**Noname manuscript No.**
(will be inserted by the editor)

# An Intrusion Detection Framework based on Hybrid Multi-Level Data Mining

**Haipeng Yao · Qiyi Wang · Luyao Wang · Peiying Zhang · Maozhen Li · Yunjie Liu**

the date of receipt and acceptance should be inserted later

**Abstract** With the dramatic opening-up of network, network security becomes a severe social problem with the rapid development of network technology. Intrusion Detection System (IDS) is an innovative and proactive network security technology, which becomes a hot topic in both industry and academia in recent years. There are four main characteristics of intrusion data that affect the performance of IDS including multicomponent, data imbalance, time-varying and unknown attacks. We propose a novel IDS framework called HMLD to address these issues, which is an exquisite designed framework based on Hybrid Multi-Level Data Mining. In this paper, we use KDDCUP99 dataset to evaluate the performance of HMLD. The experimental results show that HMLD can reach 96.70% accuracy which is nearly 1% higher than the recent proposed optimal algorithm *SVM+ELM+Modified K-Means*. In details, HMLD greatly increased the detection accuracy of DoS attacks and R2L attacks.

**Keywords** Intrusion Detection System · Multi-Level · Machine Learning · Data Engineering · KDDCUP99

## 1 Introduction

The development of network is a double-edged sword, which brings both convenience and network security problem to us. Therefore research about network security makes great sense. Traditional network security technologies such as firewall, data encryption, and user authentication system are all passive defense techniques. These methods have a good performance for known attacks, but not for unknown attacks. Different from traditional technologies, Intrusion Detection System (IDS) [1], which is firstly proposed by Denning et al. in 1986 [2], is an innovative and proactive network security technology, meaning that it can detect both known and unknown attacks. The basic architecture of IDS consists of three parts: data acquisition part, intrusion detection part and response process part. Data acquisition part of IDS is used to collect data from internet. The collected data, which consists of normal data and many different types of attacks, is then send to intrusion detection part. Intrusion detection part needs to pick out attack data from normal data and identify what type of the attack is. Response process part receives the detected attacks and processes them according to their types. As the core of IDS, intrusion detection part has become a hot topic in research in recent years.

The widely use of network produces massive data which is a valuable resource. From these data, we can extract much insightful information through data analysis techniques. We name the data collected by data acquisition part ‘intrusion data’. Intrusion data has some features that will affect the performance of IDS. We summary these features as follows:

- Multicomponent: The intrusion data is multicomponent because there exists many types of attacks in it. Hence intrusion detection is not a binary-classification problem, but a multi-classification problem, which is more complicated.
- Data imbalance: The number of attacks in different categories vary greatly from each other. It means that the intrusion data has a severe problem of data imbalance. Some attacks are difficult to be detected due to its sparseness.
- Time-varying: The distribution of data is time-varying. The variety between training data and detecting data will affect the performance of IDS.
- Unknown attacks: Some new attacks may appear as time goes by. These unknown attacks are difficult to be detected.

Haipeng Yao
Beijing University of Posts and Telecommunications
Tel.: +86-18611724130
E-mail: yaohaipeng@bupt.edu.cn

In recent years, researchers begin to use the artificial intelligence (AI) technology to deal with the problems caused by the above mentioned features. Machine learning (ML), whose main idea is building model to mine information from data, is a core subfield of AI including many algorithms. It can be roughly divided into supervised learning algorithms and unsupervised learning algorithms according to whether it has a training phase or not. Supervised learning is the most widely-used technique in machine learning such as *Support Vector Machine* (*SVM*) [3], *Artificial Neural Network* (*ANN*) [4], *Decision Tree* (*DT*) [5], *Random Forest* (*RF*) [6] and so on. Unsupervised learning mainly refers to clustering algorithms [7] such as *K-Means*[8], *DBSCAN*[9], *Affinity Propagation* (*AP*)[10] and so on. Every machine learning algorithm has its own advantages and disadvantages. Combining two or more algorithms together and taking full use of their advantages will increase the performance of IDS. Therefore, how to combine these algorithms scientifically is a noteworthy topic.

In the work of [11], Gang Wang et al. proposed a FCANN framework, which firstly clusters data using *Fuzzy C-means* algorithm, and then classifies each cluster by *ANN*. This framework integrates unsupervised learning and supervised learning to shorten the modelling time, alleviate the data imbalance problem and increase the detection rate. In literature [12], Prasanta Gogoi et al. proposed a MLH-IDS framework which has three layers, including a supervised layer, an unsupervised layer and an outlier-based layer. The supervised layer is used to detect DoS [13] and Probe [13] attacks, and the unsupervised layer is used to detect Normal data. The outlier-based layer is used to distinguish R2L [13] and U2R [13] attacks from each other. The hybrid multi-level framework takes full advantage of different ML algorithms, which is more flexible and has a better performance.

An appropriate scheme of data engineering can also improve the performance of IDS. Data engineering is an indispensable procedure in data mining which includes some widely used techniques such as data preprocessing and dimension reduction [14]. Data preprocessing techniques such as data cleaning and normalization can help remove ‘dirty data’ and turn data into suitable form. The representative method of dimension reduction is feature selection which is used to remove interfering and redundant features to improve the performance of data mining project. In IDS, the ‘intrusion data’ is usually not suitable to be detected directly, which needs to be processed by an appropriate data engineering method. Most works we mentioned above mainly focus on the combination of ML algorithms without an elaborately designed data engineering method.

In this paper, we propose a novel IDS framework named HMLD through jointly considering data engineering and machine learning. HMLD consists of three modules, including Multi-Level Hybrid Data Engineering (MH-DE) module, Multi-Level Hybrid Machine Learning (MH-ML) module, and Micro Expert Modify (MEM) module. The MH-DE module focuses on data engineering and the MH-ML module focuses on machine learning. These two modules construct a closed cycle of Hybrid Multi-Level Data Mining, which provides a separated and customized detection of different attacks. The hierarchical architecture can address the problems caused by multicomponent and data imbalance. After performing the procedure of MH-DE and MH-ML, most easily detected attacks have been marked. MEM module is used to identify those new attacks which are difficult to detect. The HMLD framework can be implemented in a variety of networks by using different algorithms and parameters.

In this paper, we use KDDCUP99 dataset to evaluate the performance of HMLD. Experimental results show that HMLD can achieve 96.70% accuracy which is nearly 1% higher than the recent proposed optimal algorithm *SVM+ELM+Modified K-Means* [15]. Meanwhile, it has a better performance than some other methods in identifying DoS and R2L attacks.

The remainder of this paper is organized as follows. Section 2 illustrates the framework of HMLD. Section 3 details the algorithms and parameters of HMLD when using KDDCUP99. Section 4 evaluates the performance of HMLD and compares it with some prior works. We conclude our work in Section 5.

## 2 The Framework of HMLD

In this section, we will introduce the framework and workflow of HMLD which can detect different categories of attacks separately by different data engineering methods and machine learning methods. The framework of HMLD is illustrated in Fig. 1. The input of HMLD contains two parts, one is the labeled training dataset *Dtrain* which is used to train the ML model, the other one is the unlabeled detecting dataset *Ddetect* which is waiting to be detected. *Dtrain* is processed by three modules: MH-DE, MH-ML and MEM as shown in the Fig. 1, to construct models that are used to detect intrusions in *Ddetect*. The output of HMLD is the labeled detecting dataset *Ddetect_Label*. Algorithm 1 is the pseudo code of HMLD which illustrates the step-by-step workflow of HMLD. We assume that there are $N$ attack categories in input data. The set of attacks in category $i$, where $i \in [1, N]$, is denoted as $S_i$. Each attack category has a corresponding package which is denoted as $Attack_i\_DS$. We define the data engineering method used on $Attack_i\_DS$ as $D_i$, and the ML model trained by $Attack_i\_DS$ is denoted by $M_i$. $P_{\_key}$, $P_{\_nonkey}$ and $P_i$ are intermediate variables in algorithm 1.

In the initialization phase, we define $D$ as $\{D_1, D_2, \cdots, D_N\}$, which is a set of $D_i$, *Ddetect_Label* as $\emptyset$ in which $\emptyset$ means an empty set, and $P_0$ as *Dtrain*. After finishing the initialization phase, *Dtrain* is firstly sent to MH-DE module. The 4-10 lines in algorithm 1 show the workflow of MH-DE module. The line 5 means we will build packages and using data engineering techniques to attacks in category $i$ where $i \in [1, N]$ one by one. The 6-8 lines in algorithm 1 show how to build package $Attack_i\_DS$. We label the attacks belonging to $S_i$ with $i$ in *Dtrain* and denote this part of data as $P_{\_key}$. Then we label remaining data which is belong to $P_{i-1}$ but not belong to $S_i$ with 0 and denote this part of data as $P_{\_nonkey}$. Thus package $Attack_i\_DS$ is comprised of $P_{\_key}$ and $P_{\_nonkey}$ by using $D_i$ to them which is shown in line 8. This step can make the packages be more suitable for ML modelling by converting format and removing redundancy. Package $P_i$ is constructed by filtering out $P_{\_key}$. We repeat this labeling process to build $N$ packets in sequence.

After MH-DE, these packages are sent to MH-ML module. The 11-17 lines in algorithm 1 show the workflow of MH-ML module. In MH-ML, each package is used as a training dataset to train an appropriate ML model as shown in line 13 in Algorithm 1. The trained model $M_i$ aims to correctly detect attacks in category $i$ as much as possible. In detecting phase, we use $M_1, M_2, \cdots, M_N$ one by one to mark and filter out attacks in different categories from *Ddetect* which is showed in line 14-16 in Algorithm 1.

After this filtering procedure, the remaining unmarked data in *Ddetect* are named *impurity data. Impurity data* mixes a large amount of normal data with some difficult detected unknown attacks. The 18-20 lines in algorithm 1 show the workflow of MEM module. We send *impurity data* to MEM module. MEM module samples micro data from the *impurity data* and marks them by experts to form a *modify set.* Then MEM module merges the *modify set* with *Dtrain* to train a new model to identify the unknown attacks in the *impurity data.* After finishing all the procedures, we can get *Ddetect_Label*, which is the result of our detecting work.

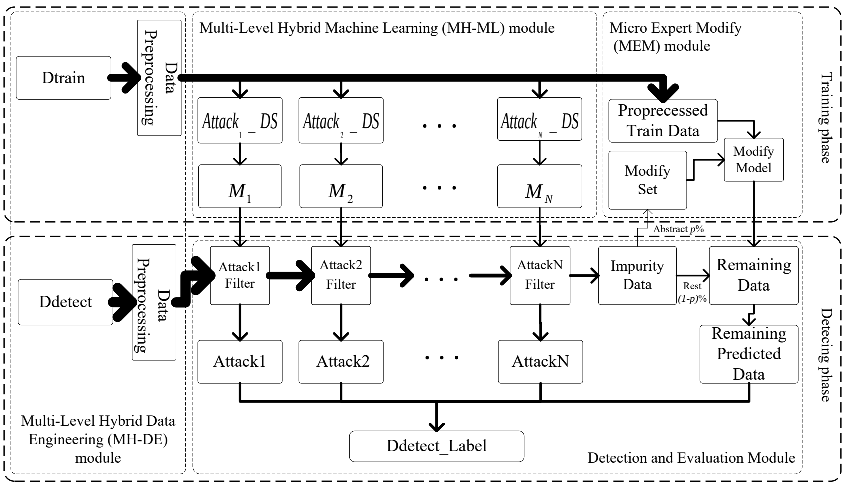

Fig. 1: Framework of HMLD

**Algorithm 1** HMLD

```
1: **Input**: $Dtrain$, $Ddetect$
2: **Output**: $Ddetect\_Label$
3: Initialization: $D = \{D_1, D_2, \cdots, D_N\}$, $Ddetect\_Label = \emptyset$, $P_0 = Dtrain$
4: **MH-DE module**
5: **for** $i \in [1, N]$ **do**
6:         $P_{\_key} = [data.label = i | data \in S_i]$
7:         $P_{\_nonkey} = [data.label = 0 | data \notin S_i \& data \in P_{i-1}]$
8:         $Attack_i\_DS = do\ D_i\ to\ (P_{\_key} + P_{\_nonkey})$
9:         $P_i = P_i - P_{\_key}$
10: **end for**
11: **MH-ML module**
12: **for** $i \in [1, N]$ **do**
13:         $M_i$ = the ML model trained using $Attack_i\_DS$
14:         Use $M_i$ to detect $Ddetect$ and label the detected attacks as $i$
15:         $Ddetect\_Label = Ddetect\_Label +$ detected attacks with label $i$
16:         $Ddetect = Ddetect -$ detected attacks with label $i$
17: **end for**
18: **MEM module**
19: Extract $p\%$ data from the remaining data in $Ddetect$ and mark them by experts forming *modify set*
20: Merge *modify set* and $Dtrain$ to retrain a ML model to detect the rest of $Ddetect$
21: **return** $Ddetect\_Label$
```

## 3 HMLD with KDDCUP99

The HMLD framework can be applied in many different types of networks by using different algorithms and parameters. In this paper, we evaluate the performance of the HMLD framework via KDDCUP99 [13] dataset. This section describes the selection details of algorithms and parameters of each module when using KDDCUP99.

### 3.1 KDDCUP99 Dataset

Firstly, we give a brief description of KDDCUP99 [13] dataset. KDDCUP99 Intrusion Detection Dataset is a classic dataset, which has 41 features and 5 types of labels. The 41 features contains *duration*, *protocol_type*, *service* and so on. The 5 types of labels are Normal, DoS, Probe, U2R, and R2L, respectively. The meaning of the labels are summarized in Table 1:

Table 1: The definitions of five labels

| Label | Meaning |
|---|---|
| Normal | normal data |
| DoS | denial-of-service |
| Probe | surveillance and probing |
| R2L | unauthorized access from a remote machine to a local machine |
| U2R | unauthorized access to local super-user privileges by a local machine |

Table 2 shows some examples of original data in KDDCUP99.

Table 2: Original data in KDDCUP99

| Data id | |
|---|---|
| 1 | 0,tcp,http,SF,181,5450,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0,0,0,0,1,0,0,9,9,1,0,0.11,0,0,0,0,0 |
| 2 | 0,icmp,ecr_i,SF,1032,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,511,511,0,0,0,0,1,0,0,255,255,1,0,1,0,0,0,0,0 |
| 3 | 0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,107,9,1,1,0,0,0.08,0.07,0,255,9,0.04,0.07,0,0,1,1,0,0 |

In this article , we extract 10% data from *kddcup.data_10_percent.gz* [13] as training data, and use *corrected.gz* [13] as testing data. Table 3 shows the number of each category of attacks.

Table 3: The distribution of training data and testing data

| Dataset | Normal | DoS | Probe | R2L | U2R | Sum |
|---|---|---|---|---|---|---|
| Training data | 9725 | 39164 | 417 | 111 | 3 | 49420 |
| Testing data | 60593 | 229853 | 4166 | 16189 | 228 | 311028 |

DoS, Probe, U2R and R2L are four attack categories. Each of them can be further separated into many subcategories. There are 22 subcategories in the training data and 39 subcategories in the testing data. The 17 new subcategories can be considered as new or unknown attack subcategories. Among the four types of attacks, R2L and U2R are much more difficult to be detected than DoS and Probe [12]. R2L is difficult to be detected because it includes many new subcategories of attacks. U2R is difficult to detect due to its sparseness.

### 3.2 MH-DE Module

MH-DE module focuses on data engineering which contains the stage of *basic data preprocessing* and the stage of *hybrid feature selection*. *Basic data preprocessing* is used to make the data suitable for modelling and *hybrid feature selection* is used to remove redundant features.

We design the process of *basic data preprocessing* according to the feature of KDDCUP99 dataset. There are three types of features in KDDCUP99, including factorial type, continuous numerical type and discrete type. Therefore, the *basic data preprocessing* for KDDCUP99 includes the *factorial feature digitizing* procedure and the *continuous feature normalizing* procedure. The former one will map the factorial features into numbers. Without this digital procedure, the dataset cannot be trained by a ML algorithm. The *continuous feature normalizing* procedure normalizes all features to the range of $[0, 1]$. This step can eliminate the effect caused by the diverse range of features.

In the stage of *hybrid feature selection*, we adopt different feature selection methods for different packages according to the category of attacks. The flow chart of MH-DE is demonstrated in Fig. 2. We number the attacks in category DoS, Probe, U2R and R2L as 1, 2, 3, 4, respectively. In MH-DE module, we firstly pick out all the DoS attacks and label them with 1. At the same time, we name the set of other data as 'Other1' and label them as 0. Then do feature selection work to the data. The selected features can distinguish DoS attacks from other data to the utmost. The authors of [16] gave the results of the optimal feature selection subsets of KDDCUP99 dataset, which is depicted in Table 4. The numbers in Table 4 represent the indexes of features in KDDCUP99. We use DoS feature selection subset in Table 4 as the feature selection subset to get DoS_DS. After that, we pick out all the Probe attack data from 'Other1' and label them with 2. In the meanwhile, we name the remaining data as 'Other2' and label them with 0. Then we use Probe feature selection subset in Table 4 to get Probe_DS. We repeat the this procedure for U2R and R2L, respectively. After finishing the procedure of MH-DE module, these five packages namely DoS_DS, Probe_DS, U2R_DS, R2L_DS and Train_DS are formed for subsequent training.

Table 4: FMIFS feature selection results of KDDCUP99

| Attack | Feature Selection Subset Generated by FMIFS |
|---|---|
| DoS | 2,3,5,6,8,12,23,24,31,32,36,37 |
| Probe | 3,4,5,17,19,22,24,25,27,28,29,30,32,33,34,35,37,40,41 |
| U2R | 1,2,3,4,6,7,8,12,13,14,15,16,17,18,19,20,21,22,29,31,32,37,40 |
| R2L | 1,3,5,6,8,9,10,11,15,17,22,23,24,32,33 |
| Overall | 2,3,5,6,9,12,17,23,24,26,29,31,32,33,34,35,36,37,39 |

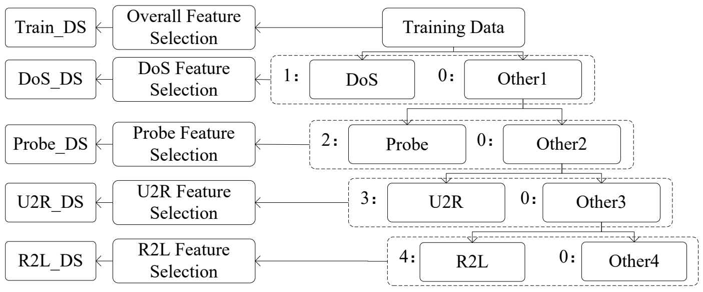

Fig. 2: Flow chart of MH-DE

## 3.3 MH-ML Module

After finishing the MH-DE module, the aforementioned five packages are sent into MH-ML module. Each package needs to build a model to filter out its corresponding category of attacks as many as possible. The authors of [11] proposed a modelling framework including a clustering phase and a classifying phase. In HMLD, we adopt this framework as our modelling framework because it can shorten the modelling time and alleviate data imbalance problem. The basic model building process is shown in Fig. 3. In this framework, the training data is firstly clustered by an unsupervised clustering algorithm to get many clusters. Towards each cluster, we train a specific supervised ML classifier.

The selection of algorithms and parameters of MH-ML module needs to be elaborate designed. We will use experiments to choose the algorithms and parameters for MH-ML when using KDDCUP99 dataset.

In clustering phase, we adopt the *K-Means* [8] algorithm thanks to its good performance and fast computation speed. The main idea of *K-Means* is clustering data into several clusters according to their similarity. We define the number of clusters as $k$, which has an impact on the performance of HMLD. Figs 4, 5, 6, 7 show the *Precision* of HMLD with $k$ ranging from 0 to 50 for package DoS_DS, Probe_DS, U2R_DS, R2L_DS, respectively. The *Precision* of detecting attacks represents the proportion of predicted attacks which are actually attacks. When $k$ is 0, it means that we do not use the *K-Means*. DoS attacks can reach highest *Precision* when $k$ is equal to 30 and 40. Therefore, we set $k$ to 30 for DoS attacks because a smaller $k$ can reduce computing resources and shorten modelling time. Probe attacks can achieve 91.91% *Precision* when $k$ is 0 and 92.08% when $k$ is 20. Though the *Precision* is a bit lower when $k$ is 0, the modelling complexity can be significantly reduced. Hence we set $k$ to 0 for Probe attacks. We set $k$ to 20 and 30 for U2R and R2L attacks respectively to maximize the performance, which can be observed from Figs 6, 7.

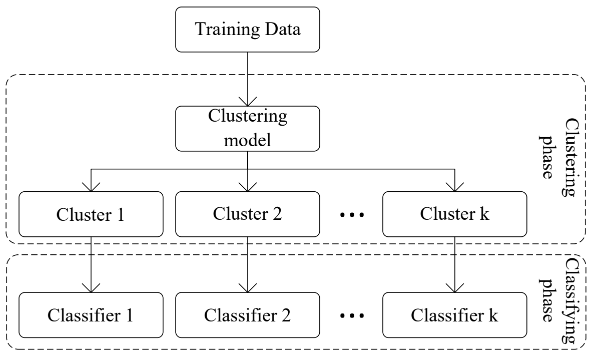

Fig. 3: Framework of model building in MH-ML

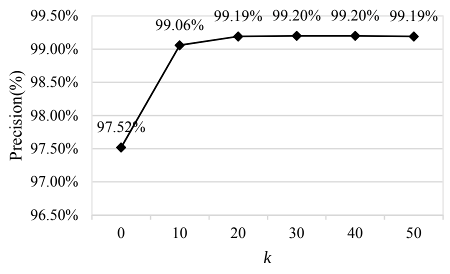

Fig. 4: Different $k$ for DoS_DS

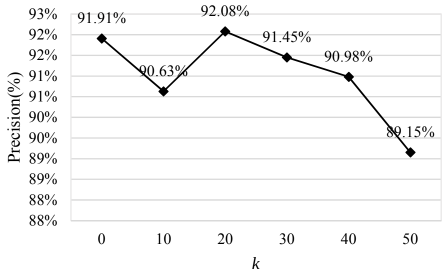

Fig. 5: Different $k$ for Probe_DS

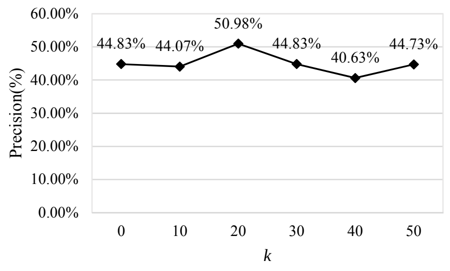

Fig. 6: Different $k$ for U2R_DS

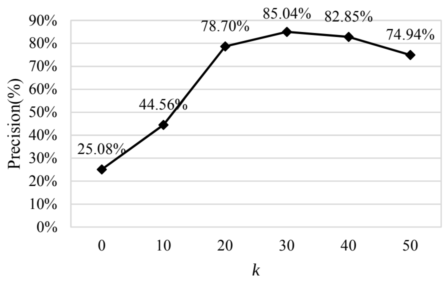

Fig. 7: Different $k$ for R2L_DS

In classifying phase, we compare many ML algorithms including *SVM*[3], *ANN* [4], *DT*[5] and *RF*[6] with different parameters. These algorithms are all supervised learning algorithms which are mainly used in classification problems. The main idea of *linear SVM* is to find a support hyperplane which can separate different types of data in best performance. If the data can be separated much better using a hypersurface than a hyperplane, we can use *kernel SVM* to replace *linear SVM*. The most frequently-used kernel of *SVM* is *rbf* kernel, which has another name as *Gaussian* kernel. Parameter $r$ is the variance of the *rbf* kernel. Parameter $C$ is the relaxation factor of *SVM*. Smaller $C$ will cause a decline of detection accuracy but can alleviate the overfitting problem. *ANN* is a multilayer neural network which contains an input layer, many hidden layers and an output layer. Each layer consists of many neurons which contains many parameters such as weights, bias and activation functions. In this paper, the number of neurons of the first hidden layer is 5 and the second hidden layer is 2. The activation functions can be *identity*, *logistic*, *tanh* or *relu*. *DT* algorithm computes the *information gain* of each feature using *Gini impurity* criteria and selects the biggest one as the root of a tree. Then repeat this procedure iteratively until the stop condition is satisfied. *CART* is a representative algorithm of *DT*. *RF* is a integration of *DT*, which samples data and features many times to build many trees. Then *RF* gets the final results by comprehensively considering all these trees.

Figs 8, 9, 10, 11 show the number of correctly detected attacks by using different algorithms with different parameters. From these figures, we can see that the performance of different algorithms varies greatly from each other for different attacks. We choose the appropriate algorithm according to two metrics, the number of detected attacks and the *Precision* of detecting attacks. The number of detected attacks can be observed from Figs 8, 9, 10, 11. A better algorithm can detect more attacks and can get a larger *Precision*.

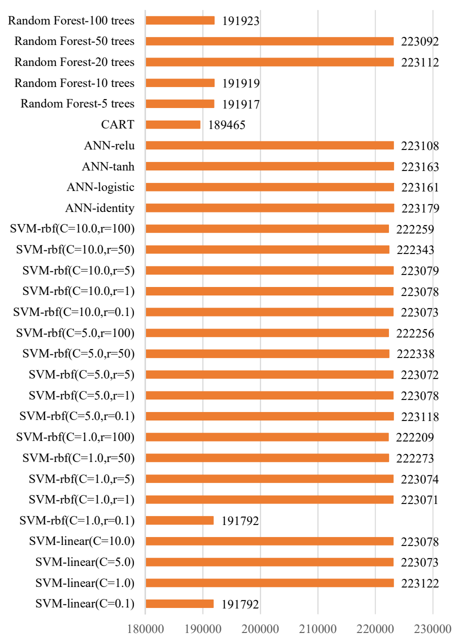

Fig. 8: Correctly Detected DoS Attacks

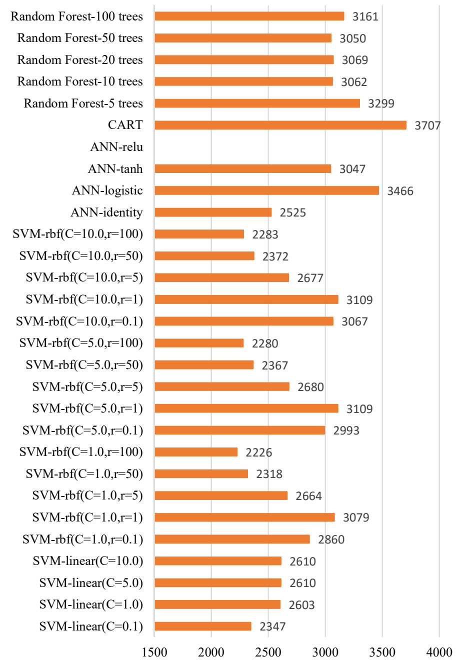

Fig. 9: Correctly Detected Probe Attacks

Hartigan1979A

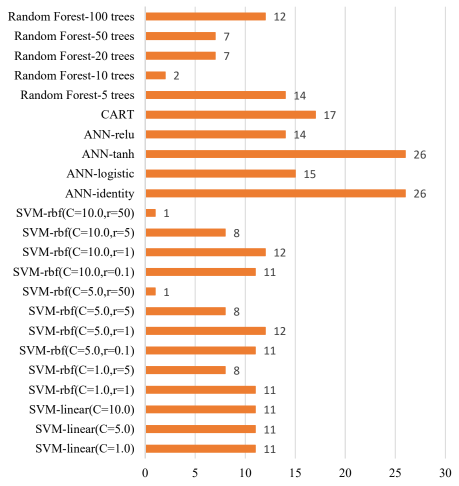

Fig. 10: Correctly Detected U2R Attacks

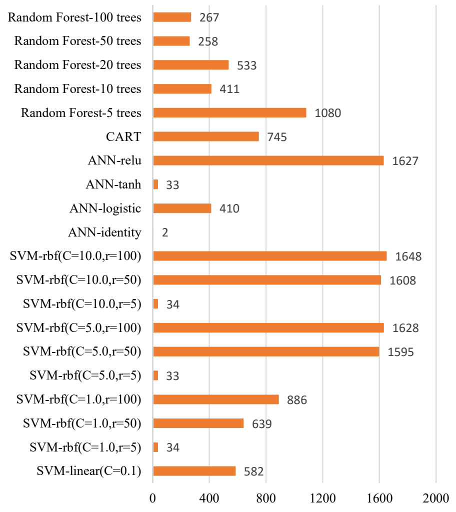

Fig. 11: Correctly Detected R2L Attacks

For DoS attacks, *SVM-linear (C = 1.0)*, *ANN-identity*, *ANN-logistic* and *ANN-tanh* all can detect more attacks than other algorithms. The *Precision* of DoS attack is 99.20% when using *SVM-linear* which is the highest among these four algorithms. Hence, we select *SVM-linear (C = 1.0)* as the classification algorithm of DoS_DS. For Probe attacks, *ANN-logistic*, *CART* can detect more attacks than others. Comparing their *Precision*, *CART* is 69.40% and *ANN -logistic* is 92.62%. For Probe attack, *ANN-logistic* algorithm is better. For U2R attacks, *ANN-tanh* and *ANN-identity* can detect more attacks than others. When using *ANN-tanh*, the *Precision* of U2R is 7.28%. But when using *ANN-identity*, the *Precision* is increased to 37.14%. Thus, we choose *ANN-identity* for U2R_DS. For R2L attacks, experiments show that *SVM-rbf* and *ANN-relu* have a better performance. However, the *Precision* is 84.2% when using *ANN-relu*, and 82.26% when using *SVM-rbf*. Therefore, we use *ANN-relu* algorithm for modelling of R2L_DS. The design of MH-ML module is shown in Fig. 12.

## 3.4 MEM module

After MH-ML, most of the attacks are marked and filtered out. We name the remaining unmarked data *impurity data*. The *impurity data* mixes a large amount of normal data with some unknown attacks. Before MEM, if we mark all the *impurity data* as Normal, we can derive the 'confusion matrix' as Table 5. In a 'confusion matrix',

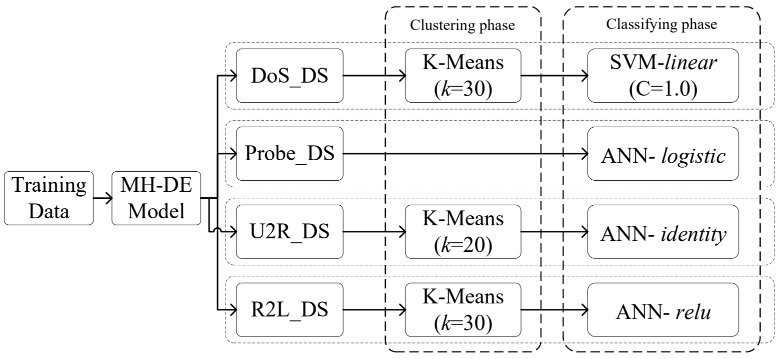

Fig. 12: Parameters and algorithms selection for MH-ML

each row represents the number of data which is actually this type, and each column represents data which is predicted as this type. For example, the number in the upper left corner is the number of data which is actual normal and also be predicted as normal. We can observe from Table 5 that a large amount of DoS attacks and R2L attacks are wrongly detected before MEM module because there appears many new subcategories of DoS and R2L attacks. Given that we have no information about these new attacks in training data, these new attacks are difficult to detect.

Table 5: Confusion matrix before MEM

| Actual vs Predict | Normal | DoS | Probe | R2L | U2R |
|---|---|---|---|---|---|
| Normal | 59391 | 711 | 202 | 272 | 17 |
| DoS | 6674 | 2231222 | 57 | 0 | 0 |
| Probe | 358 | 341 | 3466 | 0 | 1 |
| R2L | 13911 | 617 | 28 | 1626 | 7 |
| U2R | 48 | 122 | 18 | 14 | 26 |

In order to detect these new attacks efficiently, we send *impurity data* to MEM module. MEM module randomly samples $p\%$ of *impurity data* and marks them to construct the *modify set*. We use $DT$ [5] to retrain a ML model thanks to its rapid modelling speed. This model is used to detect the new attacks in *impurity data*. We experiment different value of $p$ to compare the average accuracy of HMLD, and the results are shown in Table 6. When $p\%$ is 0.3%, HMLD achieves a relative high performance. When $p\%$ is larger than 0.3%, the growth starts to slow down. Therefore, we set $p$ to 0.3.

Table 6: Comparisons of different $p$

| $p$ | 0.01% | 0.05% | 0.1% | 0.2% | 0.3% | 0.4% | 0.5% |
|---|---|---|---|---|---|---|---|
| Accuracy | 92.56% | 95.12% | 96.00% | 96.21% | 96.50% | 96.60% | 96.62% |

MEM module samples 0.3% data from *impurity data* and marks them by experts to form the *Modify Set*. Experimental results show that there are about 240 records in the *Modify Set*. After MEM, the ‘confusion matrix’ is showed in Table 7. With this micro cost, most of the unknown attacks of DoS and R2L are correctly detected.

Table 7: Confusion matrix after MEM

| Actual vs Predict | Normal | DoS | Probe | R2L | U2R |
|---|---|---|---|---|---|
| Normal | 56210 | 763 | 697 | 2699 | 39 |
| DoS | 192 | 229565 | 70 | 6 | 0 |
| Probe | 34 | 481 | 3615 | 35 | 1 |
| R2L | 4333 | 620 | 86 | 11102 | 9 |
| U2R | 28 | 122 | 32 | 20 | 26 |

## 4 Experimental results and discussions

### 4.1 Evaluation criteria

TP(true positives), TN(true negatives), FP(false positives), FN(false negatives) [16] are usually used to evaluate the performance of IDS. In IDS, we define normal data as positive and define attacks as negative. TP is the number of data which is actual positive and also predicted as positive. TN is the number of data which is actual positive but predicted as negative. FP is the number of data which is actual negative but predicted as positive. FN is the number of data which is actual negative and also predicted as negative. TP, TN, FP and FN can be written in a 'confusion matrix' as shown in Table 8.

Table 8: Confusion matrix

| | Predict as Positive | Predict as Negative |
|---|---|---|
| Actual is Positive | TP | TN |
| Actual is Negative | FP | FN |

*Precision, Recall, F-value* and *Accuracy* [16] are also defined to evaluate the performance. *Precision* given by Equation 1 is the proportion of predicted positives values which are actually positive. *Recall* given by Equation 2 is the proportion of the actual number of positives which are correctly identified. *Recall* also can be called as *Detection rate. F-value* given by Equation 3 is a harmonic mean between *Precision* and *Recall. Accuracy* given by Equation 4 is the proportion of correctly predicted data.

$$P = Precision = \frac{TP}{TP + FP}, \tag{1}$$

$$R = Recall = Detection\ \ rate = \frac{TP}{TP + FN}, \tag{2}$$

$$F - value = \frac{2 \times Precision \times Recall}{Precision + Recall}, \tag{3}$$

$$Accurary = \frac{TP + TN}{TP + TN + FN + FP}. \tag{4}$$

### 4.2 Experiments and analysis

**(1) Performance of HMLD**

In this part, we show the overall performance of HMLD in terms of evaluation indicators including *Precision, Recall, F-value* and *Accuracy.* Table 9 shows the *Precision*, *Recall* and *F-valve* of each attack. The *Accuracy* is 96.70% which can be computed by Equation 4.

Table 9: Indicators of HMLD

| Indicators(%) | Normal | DoS | Probe | R2L | U2R | Overall |
|---|---|---|---|---|---|---|
| Precision | 92.46 | 99.14 | 80.33 | 80.10 | 34.67 | 96.55 |
| Recall | 93.05 | 99.88 | 86.77 | 68.74 | 11.40 | 96.70 |
| F-value | 92.75 | 99.51 | 83.42 | 73.98 | 17.16 | 96.60 |

**(2) Comparisons of *hybrid feature selection* used by ML-DE with other feature selection methods**

In the ML-DE module of HMLD, we adopt *hybrid feature selection* as the feature selection method in data engineering. In contrast, we tested another two commonly used feature selection methods. One is doing the same feature selection to all packages, which called *unified feature selection.* When do *unified feature selection*, we use the feature subset of *Overall* in Table 4 to all packages. The other is not to do feature selection, which called *no feature selection.* Table 10 and Fig. 13 shows the comparisons of TP and *Precision* of different attacks by using *hybrid feature selection*, *unified feature selection* and *no feature selection.*

We can see that the overall performance of *hybrid feature selection* is much better than the other methods. From Table 10, the numbers of correctly detected attacks using *hybrid feature selection* are the highest for DoS, Probe and R2L. For U2R, the number is just 1 less than *no feature selection.* In terms of *Precision*, from Fig. 13 we can see that, the *Precision* using *hybrid feature selection* is much higher than the other two methods for

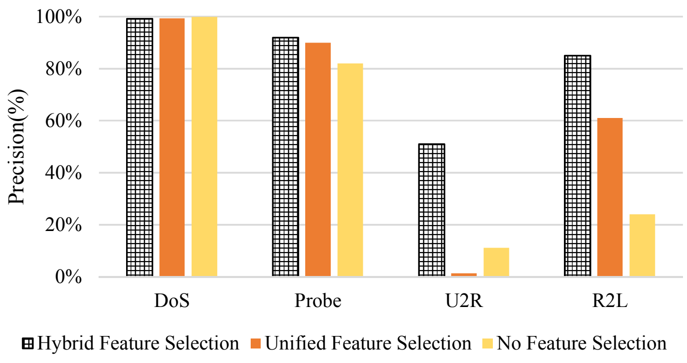

Fig. 13: Comparison of *Precision* for different attacks by using different feature selection methods

Probe, R2L and U2R. For DoS, the *Precision* is almost in the same level. The reason is that the characteristic of different attacks is reflected in different subset of features. Using customized subset of features can improve the performance on the corresponding attacks. Therefore, we can conclude that *hybrid feature selection* is far superior to other methods.

**(3) Comparisons of *Hybrid Multi-Level ML* used by MH-ML with single ML methods**

In MH-ML of HMLD, we use the method of selecting different ML algorithms for different packages according to their corresponding category of attacks. This method is called *Hybrid Multi-Level Machine Learning* method. In the meanwhile, we call the method of using the same ML algorithm such as *SVM*, *ANN* and *RF* to all packages as *Single Machine Learning* method. Fig. 14 contrasts the performance of using *Hybrid Multi-Level Machine Learning* and *Single Machine Learning*. From Fig. 14 we can observe that the *Recall* and *F-value* of using *Hybrid Multi-Level Machine Learning* method is much better than using *Single Machine Learning* methods. The *Precision* of using *Hybrid Multi-Level Machine Learning* method is higher than using *ANN* and *RF*, but a bit less than using *SVM*. Different algorithms has a different performance on detecting different attacks and *Hybrid Multi-Level Machine Learning* choose suitable algorithms to each category of attacks. Hence, the overall performance of using *Hybrid Multi-Level Machine Learning* is much better than using *Single Machine Learning*.

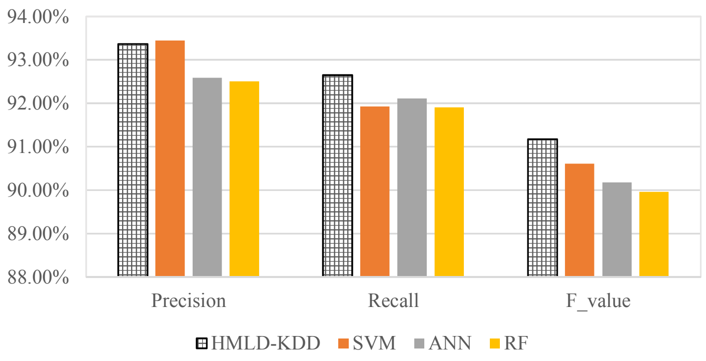

Fig. 14: Performance comparison of *Hybrid Multi-Level Machine Learning* and *Single Machine Learning*

**(4) Comparisons of HMLD with prior works**

In this subsection, we compare the performance of HMLD with some prior works [15,17,18] proposed in recent years. The authors of literature [15] proposed a modified *K-Means* to build a high-quality training dataset and train a multi-level hybrid intrusion detection model using *SVM* and *ELM*. The authors of literature [17] proposed an IDS based on *SVM*, hierarchical clustering algorithm and a simple feature selection procedure. The literature [18] is the method proposed by the winner of KDDCUP99 competition.

Table 10: Numbers of correctly detected attacks

| No.of Correctly Detected Attacks | DoS | Probe | U2R | R2L |
|---|---|---|---|---|
| Hybrid Feature Selection | **223122** | **3466** | 26 | **1626** |
| Unified Feature Selection | 222630 | 2662 | 4 | 748 |
| No Feature Selection | 208424 | 3355 | **27** | 252 |

Table 11 and Table 12 show comparisons between HMLD and these aforementioned works. In Table 11, each row represents the detection rates of different data types generated by one algorithm. Each column represents the detection rates of one data type by using different algorithms.

Table 11: *Detection rate(%)* generated by HMLD and some prior works

| Algorithms | Normal | DoS | Probe | R2L | U2R |
|---|---|---|---|---|---|
| HMLD-KDD | 93.05 | **99.88** | 86.77 | **68.74** | 11.40 |
| SVM+ELM+Modify K-Means | 98.13 | 99.54 | 87.22 | 21.93 | **31.79** |
| SVM+BIRCH clustering | 99.30 | 99.50 | **97.50** | 19.70 | 28.80 |
| Winner of the KDDCUP99 | **99.50** | 97.10 | 83.30 | 13.20 | 8.40 |

Table 12: *Accuracy(%)* comparisons of HMLD and some prior works

| Algorithms | Accuracy(%) |
|---|---|
| HMLD-KDD | **96.70** |
| SVM+ELM+Modify K-Means | 95.75 |
| SVM+BIRCH clustering | 95.70 |
| Winner of the KDDCUP99 | 93.30 |

According to Table 11 , HMLD has a high *Detection rate* of DoS and R2L attacks. Though the *Detection rate* of Normal and Probe is not the highest, they have the rates at or just below the average. The *Detection rate* of U2R is low, which is because the number of U2R attacks in training data is too small. From Table 7, we can see that most of the U2R attacks are predicted as DoS attacks. Though detected as wrong category, it is still classified as an attack category, not a normal class. From Table 12, we can observe that the overall performance of HMLD is better than the prior works, with a detection *Accuracy* reaching 96.70%, which is nearly 1% higher than the optimal in the other. In summary, the overall performance of HMLD model is better than prior works.

## 5 Conclusion

With the development and widely use of networks, network security becomes a social problem we should concern. In this paper, we propose a novel IDS framework called HMLD consisting of three modules, MH-DE module, MH-ML module and MEM module, which can be implemented in different types of networks. The combination of these three modules can detect both known and unknown attacks and improve the performance. In this paper, we use KDDCUP99 dataset to evaluate the performance of HMLD. The experimental results show that the performance of our method is better than prior works.

## References

1. Alriyami Q M, Asimakopoulou E and Bessis N, "A Survey of Intrusion Detection Systems for Mobile Ad Hoc Networks," in *International Conference on Intelligent Networking and Collaborative Systems(INCoS)*, Salerno, Italy, Sept. 2014, pp. 427-432.
2. Denning D E, "An Intrusion-Detection Model," in *IEEE Symposium on Security and Privacy*, Oakland, CA, USA, Apr. 1986, pp. 222-232.
3. Vapnik V and Cortes C, "Support vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273-297, Nov. 1995.
4. Zhang G, Patuwo B E and Hu M Y, "Forecasting with artificial neural networks:: The state of the art," *International Journal of Forecasting*, vol. 14, no. 1, pp. 35-62, Nov. 1998.
5. Quinlan J R, "Induction of Decision Trees," *Machine Learning*, vol. 1, no. 1, pp. 81-106, Nov. 1986.
6. Cutler A, Cutler D R and Stevens J R, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 157-176, Nov. 2004.
7. Jain A K, Murty M N and Flynn P J, "Data clustering: a review," *ACM Computing Surveys(CSUR)*, vol. 31, no. 3, pp. 264-323, Nov. 1999.
8. Hartigan J A and Wong M A, "A K-means Clustering Algorithm," *Applied Statistics*, vol. 28, no. 1, pp. 100-108, Nov. 1979.
9. Khan K, Rehman S U, Aziz K, Fong S and Sarasvady S, "DBSCAN: Past, present and future," in *Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT)*, Bangalore, India, Feb. 2014, pp. 232-238.
10. Wang K, Zhang J, Li D, Zhang X and Guo T, "Adaptive Affinity Propagation Clustering," *Acta Automatica Sinica*, vol. 33, no. 12, pp. 1242-1246, Nov. 2007.
11. Wang G, Hao J, Ma J and Huang L, "A new approach to intrusion detection using Artificial Neural Networks and fuzzy clustering," *Expert Systems with Applications*, vol. 37, no. 9, pp. 6225-6232, Nov. 2010.
12. Gogoi P, Bhattacharyya D K, Borah B and Kalita J K, "MLH-IDS: A Multi-Level Hybrid Intrusion Detection Method," *Computer Journal*, vol. 57, no. 4, pp. 602-623, Nov. 2014.
13. KDD Cup 1999 Data. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
14. Guyon I and Elisseeff A, "An Introduction to Variable and Feature Selection," *Journal of Machine Learning Research*, vol. 3, no. 6, pp. 1157-1182, Nov. 2003.

15. Al-Yaseen W L, Othman Z A and Nazri M Z A, “Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system,” *Expert Systems with Applications*, vol. 67, pp. 296-303, Nov. 2017.
16. Ambusaidi M, He X, Nanda P and Tan Z, “Building an Intrusion Detection System Using a Filter-Based Feature Selection Algorithm,” *IEEE Transactions on Computers*, vol. 65, no. 10, pp. 2986-2998, Nov. 2016.
17. Horng S J, Su M Y, Chen Y H, Kao T W, Chen R J, Lai J L and Perkasa C D, “A novel intrusion detection system based on hierarchical clustering and support vector machines,” *Expert Systems with Applications*, vol. 38, no. 1, pp. 306-313, Nov. 2011.
18. Elkan C, “Results of the KDD’99 classifier learning,” *ACM SIGKDD Explorations Newsletter*, vol. 1, no. 2, pp. 63-64, Nov. 2000.