



Autoperman: Automatic Network Traffic Anomaly Detection with Ensemble Learning

Shangbin Han¹, Qianhong Wu¹, Han Zhang^{2(✉)}, Bo Qin³, Jiangyuan Yao⁴,
and Willy Susilo⁵

¹ School of Cyber Science and Technology, Beihang University, Beijing, China

² INSC&BNRist, Tsinghua University, Beijing, China
zhhan@tsinghua.edu.cn

³ School of Information, Renmin University of China, Beijing, China

⁴ School of Computer Science and Technology, Hainan University, Haikou, China

⁵ School of Computing and Information Technology, University of Wollongong,
Wollongong, Australia

Abstract. Network traffic, which records users' behaviors, is valuable data resources for diagnosing the health of the network. Mining anomaly in network is essential for network defense. Although traditional machine learning approaches have good performance, their dependence on huge training data set with expensive labels make them impractical. Furthermore, after complex hyperparameters tuning, the detection model may not work. Facing these challenges, in this paper, we propose Autoperman through supervised learning. In Autoperman, machine learning algorithms with fixed hyperparameters as feature extractors are integrated, which utilize a small amount of training data to be initialized. Then Random Forest is selected as the anomaly classifier and achieves automatic parameters tuning via well studied online optimization theory. We compare the performance of Autoperman against traditional anomaly detection algorithms using public traffic datasets. The results demonstrate that Autoperman can perform about 6.9%, 34.2%, 4.3%, 2.2%, 37.6 % better than L-SVM, NL-SVM, LR, MLP, K-means, respectively.

Keywords: Anomaly detection · Autoperman · Ensemble learning

1 Introduction

The interconnected network has an unprecedented impact on economies and societies around the globe. By January 2021, the number of Internet users accounts for 59.5% of the global population [1]. The network presents users' convenience, which could not be achieved previously. However, it faces severe security issues, such as DDoS attack and malware.

Anomaly detection [2, 4, 6, 20] plays an important role in timely response to stop service failures. In the past, manual anomaly detection was a natural solution to this problem. The IT operators define a regular expression to match

anomalies. However, the rapid growth of traffic volume has imposed great challenges for manual monitoring methods. The manual anomaly detection cannot cope with large-scale traffic efficiently.

As a result, there is a huge demand in designing an automated anomaly detection framework. To this end, machine learning technologies (SVM [9], MLP [19]) is progressively being used in the anomaly detection area. However, we find that they have the following **limitations**: First, the definition of anomaly is limited. All anomalies are defined strictly at the data level. The anomalies that do not exist in the training data cannot be classified. Second, the traditional machine learning methods need huge training data with expensive labels to drive an optimal detection model. Third, it is hard to tune hyperparameters of a detection model in an offline manner. Even the trained model with optimal hyperparameters may be unable to work in the future due to the fast changing network behavior over time. Facing these challenges, in this paper, we make the following **contributions**:

First, we find that traditional machine learning anomaly detection algorithms will reduce their performance when training data is not enough and may not work over time.

Second, we propose an anomaly detection framework named Autoperman. It needs a small amount of training data to initialize model parameters. Autoperman uses popular detectors with fixed hyperparameters to extract features, and uses the dynamic labels defined by the application layer (feedback of IT operators) for further automatic update of classification model over time.

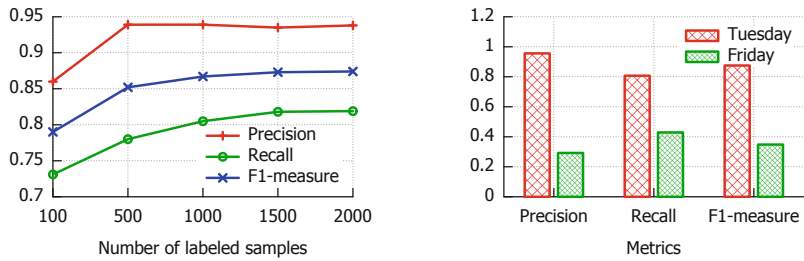
Third, we compare the performance of Autoperman against traditional anomaly detection algorithms using three public traffic datasets (i.e., CICIDS2017 [26]). The results show that Autoperman can produce competitive performance, which performs about 6.9%, 34.2%, 4.3%, 2.2%, 37.6 % better than L-SVM [13], NL-SVM [9], LR [23], MLP [19], K-means [15], respectively.

2 Background and Motivation

Traditional data-driven anomaly detection framework usually comprises three components: traffic collection, feature generation and anomaly detection. Traffic collection captures the traffic and sends it to the designated analysis equipment. Feature generation is the stage of feature extraction and labeling. Finally, the trained module predicts whether the current traffic is abnormal or not.

In reality, the data-driven anomaly detection needs huge training data to drive an optimal model. We take SVM and the public CICIDS2017 data set [26] as the example to illustrate this point. As shown in Fig. 1(a), the performance of the detection model is improved with the increase of training data. It means that more expensive labels are needed to derive an optimal model.

Besides, model training is time-consuming. Tuning hyperparameters of SVM is impractical for the data on Tuesday of CICIDS2017 with about half-a-million instances. In our experiment setting, $\frac{1}{9}$ of the data are randomly selected, and 12



(a) SVM needs more training data to de- (b) SVM may not work over time after
rive a satisfactory model. turning hyperparameters.

Fig. 1. Limitations of SVM.

groups of hyperparameter combinations of SVM are tuned through grid search and cross-validation. The time cost is as high as 3.5 h. Unfortunately, they may not work over time. We use the traffic on Tuesday and Friday of CICIDS2017 to show the problem. As shown in Fig. 1(b), the model is trained with data on Tuesday, and the corresponding detection shows good performance. However, when the well trained model is used for detecting traffic on Friday, the resulting performance is poor. All the metrics are lower than 0.5. The reason for this is that new attacks emerge over time, which affects the data pattern.

3 The Autoperman Framework

In this section, we present the Autoperman framework: the feature generation and the anomaly detection algorithm. The traffic collector is omitted here, since it is not the focus of this work.

3.1 Overview

Autoperman is an anomaly detection framework without tuning hyperparameters manually, designed for automatic and efficient abnormal traffic detection

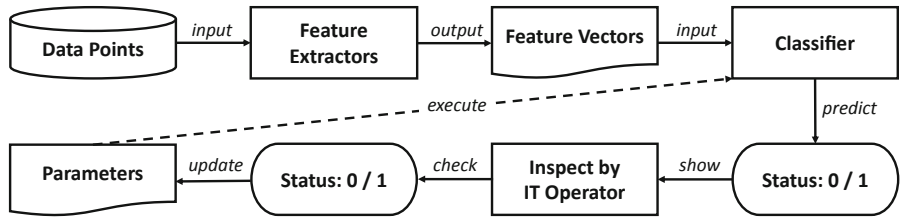


Fig. 2. Autoperman contains feature generation and automatic classifier update.

in large scale network. It proceeds by detecting anomalous traffic in an online fashion through supervised machine learning.

Instead of the heavy labeling workload, we leverage the feedback of IT operators to achieve the supervised learning. Figure 2 shows the process. First, network traffic is expressed in the form of value vector, which is a point on the hyperplane. We call them data points. Second, the existing machine learning algorithms (e.g., SVM) with different fixed hyperparameters are used as feature extractors to generate the feature vectors. Before the execution, all the feature extractors are initialized by small amount of training data. Third, random forest is adopted as the classifier to derive current status, whose function is to select the feature extractors. Fourth, IT operators feedback the labels for model update only when false alarm happens. That is, if the positive report shows an error, the real status is revealed. While for the negative report, IT operators selectively review and provide feedback when there is an error. Finally, the difference between the predicted status and real status are used to automatically update parameters of classifier. This online iterative parameter updating method continuously optimizes the classification model to achieve the optimal detection performance.

The advantages of Autoperman are that it can save most of the work of expert labeling, the time consuming of repetitive parameter tuning and realize automatic and efficient anomaly detection over time. Next, we give the design details for the feature generation and parameter update of classifier.

3.2 Feature Generation

Many learning based anomaly detectors have been used to detect anomalies. We model the detector as a mapping function: $\phi(\mathbf{x}) \rightarrow score \in [0, 1]$, where $\mathbf{x}$ is a data point, and the *score* represents the severity of anomaly. The *score* is a metric of the anomaly which can be seen as a feature.

The *score* is derived from the well-trained model. To avoid time-consuming hyperparameters tuning, we sample the hyperparameters of each detector to form different detectors. The *scores* predicted by all the detectors are concatenated to generate the feature vectors. Next, we introduce the selected anomaly detectors that implemented in scikit-learn [24] and corresponding hyperparameters combination.

1) **K-means.** It groups similar items with the number of clusters represented by the variable K . The algorithm works iteratively to assign each data point to one of the K clusters based on the features that are provided. We select the parameter $K = 2$, since anomaly detection is a binary classification problem.

2) **Logistic Regression.** It uses a *sigmoid function* to map any real values to probabilities. We consider linear logistic regression with different regular terms. We select the parameters from set $\{0.1, 0.3, 0.5, 0.7, 0.9, 1.1, 1.3, 1.5, 1.7, 1.9\}$ for l_1 -regularization and l_2 -regularization, respectively.

3) **Decision Tree.** It is a tree structure for classification. The key hyperparameters of decision tree we consider are the criterion that split data points and the maximum depth of the decision tree. For the criterion, we can choose ‘gini’

or ‘entropy’. Additionally, for the tree depth, we can choose from the set $\{3, 5, 7, 9, 11, 13, 15, 17, 19\}$ and do not set the maximum depth.

4) **SVM.** It aims to create the best decision boundary to classify n -dimensional feature space into groups. We employ two kinds of SVM: linear SVM and non-linear SVM. For the linear SVM, we need to sample the parameters of regularization term and corresponding coefficient. The regularization term has two options $\{l_1, l_2\}$, and the coefficient can be selected from $\{0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0\}$. For the non-linear SVM, we have two parameters to sample, i.e., kernel function and the degree of a polynomial kernel function. We consider three kernel functions, such as *poly*, *rbf* and *sigmoid*. If the kernel function is *poly*, we sample its degree from the set $\{1, 2, 3, 4, 5\}$, otherwise we ignore the parameter.

5) **MLP.** MLP is a feed-forward artificial neural network model that maps multiple inputs to a set of outputs. In this paper, we fixed two hyperparameters. For the number of hidden layers, we get the value from the set $\{0, 1, 2, 3\}$. The neurons number of each layer can be chosen from $\{32, 64, 128, 256, 512\}$.

Finally, considering all the hyperparameters combination of each detector, we can get total 88 features.

3.3 Classifier

In this paper, we select random forest as the classifier due to its randomness.

Random Forest. Random forest consists of a large number of individual decision trees. Each tree in the random forest will make a class probability prediction, and the average prediction taken over all individual trees will be the final prediction of the model [24].

The final result is obtained via equal weight distribution of each decision tree. However, the characteristics of network traffic will change over time. Hence the equal weight should be adjusted *automatically* according to the network environment. Next, we describe how to update weight by leveraging online optimization.

Automatic Update of Weight. As noted in Fig. 2, the prediction status is set before the real status is revealed (findings of IT operator). Therefore the prediction can be cast as an online convex optimization (OCO) problem [11].

The metric of *regret* commonly used in the machine learning literature is introduced to measure the performance of online algorithms. $Regret_T$ is the accumulated difference of losses between *weight* (vector $\mathbf{w}$) and an offline *optimal weight* (a vector $\mathbf{w}^*$) called the optimal static policy, which is taken when the entire data is known in hindsight. Specifically, the worst-case static regret constructed in a horizon T -iterations repeated game is defined as follow:

$$Regret_T = \max_{\mathcal{F}} \left\{ \sum_{t=1}^T f_t(\mathbf{w}_t) - \min_{\mathbf{w}^* \in \mathcal{S}} \sum_{t=1}^T f_t(\mathbf{w}^*) \right\} \quad (1)$$

Let the concatenated probability values predicted by decision trees in Random Forest be vector $\mathbf{v}_t$, the prediction of Random Forest be $\hat{y}_t$, the IT operator's feedback be y_t , and the loss function be f_t . We hope that when the prediction is accurate ($\hat{y}_t = y_t$), the loss can be small, while the wrong prediction has the opposite effect. Therefore, we would like to use the following convex loss function of $\mathbf{w}_t$ over any convex vector set $\mathcal{S}$:

$$f_t(\mathbf{w}_t) = -y_t \log p_t - (1 - y_t) \log(1 - p_t) \quad (2)$$

where p_t is a discrete anomaly probability distribution: $p_t = \frac{1}{1 + \exp(-\mathbf{w}_t \cdot \mathbf{v}_t)}$. If $p_t > 0.5$, then the prediction $\hat{y}_t = 1$, otherwise $\hat{y}_t = 0$. Therefore, when IT operator returns a feedback $\hat{y}_t = 1$, the loss will decrease with increasing p_t , and when the feedback $\hat{y}_t = 0$, the situation is reversed as expected. Next we can give the details of weight update.

Follow the regularized leader (FTRL) is an excellent algorithm for OCO framework. To derive the best *weight* $\mathbf{w}_t$ at current t , any vector that has minimal loss on all past rounds is preferred:

$$\mathbf{w}_{t+1} = \arg \min_{\mathbf{w}} \sum_{i=1}^t f_i(\mathbf{w}) + R(\mathbf{w}) \quad (3)$$

where $R(\mathbf{w})$ is a L_2 regularized function which can stabilize the update of weight $\mathbf{w}$. We set the regularization term $R(\mathbf{w})$ as: $R(\mathbf{w}) = \frac{1}{2\eta} \|\mathbf{w}\|_2^2$, where η is the learning rate. According to KKT condition [8], we can obtain the solution by setting the derivative of (3) to be 0. Therefore, we can obtain the update policy of best weight as follows: $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla f_t(\mathbf{w}_t)$, where $\nabla f_t(\mathbf{w}_t)$ is the gradient of f_t at $\mathbf{w}_t$: $\nabla f_t(\mathbf{w}_t) = (p_t - y_t) \mathbf{x}_t$.

Finally, Algorithm 1 shows the proposed Online FTRL for updating weight (OFUW) algorithm. Algorithm 1 takes initial weight $\mathbf{w}$ and learning rate η_t as input. After receiving a probability value vector $\mathbf{v}_t$ (Line 2), it uses the current weight $\mathbf{w}_t$ to predict anomaly (Line 3). Subsequently, the true status of network is observed by IT operators (Line 4). At last, $\mathbf{w}_t$ is updated with the loss function (Line 5–6).

Algorithm 1: Online FTRL for updating weight (OFUW)

Input: $\mathbf{w}_1 \in \mathcal{S}$, learning rate η

- 1 **for** $t \in T$ **do**
 - 2 Get vector $\mathbf{v}_t$ from Random Forest;
 - 3 Predict the status $\hat{y}_t$ with p_t ;
 - 4 IT operator returns the real status $y_t \in \{0, 1\}$ if $\hat{y}_t$ is wrong;
 - 5 $\nabla f_t(\mathbf{w}_t) = (p_t - y_t) \mathbf{v}_t$;
 - 6 $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla f_t(\mathbf{w}_t)$;
-

In theory [25], given an appropriate choice of $\eta = \psi(\frac{1}{\sqrt{t}})$, the FTRL algorithm can derive a theoretical guarantees of upper bound of regret, which is a sublinear

function of T , i.e., $o(\sqrt{T})$. This shows Autoperman with OFUW has no regret as T goes to infinity.

4 Evaluation

In this section, we evaluate the performance of Autoperman with public traffic data sets.

4.1 Data Sets

In our evaluation, three public network traffic data sets are used to examine the performance of Autoperman:

1. **Kitsune network attack data set.** It is generated on an IoT network [17], which contains Denial of Service (SSDP Flood, SYN DoS), Botnet Malware (Mirai), etc. Each attack data set is labeled as malicious or not.
2. **UNSW-NB15 data set.** It is a traffic data set mixed with normal real modern network behaviors and synthetic contemporary attacks [18]. The data set has totally 49 features with the class label (0 or 1).
3. **CICIDS2017 data set.** It is generated from a proposed B-Profile system [26]. The data set is captured from July 3, 2017 (Monday) to July 7, 2017 (Friday). The normal instances are labeled with “BENIGN” while the remaining with common attacks.

4.2 Experiment Setup

We conduct all our experiment on a Linux server, which is configured with 32G memory, 9T hard disk and 16 Intel Xeon E5-2630 CPUs. All the feature extractors in Autoperman are implemented from scikit-learn [24] and they are all initialized with default hyperparameters using small training samples, e.g., 500 samples. For classifier Random Forest, the number of trees is set to 100, the depth of each tree is considered no maximum. For algorithm OFUW, we set the default value of learning rate η to $\frac{1}{\sqrt{t}}$, where t is the iteration times. To simulate the feedback of IT operators, Autoperman uses the labels of public traffic data set only when false alarm happens.

4.3 Baselines

We compare the performance with five methods: one classical unsupervised learning methods K-means [15], three classical supervised learning methods linear SVM (L-SVM) [13], non-linear SVM (NL-SVM) [9] and Logistic regression (LR) [23], one neural network method Multi-layer Perceptron (MLP) [19].

We use the following metrics to evaluate the performance:

1. Precision: It calculates the proportion of predicted positive instance that are actually positive instance. It is expressed as: $\text{Precision} = \frac{TP}{TP+FP}$, where TP denotes True Positives and FP denotes False Positives.
2. Recall: It is the ability of a detection model to identify positive instance, which is defined as: $\text{Recall} = \frac{TP}{TP+FN}$, where FN denotes False Negatives.
3. F1-measure: It considers both Precision and Recall, and is defined as the harmonic average of the two metric: $\text{F1-measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$.

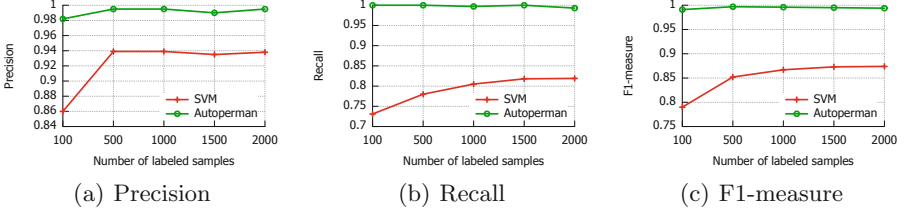


Fig. 3. Autoperman can maintain good performance with a small amount of labels.

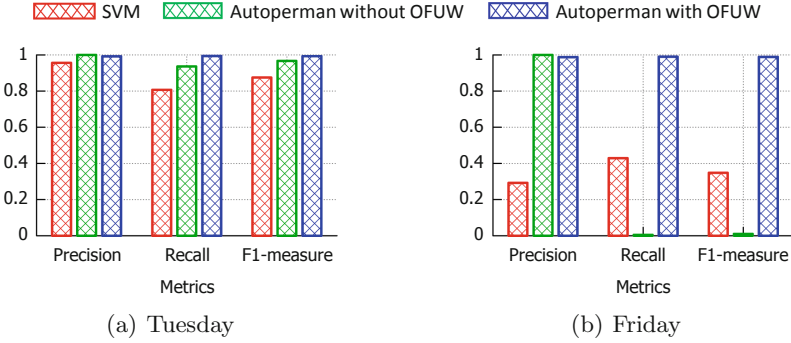


Fig. 4. Autoperman can automatically adapt to the network over time.

4.4 Revisiting the Motivating Example

In this section, we will revisiting the motivating example to verify the effectiveness of Autoperman. All the parameters are set with default value. We also plot the results of SVM to make comparisons.

First, we use the same traffic trace to investigate the performance impact of training samples number for Autoperman. The results are illustrated in Fig. 3. From this figure, we can see that the performance of Autoperman fluctuates very little w.r.t. different number of training samples. Additionally, SVM needs more data to perform better. When the amount of training data is 100, the precision of Autoperman can reach up to 0.98, while SVM is lower than 0.9. Therefore,

the detection performance of Autoperman is much better than that of SVM. This is because Autoperman can use feedback from IT operators to improve performance. Autoperman uses less labels to produce comparable performance, which has less labeling cost.

Second, we investigate the adaptability of Autoperman. We also paint the SVM and Autoperman without OFUW for comparison. Subsequently, Fig. 4 shows the results. We use the data on Tuesday to train detection models and use the trained models to detect anomalies on Tuesday and Friday. Specifically, Fig. 4(a) shows the performance comparison for SVM and Autoperman on Tuesday. We can see that both SVM and Autoperman can achieve satisfactory performance. However, as Fig. 4(b) shown, SVM has poor detection accuracy. Although the precision of Autoperman without OFUW is high, the Recall and F1-measure are almost zero. In contrast, Autoperman with OFUW can work very well in this case. The reason is that Autoperman can update parameters to adapt to changes in the network. This means that Autoperman can automatically adapt to the dynamic network environment without repetitive and time-consuming hyperparameters tuning.

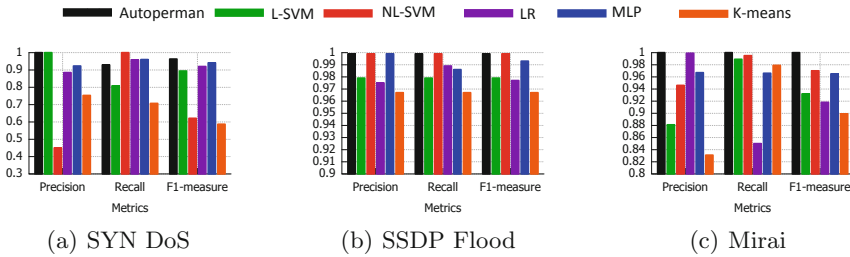


Fig. 5. Detection accuracy under Kitsune network attack data set.

4.5 Performance Comparison

In this part, we compare the performance of Autoperman with baseline methods on the three public traffic data sets. Figure 5 shows the results on Kitsune network attack data set. On the whole, Autoperman shows the best performance. Take SYN DoS (Fig. 5(a)) for example, in terms of F1-measure metric, Autoperman performs about 6.9%, 34.2%, 4.3%, 2.2%, 37.6% better than L-SVM, NL-SVM, LR, MLP, K-means, respectively. The reason for this is that Autoperman can use feedback of IT operator to automatically update to adapt to the dynamic characteristics of network data.

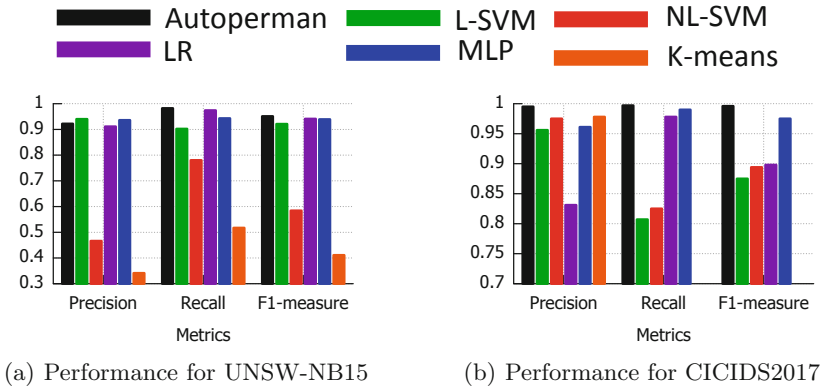


Fig. 6. Detection accuracy under UNSW-NB15 and CICIDS2017 data sets.

Figure 6(a) shows the performance for UNSW-NB15 data set. The results are basically the same as Fig. 5. We can see that Autoperman can improve the detection accuracy by up to 50%. Figure 6(b) shows the performance under CICIDS2017 data set. We can see that Autoperman is obviously better than other algorithms. Although K-means has high precision, the other two metrics are basically 0, which are not painted here. This means the detection model cannot find anomalies. Generally, Autoperman shows better performance, and it only needs a small amount of labels to get competitive accuracy.

5 Related Work

Anomaly detection is an important component in the field of network security. A much more convincing line of research has been proposed.

Supervised machine learning is widely used in the anomaly detection domain. Shubhra et al. [9] optimized the SVM algorithm to improve the detection accuracy. Kasim et al. [13] employed SVM as the classifier to identify anomalies from features processed by autoencoder. Sultana et al. [23] used logistic regression for anomaly detection in smart power grids. Mao et al. [16] applied the decision tree to selected feature vectors for anomaly detection.

Ahmed et al. [3] used the decision tree and rule-based models to construct different anomaly detectors. They have high detection accuracy. Unfortunately, these methods rely heavily on huge labels.

Besides supervised learning methods, many unsupervised learning methods have been proposed. Yang et al. [15] evaluated different clustering methods, K-means, Canopy, FarthestFirst for network traffic classification, and K-means shows the better performance in accuracy. Keval et al. [7] used K-means clustering to identify anomalies in traffic videos. Sahil et al. [10] optimized DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm to achieve clustering of abnormal Internet of things data. Ning et al. [12] proposed a multi-kernel clustering based anomaly detection methods to make up

the absent traffic characteristics. These approaches have the advantage of not using labels. However, the detection accuracy is reduced.

The recent popular deep learning approaches [22] were also applied to detect anomalies in network traffic. Teoh et al. [19, 29] proposed advanced deep learning techniques MLP to detect malicious traffic. Vinod et al. [14] evaluated different machine learning algorithms for anomaly detection, including MLP, decision tree, etc. [17, 21, 28] proposed autoencoder based anomaly detection methods. [5, 27] adopted the recurrent neural network LSTM as the anomaly detector. These methods have strong learning ability and perform very well. However, they require a massive amount of training data combined with significant computational resources.

6 Conclusion

In this paper, we proposed an Autoperman framework to solve the limitations of traditional network traffic anomaly detection based on machine learning algorithms. Autoperman framework contains feature generation and online anomaly detection. Autoperman consumes a small amount of training data to initialize the detection model and automatically tune parameters through online optimization theory, which has less labeling cost and does not need time-consuming parameter tuning. We conducted experiments to show that Autoperman has better performance than traditional methods on two representative public traffic datasets.

Acknowledgements. This paper is supported by the National Key R&D Program of China through project 2020YFB1005600, the Beijing Natural Science Foundation through project M21031, the Natural Science Foundation of China through projects U21A20467, 61932011 and 61972019.

References

1. Digital 2021: Global overview report. <https://datareportal.com/reports/digital-2021-global-overview-report>
2. Afzal, R., Murugesan, R.K.: Rule-based anomaly detection model with stateful correlation enhancing mobile network security. *Intell. Autom. Soft Comput.* **31**(3), 1825–1841 (2022)
3. Ahmim, A., Maglaras, L., Ferrag, M.A., Derdour, M., Janicke, H.: A novel hierarchical intrusion detection system based on decision tree and rules-based models. In: 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS), pp. 228–233. IEEE (2019)
4. Assiri, A.: Anomaly classification using genetic algorithm-based random forest model for network attack detection. *CMC-Comput. Mater. Continua* **66**(1), 767–778 (2021)
5. Chastikova, V., Sotnikov, V.: Method of analyzing computer traffic based on recurrent neural networks. *J. Phys. Conf. Ser.* **1353**, 012133 (2019)
6. Dong, Z., Han, J., et al.: Deep learning anomaly detection based on hierarchical status-connection features in networked control systems (2021)

7. Doshi, K., Yilmaz, Y.: Fast unsupervised anomaly detection in traffic videos. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pp. 624–625 (2020)
8. Dreves, A., Fachinei, F., Kanzow, C., Sagratella, S.: On the solution of the KKT conditions of generalized Nash equilibrium problems. *SIAM J. Optim.* **21**(3), 1082–1108 (2011)
9. Dwivedi, S., Vardhan, M., Tripathi, S.: Building an efficient intrusion detection system using grasshopper optimization algorithm for anomaly detection. *Clust. Comput.* **24**(3), 1881–1900 (2021). <https://doi.org/10.1007/s10586-020-03229-5>
10. Garg, S., Kaur, K., Batra, S., Kaddoum, G., Kumar, N., Boukerche, A.: A multi-stage anomaly detection scheme for augmenting the security in IoT-enabled applications. *Futur. Gener. Comput. Syst.* **104**, 105–118 (2020)
11. Hazan, E., et al.: Introduction to online convex optimization. *Found. Trends® Optim.* **2**(3–4), 157–325 (2016)
12. Hu, N., Tian, Z., Lu, H., Du, X., Guizani, M.: A multiple-kernel clustering based intrusion detection scheme for 5G and IoT networks. *Int. J. Mach. Learn. Cybernet.* **12**, 1–16 (2021)
13. Kasim, Ö.: An efficient and robust deep learning based network anomaly detection against distributed denial of service attacks. *Comput. Netw.* **180**, 107390 (2020)
14. Kumar, V., Choudhary, V., Sahrawat, V., Kumar, V.: Detecting intrusions and attacks in the network traffic using anomaly based techniques. In: 2020 5th International Conference on Communication and Electronics Systems (ICCES), pp. 554–560. IEEE (2020)
15. Liu, Y., Xue, H., Wei, G., Wu, L., Wang, Yu.: A comparative study on network traffic clustering. In: Liu, J.K., Huang, X. (eds.) NSS 2019. LNCS, vol. 11928, pp. 443–455. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36938-5_27
16. Mao, J., Hu, Y., Jiang, D., Wei, T., Shen, F.: CBFS: a clustering-based feature selection mechanism for network anomaly detection. *IEEE Access* **8**, 116216–116225 (2020)
17. Mirsky, Y., Doitshman, T., Elovici, Y., Shabtai, A.: Kitsune: an ensemble of autoencoders for online network intrusion detection. *arXiv preprint arXiv:1802.09089* (2018)
18. Moustafa, N., Slay, J.: UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: 2015 Military Communications and Information Systems Conference (MilCIS), pp. 1–6. IEEE (2015)
19. MR, G.R., Somu, N., Mathur, A.: A multilayer perceptron model for anomaly detection in water treatment plants. *Int. J. Crit. Infrastruct. Prot.* **31**, 100393 (2020)
20. Muneer, A., Taib, S.M., Fati, S.M., Balogun, A.O., Aziz, I.A.: A hybrid deep learning-based unsupervised anomaly detection in high dimensional data. *Comput. Mater. Contin* **71**, 6073–6088 (2021)
21. Neuschmied, H., Winter, M., Hofer-Schmitz, K., Stojanovic, B., Kleb, U.: Two stage anomaly detection for network intrusion detection. In: Proceedings of the ICISSP (2021)
22. Nkenyereye, L., Tama, B.A., Lim, S.: A stacking-based deep neural network approach for effective network anomaly detection. *CMC-Comput. Mater. Continua* **66**(2), 2217–2227 (2021)
23. Noreen, S.S., Bayne, S.B., Shaffer, E., Porschet, D., Berman, M.: Anomaly detection in cyber-physical system using logistic regression analysis. In: 2019 IEEE Texas Power and Energy Conference (TPEC), pp. 1–6. IEEE (2019)

24. Pedregosa, F., et al.: Scikit-learn: machine learning in python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011)
25. Shalev-Shwartz, S., et al.: Online learning and online convex optimization. *Found. Trends Mach. Learn.* **4**(2), 107–194 (2011)
26. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: *ICISSp*, pp. 108–116 (2018)
27. Shi, Z., Li, J., Wu, C., Li, J.: DeepWindow: an efficient method for online network traffic anomaly detection. In: *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 2403–2408. IEEE (2019)
28. Siddiqui, A.J., Boukerche, A.: Adaptive ensembles of autoencoders for unsupervised IoT network intrusion detection. *Computing* **103**, 1–24 (2021)
29. Teoh, T., Chiew, G., Franco, E.J., Ng, P., Benjamin, M., Goh, Y.: Anomaly detection in cyber security attacks on networks using MLP deep learning. In: *2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE)*, pp. 1–5. IEEE (2018)