a new regularization term. The objective function is usually defined as:

$$obj(\theta) = l(\theta) + \Omega(\theta) \tag{2}$$

with $l$ is the loss function (e.g. squared-error $\sum (y_i - \hat{y}_i)^2$) and $\Omega$ is regularization term.

In the original version of gbm, no regularization term is used in the objective function. In contrast, xgb explicitly added regularization term, and an author of xgb in fact called their model as "regularized gradient boosting".

On the other hand, gbm introduced *shrinkage* as regularization techniques that are being used also by xgb. To use shrinkage, while updating the model in each iteration:

$$F_m(X) = F_{m-1}(X) + \nu.\rho_m h(x; a_m), 0 < \nu \leq 1 \tag{3}$$

The idea is, instead of taking the *full* step toward the steepest-descent direction, we take only a *part* of the step determining by the value of $\nu$.

### B. Active Learning

Active Learning is a sub-field of machine learning wherein the algorithm actively select the next instance for labelling [32]. Hence, active learning can reduce the requirement of training size while maintaining the performance of the algorithms by choosing only *predictive* instance from the pool for labelling. Active learning is visualized in Figure 2 [32]. Active learning assumes an *oracle* that can precisely label all unknown instances. In our setting, the oracle can be considered as a heavy algorithm like xgboost.

Active learning can employ different strategies to select the next sample [32]–[34].

Uncertainty-based sampling is probably the first active learning technique [35]. In this strategy the model pick the *least confidence* sample, defined as:

$$X^*_{LC} = argmax_X(1 - P_\theta(\hat{y}|X)) \tag{4}$$

wherein:

$$\hat{y} = argmax_y P_\theta(y|X) \tag{5}$$

Other confidence measures are proposed such as entropy or margin [34].

The other strategy is called *Query by Committee* [36]. The core idea of this method is to build a set of models, then let all the models to predict in all the unknown instances. The instance that the models disagree most will be chosen to

send to the oracle. The authors of [37] proposed two methods: *Bagging* and *Boosting* to construct the committee. The idea of bagging and boosting committee is borrowed from the same terminologies in machine learning. Furthermore, there are multiple ways to measure the *disagreement* of voters. Two popular metrics are K-L divergence and vote entropy.

The third strategy is called *Expected Model Change*. The idea of this strategy is to select the instance which might change the performance of the model most. One example of the strategy is *Expected Gradient Length* [38]:

$$X^*_{EGL} = argmax_X \sum_i P_\theta(y_i|X)|| \bigtriangledown l_\theta(L \cup (X_i, y_i))|| \quad (6)$$

The idea is to check what instance if added will change the gradient of the learning model the most. Other examples include Variance Reduction Method [39].

One very important idea in existing active learning methods is "Informative instances should not only be those which are uncertain, but also those which are 'representative' of the underlying distribution" [32]. Hence, the density-based method is proposed:

$$X^*_{ID} = argmax_X(\phi_A Xx(\frac{1}{U}\sum_{u=1}^{U} sim(x, x^{(u)}))^\beta) \quad (7)$$

The idea of the Equation 7 is avoid selecting some *outlier* instance by using the function *similarity* (sim).

## III. Our Approach

We rely on different idea than the density-based active learning [32]. We recall the Bayesian formula:

$$P(A|B) = \frac{P(B|A)}{P(B)}P(A) \quad (8)$$

Naive Bayes assume the independence between features which means:

$$P(f_1, f_2, ..f_n|BENIGN) = \prod_1^n P(f_i|BENIGN) \quad (9)$$

then assign the prediction as the highest probability class. From the Equation 9 we can claim that Naive Bayes has a computational speed advantage compared to other algorithms as it can be computed very fast.

It is well-known in practice that Naive Bayes algorithm does not work well with *too big* dataset [40]. The question is, what should be the proper dataset for Naive Bayes algorithm?

As we can see from the Bayesian formula, we might expect the higher change of $P(A|B)$ if we observe a *rare* event, i.e. if $P(B)$ is small. This observation contradicts the density-based idea. Given that we are aiming to improve the perform ace of Naive Bayes, our idea fits perfectly to the scheme. Hence, instead of choosing a next instance that is not too far from the distribution as stated in other studies [32], [34], [41], we actively choose the instance that is not so common in the observed data.

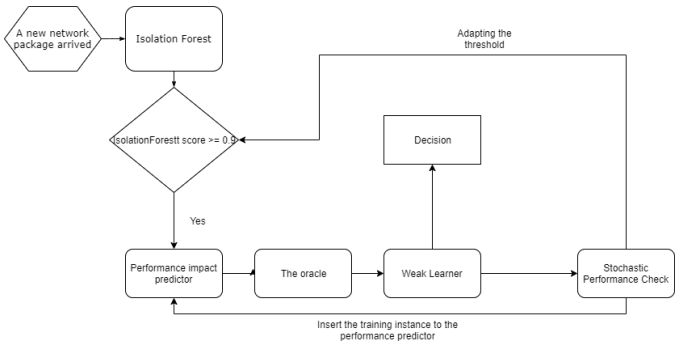


Fig. 3. Active Learning for IDSs

We use Isolation Forest to measure the difference of an instance to the observed distribution. Furthermore, we trained a xgboost model to predict the *performance impact* in term of change in AUC score if we add a particular training instance into the training dataset.

We visualize our proposal in Figure 3. The threshold for Isolation Forest score is adapted by the probability of picking up a single training instance over time.

## IV. Experimental Results

### A. Dataset

We evaluate our approach using CICIDS'12 dataset [9]. The dataset is collected in total seven days in 2010 with different types of attacks [42] from a real-world computer system. We describe the details number of each type of network flows in following:

- Monday
  - 529918 BENIGN flows.
- Tuesday
  - 432074 BENIGN flows.
  - 7938 FTP-Patator.
  - 5897 SSH-Patator.
- Wednesday
  - 440031 BENIGN.
  - 231073 DoS Hulk.
  - 10293 DoS GoldenEye.
  - 5796 DoS slowloris.
  - 5499 DoS Slowhttptest.
  - 11 Heartbleed.
- Thursday morning
  - 168186 BENIGN.
  - 1507 Web Attack Brute Force.
  - 652 Web Attack XSS.
  - 21 Web Attack SQL Injection.
- Thursday afternoon
  - 288566 BENIGN.
  - 36 Infiltration.
- Friday morning

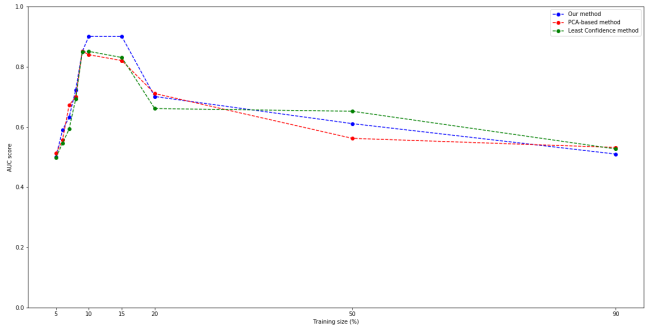


Fig. 4. Performance of different active learning strategies

  - 189067 BENIGN.
  - 1966 Bot.
- Friday afternoon 1
  - 97718 BENIGN.
  - 128027 DDos.
- Friday afternoon 2
  - 127537 BENIGN.
  - 158930 PortScan.

Each network flow is provided with a list of 78 features. We might notice that the distribution of the classes is extremely imbalanced: while the majority of the network flows are BENIGN, the DDos flows occupied a huge share of the network but there is very few Heartbleed flows for instance.

We follow the settings of [11] for a fair comparison.

### B. Experimental Results

We evaluate Least Confidence strategy against our strategy. Both strategies started with $5\%$ of the training set as the initial set. When the network flow arrived the algorithm will decide to include the particular network flow into the training set or not. In both cases, a test-set is hold out for evaluation.

In general, our method can achieve the AUC score of $90\%$ as the highest score, compared to the score of $85\%$ achieved by the method based on Lease Confidence or PCA [11].

We display the results in Figure 4. We could see that all the active learning strategies peak their performance when we select around $10\%$ of the training data, clearly suggested that there is a proper sub-training size which accounts for around one tenth of the training samples that actually bring the predictive power to the machine learning algorithms. Furthermore, the performance of Naive Bayes drops quickly if we increase the number of training size (by adaptive threshold method we described above). Contradict with the common sense in the machine learning community that we always need more data, the results showed that we rather need *good* data.

## V. Conclusions

In this paper we present our proposal to use outlier detection algorithm as the active learning base learner to improve the performance of lightweight intrusion detection methods. The algorithm does not require huge computational power, hence it is suitable for low-power devices like IoT devices or smartphones. In the future we will study other methods to improve the performance of lightweight methods to be comparable with ensemble methods.

()

## References


[1] A. Milenkoski, M. Vieira, S. Kounev, A. Avritzer, and B. D. Payne, "Evaluating computer intrusion detection systems: A survey of common practices," *ACM Comput. Surv.*, vol. 48, no. 1, pp. 12:1–12:41, 2015.

[2] Z. He, T. Zhang, and R. B. Lee, "Machine learning based ddos attack detection from source side in cloud," in *CSCloud*. IEEE Computer Society, 2017, pp. 114–120.

[3] K. Yang, J. Ren, Y. Zhu, and W. Zhang, "Active learning for wireless iot intrusion detection," *IEEE Wireless Commun.*, vol. 25, no. 6, pp. 19–25, 2018.

[4] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for iot security based on learning techniques," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 3, pp. 2671–2701, 2019.

[5] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *CISDA*. IEEE, 2009, pp. 1–6.

[6] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," *ACM Transactions on Information and System Security (TISSEC)*, vol. 3, no. 4, pp. 262–294, 2000.

[7] M. V. Mahoney and P. K. Chan, "An analysis of the 1999 darpa/lincoln laboratory evaluation data for network anomaly detection," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2003, pp. 220–237.

[8] A. Özgür and H. Erdem, "A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015," *PeerJ PrePrints*, vol. 4, p. e1954, 2016.

[9] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers & Security*, vol. 31, no. 3, pp. 357–374, 2012.

[10] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *ICISSP*. SciTePress, 2018, pp. 108–116.

[11] Q.-V. Dang, "Studying machine learning techniques for intrusion detection systems," in *Future Data and Security Engineering*, T. K. Dang, J. Küng, M. Takizawa, and S. H. Bui, Eds. Cham: Springer International Publishing, 2019, pp. 411–426.

[12] C. Zhang and Y. Ma, *Ensemble machine learning: methods and applications*. Springer, 2012.

[13] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *KDD*. ACM, 2016, pp. 785–794.

[14] X. Li and N. Ye, "Decision tree classifiers for computer intrusion detection," *Journal of Parallel and Distributed Computing Practices*, vol. 4, no. 2, pp. 179–190, 2001.

[15] C. Krügel and T. Toth, "Using decision trees to improve signature-based intrusion detection," in *RAID*, ser. Lecture Notes in Computer Science, vol. 2820. Springer, 2003, pp. 173–191.

[16] N. B. Amor, S. Benferhat, and Z. Elouedi, "Naive bayes vs decision trees in intrusion detection systems," in *SAC*. ACM, 2004, pp. 420–424.

[17] G. Stein, B. Chen, A. S. Wu, and K. A. Hua, "Decision tree classifier for network intrusion detection with ga-based feature selection," in *ACM Southeast Regional Conference (2)*. ACM, 2005, pp. 136–141.

[18] R. R. Reddy, Y. Ramadevi, and K. V. N. Sunitha, "Effective discriminant function for intrusion detection using SVM," in *ICACCI*. IEEE, 2016, pp. 1148–1153.

[19] D. Bhamare, T. Salman, M. Samaka, A. Erbad, and R. Jain, "Feasibility of supervised machine learning for cloud security," *CoRR*, vol. abs/1810.09878, 2018.

[20] P. A. A. Resende and A. C. Drummond, "A survey of random forest based methods for intrusion detection systems," *ACM Comput. Surv.*, vol. 51, no. 3, pp. 48:1–48:36, 2018.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1106–1114.

[22] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for internet of things," *Future Generation Comp. Syst.*, vol. 82, pp. 761–768, 2018.

[23] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41 525–41 550, 2019.

[24] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "HAST-IDS: learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2018.

[25] S. D. D. Antón, S. Sinha, and H. D. Schotten, "Anomaly-based intrusion detection in industrial data with SVM and random forests," in *SoftCOM*. IEEE, 2019, pp. 1–6.

[26] Z. Chiba, N. Abghour, K. Moussaid, A. E. Omri, and M. Rida, "Intelligent and improved self-adaptive anomaly based intrusion detection system for networks," *IJCNIS*, vol. 11, no. 2, 2019.

[27] E. Eskin, "Anomaly detection over noisy data using learned probability distributions," in *ICML*. Morgan Kaufmann, 2000, pp. 255–262.

[28] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 1, pp. 303–336, 2014.

[29] F. T. Liu, K. M. Ting, and Z. Zhou, "Isolation forest," in *ICDM*. IEEE Computer Society, 2008, pp. 413–422.

[30] Q. Dang, "Outlier detection on network flow analysis," *CoRR*, vol. abs/1808.02024, 2018.

[31] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[32] B. Settles, "Active learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2009.

[33] A. Ziai, "Active learning for network intrusion detection," *CoRR*, vol. abs/1904.01555, 2019.

[34] G. M. Alqaralleh, M. A. Alshraideh, and A. Alrodan, "A comparison study between different sampling strategies for intrusion detection system of active learning model," *JCS*, vol. 14, no. 8, pp. 1155–1173, 2018.

[35] D. D. Lewis and W. A. Gale, "A sequential algorithm for training text classifiers," in *SIGIR*. ACM/Springer, 1994, pp. 3–12.

[36] H. S. Seung, M. Opper, and H. Sompolinsky, "Query by committee," in *COLT*. ACM, 1992, pp. 287–294.

[37] N. Abe and H. Mamitsuka, "Query learning strategies using boosting and bagging," in *ICML*. Morgan Kaufmann, 1998, pp. 1–9.

[38] B. Settles, M. Craven, and S. Ray, "Multiple-instance active learning," in *NIPS*. Curran Associates, Inc., 2007, pp. 1289–1296.

[39] J. O'Neill, S. J. Delany, and B. M. Namee, "Model-free and model-based active learning for regression," in *UKCI*, ser. Advances in Intelligent Systems and Computing, vol. 513. Springer, 2016, pp. 375–386.

[40] A. Burkov, *Machine Learning Engineering*. LeanPub, 2019.

[41] R. K. Deka, D. K. Bhattacharyya, and J. K. Kalita, "Active learning to detect ddos attack using ranked features," *Computer Communications*, vol. 145, pp. 203–222, 2019.

[42] Q. Dang and J. François, "Utilizing attack enumerations to study SDN/NFV vulnerabilities," in *NetSoft*. IEEE, 2018, pp. 356–361.