Adaptive Deep Learning for Network Intrusion Detection by Risk Analysis

Lijun Zhang\textsuperscript{a,b}, Xingyu Lu\textsuperscript{a,b}, Zhaoqiang Chen\textsuperscript{a,b}, Tianwei Liu\textsuperscript{a,b}, Qun Chen\textsuperscript{a,b,∗}, Zhanhuai Li\textsuperscript{a,b}

\textsuperscript{a}School of Computer Science, Northwestern Polytechnical University, Xi’an, China
\textsuperscript{b}Key Laboratory of Big Data Storage and Management, Northwestern Polytechnical University, Ministry of Industry and Information Technology, Xi’an, China

Abstract

With increasing connectedness, network intrusion has become a critical security concern for modern information systems. The state-of-the-art performance of Network Intrusion Detection (NID) has been achieved by deep learning. Unfortunately, NID remains very challenging, and in real scenarios, deep models may still mislabel many network activities. Therefore, there is a need for risk analysis, which aims to know which activities may be mislabeled and why.

In this paper, we propose a novel solution of interpretable risk analysis for NID that can rank the activities in a task by their mislabeling risk. Built upon the existing framework of LearnRisk, it first extracts interpretable risk features and then trains a risk model by a learning-to-rank objective. It constructs risk features based on domain knowledge of network intrusion as well as statistical characteristics of activities. Furthermore, we demonstrate how to leverage risk analysis to improve prediction accuracy of deep models. Specifically, we present an adaptive training approach for NID that can effectively fine-tune a deep model towards a particular workload by minimizing its misprediction risk. Finally, we empirically evaluate the performance of the proposed solutions on real benchmark data. Our extensive experiments have shown that the proposed solution of risk analysis can identify mislabeled activities with considerably higher accuracy than the existing alternatives, and the proposed solution of adaptive training can effectively improve the performance of deep models by considerable margins in both offline and online settings.

Keywords: Network intrusion detection, risk analysis, adaptive deep learning.

1. Introduction

With increasing connectedness, network intrusion has become a critical security concern for modern information systems. Various network intrusion incidents reported in recent years \cite{1, 2} have brought about considerable financial and societal damage. The purpose of Network Intrusion Detection (NID) is to identify malicious activities in network traffic. As shown in Figure 1 where network activities are represented by records in a table, NID needs to distinguish abnormal network activities from normal ones based on their particular features (e.g., duration, protocol, service and state). In this example, it can be observed that \textit{r2} uses the udp protocol and the dns service with fixed-size source-to-destination bytes. Since \textit{r2} matches the characteristics of a generic attack, it is considered to be a network intrusion. However, \textit{r1} represents a normal activity. NID is usually considered as a binary classification problem tasked with labeling each activity as \textit{Normal} or \textit{Abnormal}. Even though NID can be performed based on expert rules and data mining \cite{3, 4}, its state-of-the-art performance has been achieved by deep learning \cite{5, 6, 7}.

Unfortunately, the efficacy of deep models depends on large quantities of accurately labeled training data, which may not be readily available in real scenarios. In addition, it is common in modern networks that traffic distribution continuously shifts. As a result, even a well trained deep model may still mislabel many activities in real scenarios. It is noteworthy that the limited interpretability of deep models further exacerbates the challenge. Therefore, there is a need for interpretable risk analysis, which can detect the activities mislabeled by a machine classifier and explain why.

\textsuperscript{∗}Corresponding author
Email: chenbenben@nwpu.edu.cn (Qun Chen)

Preprint submitted to Journal of \LaTeX\ Templates
February 20, 2022
Toward this end, we propose a novel solution of risk analysis for NID in this paper. Built upon the recently proposed framework of LearnRisk [8], the proposed solution first extracts interpretable risk features, then trains a risk model by a learning-to-rank objective and finally applies the learned risk model to rank activities by their misprediction risk. As shown in Figure 2, it consists of three steps: risk feature generation, risk model construction and risk model training. To ensure that risk features are interpretable, discriminating and high-coverage, we construct various risk metrics from different perspectives, including domain knowledge and statistical abnormalcy. Specifically, we leverage the CIA domain knowledge [9], which stands for Confidentiality, Integrity and Availability, to quantify security compromise. Confidentiality measures information concealment, Integrity measures information trustworthiness and Availability measures the ability to use desired information. In addition, we measure intrusion risk from the perspective of anomaly detection [10, 11, 12], based on the observation that intrusions are usually statistical outliers among a larger number of normal activities.

Furthermore, since risk analysis can measure the misprediction risk of a classifier on unlabeled activities, it provides classifier training with a viable way to adapt towards a particular workload. Therefore, we also present a novel solution of adaptive deep training for NID based on the results of risk analysis. As shown in Figure 3, it consists of two phases: the traditional training phase and the following adaptive training phase. In the first phase, it pre-trains a deep model in the traditional way based on labeled training data, while the second phase further fine-tunes the model towards unlabeled target data by minimizing misprediction risk. The main contributions of this paper can be summarized as follows:

1. We propose a novel solution of interpretable risk analysis for NID. In particular, we present a technique of risk feature generation that can effectively fuse various intrusion risk factors for risk measurement.
2. We present a novel solution of adaptive deep learning for NID, which can effectively tune a deep model towards a target workload by minimizing misprediction risk.
3. We empirically validate the efficacy of the proposed solutions on real benchmark data by a comparative study. Our experiments have shown that the proposed solution of risk analysis can identify the mislabeled activities.
### 2. Related Work

**Network Intrusion Detection.** Extensively studied in the literature [13] [14] [15], NID has been a research hotspot on network security. NID was traditionally considered as a classification problem and solved by various machine learning approaches, including Bayesian model [16], decision tree [3] [17], random forest [18] and SVM [4] [19] to name a few. More recently, deep learning [20] [21] [22] have been applied to NID and achieved the state-of-the-art performance. The proposed deep models included DNN [5] [23], CNN [24], RNN [25], BiLSTM [6] and DBN [26]. It has also been shown that the combination of stacked sparse autoencoder and a variant of RNN could achieve promising performance for NID [7]. It is noteworthy that the efficacy of these deep models depends on large quantities of labeled training data, which unfortunately may not be sufficiently available in real scenarios.

**Risk Analysis.** It has been well recognized that many classification tasks, including NID, are very challenging in real scenarios such that even a well trained deep model is prone to classification errors. Therefore, in recent years, there has been a growing interest in the study of risk analysis [27] [28] [29] [30] [31] [32] [8], alternatively called confidence ranking or trust scoring in the literature, which aimed to analyze misprediction risk by a machine classifier. Targeting image recognition, the authors of [27] proposed a simple baseline that employed the probabilities from softmax distributions. The authors of [30] instead proposed to measure prediction risk by comparing the distances of an instance to the cluster with the same label and its nearest cluster with a different label. More recently, we proposed an interpretable and learnable risk analysis framework of LearnRisk for the task of entity resolution [8]. LearnRisk is the first learnable approach for risk analysis, and is also more interpretable than its previous alternatives. In this paper, we have built the solution of risk analysis for NID based on the framework of LearnRisk.

**Model Training.** Over-fitting, which refers to the phenomenon that a model well-tuned on training data does not perform satisfactorily on target data, is a major challenge for model training. The de facto standard approach to alleviate over-fitting is to leverage validation data for model selection (e.g., cross validation [33]). An alternative

---

**Table 1: Model Training and Validation**

<table>
<thead>
<tr>
<th>Phase I: Traditional Training</th>
<th>Phase II: Risk-based Training</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Data</td>
<td>Classifier Parameter Tuning</td>
</tr>
<tr>
<td>Validation Data</td>
<td>Hyperparameter Tuning</td>
</tr>
<tr>
<td>Test/Target Data</td>
<td>Risk Model Parameter Tuning</td>
</tr>
<tr>
<td>Parameter Tuning</td>
<td>Risk Model Parameter Tuning</td>
</tr>
<tr>
<td>Test/Target Data</td>
<td>Risk Model Parameter Tuning</td>
</tr>
</tbody>
</table>

---

**Figure 3: Risk-based adaptive training for NID.**

with considerably higher accuracy than the existing alternatives, and adaptive deep learning can effectively improve the performance of deep models in both offline and online settings.

The rest of this paper is organized as follows: Section 2 reviews related work. Section 3 defines the NID task and introduces the LearnRisk framework. Section 4 presents the solution of risk analysis. Section 5 presents the solution of adaptive deep learning. Section 6 presents our empirical evaluation results. Finally, Section 7 concludes this paper with some thoughts on future work.
approach is by regularization \[34,35\], which tries to simplify a model to a manageable level by reducing the number of parameters.

Many adaptation approaches have also been proposed to alleviate distribution misalignment between training and target data, most notably among them transfer learning \[36,37,38\] and adaptive representation learning \[39,40,41,42,43\]. Transfer learning aimed to adapt a model learned on training data in a source domain to a target domain. Similarly, adaptive representation learning, which was originally proposed for image classification, studied how to learn domain-invariant features shared among diversified domains. However, distribution misalignment remains very challenging. One of the reasons is that the existing solutions focused on how to extract and leverage the common knowledge shared between a source task and a target task; unfortunately, they have limited capability to tune a model towards a target workload by its particular characteristics.

The approach of risk-based adaptive training was first proposed for the task of entity resolution in \[44\]. Unlike previous approaches, it can effectively adapt a model to a target workload by its particular characteristics. In this paper, we presented the solution of adaptive deep training for NID based on the same two-phase training framework. It is worthy to point out that although LearnRisk used the combination of risk features to measure misprediction risk, there are fundamental differences between risk-based adaptive training and ensemble learning \[45,46\] due to the following reasons: 1) different from traditional labeling functions, the purpose of LearnRisk is to estimate the misclassification risk of an instance as predicted by a classifier; 2) more importantly, the ensemble approach trains multiple models and tunes them based on training data; in contrast, risk-based adaptive training trains only one model and tunes its parameters towards a target workload. In principle, risk-based adaptive training can similarly work with an ensemble classifier; however, technical details need to be further investigated in future work.

3. Preliminaries

In this section, we define the task of risk analysis for NID and briefly introduce the LearnRisk framework.

3.1. Task Statement

In this paper, we considers NID as a binary classification problem. A classifier needs to label every unlabeled activity record as normal or abnormal. As usual, we measure classification quality by the metric of F1, which is a combination of precision and recall as follows:

\[
F1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
\]  

(1)

As usual, we suppose that an NID task of \(R\) consists of a set of labeled training data, \(R^t = \{(x_i^t, y_i^t)\}|t\), where each \((x_i^t, y_i^t)\) denotes a training instance with its feature vector of \(x_i^t\) and ground-truth label of \(y_i^t\), a set of labeled validation data, \(R^v = \{(x_i^v, y_i^v)\}|v\), and a set of unlabeled target data, \(R^n = \{(x_i^n, ?)\}|n\). Note that the validation data, \(R^v\), are provided as a proxy workload of \(R^n\). In this paper, we consider both offline and online settings. In the offline setting, the entire target workload of \(R^n\) is supposed to be available for classifier training, while in the online setting, the workload of \(R^n\) is only provided incrementally. Given a task of \(R\) consisting of \(R^t, R^v\) and \(R^n\), NID aims to learn an optimal classifier, \(M\), such that the performance of \(M\) on \(R^n\) in terms of \(F1\), or \(F1(M, R^n)\), is maximized.

Unfortunately, a machine classifier may mislabel some activities. The purpose of risk analysis is then to rank the activities in a given workload by their mislabeling risk such that mislabeled activities can be generally ranked before correctly labeled ones. Specifically, as in previous work \[27,47\], we use the metric of Receiver Operating Characteristic (ROC) to evaluate the performance of risk analysis. The metric of ROC illustrates the true positive rate, denoted by \(TPR\), against the false positive rate, denoted by \(FPR\), at different threshold settings. We denote the number of true positives by \(n_t\), the number of false positives by \(n_f\), the number of true negatives by \(n_t^c\), and the number of false negatives by \(n_f^c\). Then, we measure the true positive rate, \(TPR\), by \(\frac{n_t}{n_t + n_f^c}\), and the false positive rate, \(FPR\), by \(\frac{n_f^c}{n_t^c + n_f^c}\). With regard to risk analysis, a positive refers to a mislabeled activity while a negative refers to a correctly labeled one. The metric of ROC describes the relative tradeoff between benefit (true positives, i.e., mislabeled activities) and cost (false positives, i.e., correctly labeled activities). Through ROC, the Area Under ROC (AUROC) can be considered as the probability that a risk model assigns higher score to a randomly chosen positive than a randomly chosen negative.
Therefore, a model with a higher AUROC achieves better performance. The examples of ROC have been shown in Figure 4.

Formally, we define the task of risk analysis as follows:

**Definition 1. [Risk Analysis for NID].** Suppose that an NID task of $R$ consists of $R^c$, $R^v$ and $R^t$, and a classifier of $M$ has labeled each activity record in $R^t$. The task of risk analysis is to quantitatively measure the mislabeling risk of each record in $R^t$ such that the metric of AUROC is maximized.

### 3.2. The LearnRisk Framework

The LearnRisk framework consists of three essential steps: risk feature generation, risk model construction and risk model training.

#### 3.2.1. Risk feature generation

The first step automatically generates interpretable risk features in the form of rules by one-sided decision trees. It needs to ensure that the generated features are not only discriminative, i.e., each rule is highly indicative of one class label over the other, but high-coverage, i.e., its validity covers over a large subpopulation of the workload. As opposed to the traditional labeling functions classifying records into Normal or Abnormal, a rule of risk feature focuses exclusively on one single class. Therefore, a risk feature can be regarded as an indicator of the situation where the predictions of a classifier go against the knowledge embedded in it. An example of risk feature for NID is:

$$r_i[sttl] > 61.0 \land r_i[dpkts] > 135.0 \Rightarrow abnormal(r_i),$$

where $r_i$ denotes an activity record, $r_i[sttl]$ denotes its attribute value at $sttl$, i.e., the source-to-destination time to live, and $r_i[dpkts]$ denotes its attribute value at $dpkts$, i.e., the destination-to-source packet count. It has been well recognized that if a network activity takes longer time to live and more packets, it is more likely to be an intrusion. With this knowledge, an activity predicted as Normal, whose $sttl$ value is however greater than 61.0 and $dpkts$ value is greater than 135.0, is supposed to be at high risk of being misclassified.

#### 3.2.2. Risk model construction

The second step makes use of risk features to judge a classifier’s outputs backing up its decisions with human-friendly explanations. Towards this end, LearnRisk, drawing inspiration from investment theory, models the label probability distribution (portfolio reward) of a record as the aggregation of the distributions of its compositional features (stocks rewards).

Specifically, for the binary classification tasks such as NID, LearnRisk represents the label probability (label of Abnormal for NID) of a record, $r_i$, by a random normal variable, $p_i = \mathcal{N}(\mu_i, \sigma_i^2)$, where $\mu_i$ and $\sigma_i^2$ denote its expectation and variance respectively. Given a set of $m$ risk features, $\{f_1, f_2, ..., f_m\}$, we denote their corresponding expectation...
and variance vectors by $\mu_F = [\mu_{f_1}, \mu_{f_2}, \ldots, \mu_{f_m}]^T$ and $\sigma^2_F = [\sigma^2_{f_1}, \sigma^2_{f_2}, \ldots, \sigma^2_{f_m}]^T$, respectively. We also denote their corresponding feature weight vector by $w = [w_1, w_2, \ldots, w_m]$. Then, the distribution of a record, $r_i$, can be represented by:

$$\mu_i = z_i(w \circ \mu_F),$$

and

$$\sigma^2_i = z_i(w^2 \circ \sigma^2_F).$$

Where $\circ$ represents the element-wise product and $z_i$ denotes a one-hot feature vector.

It is worthy to point out that besides one-sided decision rules, LearnRisk also incorporates classifier output as one of risk features. To capture the fluctuation risk of label status, LearnRisk measures misprediction risk by the metric of Value-at-Risk (VaR) [48]. Provided with a confidence level of $\theta$, the metric of VaR represents the maximum loss after excluding the worst outcomes whose combined probability is at most $1-\theta$.

### 3.2.3. Risk model training

Finally, LearnRisk trains a risk model on labeled validation data by optimizing a learn-to-rank objective. It tunes the variances ($\sigma^2_i$) of risk features as well as their weights ($w_i$). As for their expectations ($\mu_i$), they are considered as prior knowledge, and thus estimated based on labeled training data. Once trained, a risk model can be used to assess the misclassification risk on a target workload labeled by a classifier.

### 4. Risk Analysis for NID

In this section, we present the solution of risk analysis. We first present how to generate risk features, and then describe how to train a risk model.

#### 4.1. Risk Feature Generation

Since risk features are generated based on basic risk metrics, their efficacy heavily depends on risk metrics. As shown in Figure 5, we construct basic risk metrics for NID based on both intrusion domain knowledge and statistical abnormalcy.

![Figure 5: Risk feature generation for NID.](image)
4.1.1. Risk Metrics of Domain Knowledge

With regard to NID, it has been widely recognized that domain knowledge can improve a machine classifier’s generalizability to unknown attacks as well as its interpretability. In this work, we leverage the popular principle of CIA for risk measurement. The CIA principle consists of three properties: C (Confidentiality) — information concealment, I (Integrity) — information trustworthiness, and A (Availability) — ability to use desired information. For example, eavesdropping unencrypted data harms confidentiality, an unauthorized attempt to change data compromises integrity, and a deliberate arrangement of denial to access data hampers availability.

Table 1: The example mappings between network attacks with CIA properties.

<table>
<thead>
<tr>
<th>Attack</th>
<th>Related CIA components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fuzzers</td>
<td>A</td>
</tr>
<tr>
<td>Analysis</td>
<td>C,I,A</td>
</tr>
<tr>
<td>Backdoors</td>
<td>C</td>
</tr>
<tr>
<td>DOS</td>
<td>A</td>
</tr>
<tr>
<td>Exploits</td>
<td>A</td>
</tr>
<tr>
<td>Generic</td>
<td>C</td>
</tr>
<tr>
<td>Reconnaissance</td>
<td>C</td>
</tr>
<tr>
<td>Shellcode</td>
<td>C,I,A</td>
</tr>
<tr>
<td>Worms</td>
<td>C,I,A</td>
</tr>
</tbody>
</table>

As in [49], we construct CIA risk metrics by two steps: attribute mapping and metrics construction. For simplicity of presentation, we denote the attribute set of activity records by \( T \), \( \{ t_1, t_2, \ldots, t_n \in T \} \).

Attribute Mapping. The step of attribute mapping analyzes the association of network attacks with CIA properties. For instance, a DOS attack usually hampers the availability of service by initiating an intensive “denial of service” request to a computer to exhaust its resources. Therefore, DoS is related to the A property. Worm is another common virus, which spreads by continuously gaining part or all of the control rights on a vulnerable computer. Since worm could destroy data and harm confidentiality, integrity, and availability, it is related to all the three properties of CIA.

Table 2: The example mappings between activity attributes and CIA properties.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Related attacks: the top-3 attributes</th>
<th>Related CIA component</th>
</tr>
</thead>
<tbody>
<tr>
<td>sbytes</td>
<td>Analysis, Backdoors, DOS, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode, Worms</td>
<td>C,I,A</td>
</tr>
<tr>
<td>sload</td>
<td>Analysis, Backdoors</td>
<td>C,I,A</td>
</tr>
<tr>
<td>ct_svc_src</td>
<td>Analysis</td>
<td>C,I,A</td>
</tr>
<tr>
<td>ct_dst_sport_ltm</td>
<td>Backdoors</td>
<td>C</td>
</tr>
<tr>
<td>smean</td>
<td>DOS, Generic, Reconnaissance, Shellcode, Worms</td>
<td>C,I,A</td>
</tr>
<tr>
<td>ct_svc_dst</td>
<td>DOS, Exploits</td>
<td>A</td>
</tr>
<tr>
<td>dbytes</td>
<td>Exploits</td>
<td>A</td>
</tr>
<tr>
<td>ct_dst_svc_ltm</td>
<td>Fuzzers</td>
<td>A</td>
</tr>
<tr>
<td>sttl</td>
<td>Fuzzers, Worms</td>
<td>C,I,A</td>
</tr>
<tr>
<td>dmean</td>
<td>Generic, Reconnaissance, Shellcode</td>
<td>C,I,A</td>
</tr>
</tbody>
</table>

As in [50], for each type of attack, we extract its top-3 most important attributes based on the information gain analysis of XGBoost. Then, we map each attribute to the three properties of CIA by connecting the mapping between attacks and CIA and the mapping between attacks and attributes. Some example mappings between attributes and CIA properties have been illustrated in Table 2.

Metrics Construction. Based on the results of attribute mapping, the step of metric construction estimates the correlation coefficients between attributes and label status. Specifically, we use the coefficient of Pearson correlation to quantify an attribute’s influence on label status as its value increases. Some examples of attributes and their estimated correlation coefficients have been shown in Table 3.
neighbors to its local reachability density as follows:

\[ \text{LOF}(r) = \frac{1}{|N_{r_k}|} \sum_{r_i \in N_{r_k}} \frac{LRD(r_i)}{LRD(r_i)} \]  

in which \( N_r \) denotes the set of \( r_i \)'s k nearest neighbors, and \( LRD(r_i) \) denotes \( r_i \)'s local reachability density.

4.1.2. Risk Metrics of Statistical Anomaly

Since NID can be considered as a task of anomaly detection, many effective machine learning approaches for anomaly detection can be naturally exploited for risk measurement. In this work, we use Local Outlier Factor (LOF), Isolation Forest (IF) and OneClassSVM to quantify an activity\'s degree of anomaly. In this work, we use Local Outlier Factor (LOF), Isolation Forest (IF) and OneClassSVM to quantify an activity\'s degree of anomaly.

Local Outlier Factor (LOF). LOF measures the probability of a point being an outlier from the density perspective \[10\]. Given a point \( r_i \), we denote its distance from its k-th nearest neighbor by k-distance(\( r_i \)). Accordingly, the reachability distance from \( r_i \) to another point \( r_j \), denoted by reach-dist(\( r_i, r_j \)), is defined by the maximum of k-distance(\( r_i \)) and the direct distance between \( r_i \) and \( r_j \). Then, the local reachability density of the point \( r_i \) is defined by the reciprocal of the average reachability distance between \( r_i \) and its k nearest neighbors.

According to the definition of local reachability density, a small local reachability density means a longer distance from other points. The metric of LOF measures a point\'s degree of anomaly by its relative density compared with its neighbors. Formally, the LOF score of a point \( r_i \) is defined by the ratio of the average local reachability density of its neighbors to its local reachability density as follows:

\[ \text{LOF}(r_i) = \frac{\sum_{r_j \in N_{r_i}} \frac{LRD(r_j)}{|N_{r_i}|}}{LRD(r_i)} \]  

in which \( N_{r_i} \) denotes the set of \( r_i \)'s k nearest neighbors, and \( LRD(r_i) \) denotes \( r_i \)'s local reachability density.

Table 3: The example correlation coefficients between attributes and label status.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Correlation coefficient</th>
<th>Influence on label status</th>
</tr>
</thead>
<tbody>
<tr>
<td>sbytes</td>
<td>-0.34</td>
<td>negative</td>
</tr>
<tr>
<td>sload</td>
<td>0.30</td>
<td>positive</td>
</tr>
<tr>
<td>ct_srv_src</td>
<td>0.24</td>
<td>positive</td>
</tr>
<tr>
<td>ct_dst_sport_ltm</td>
<td>0.46</td>
<td>positive</td>
</tr>
<tr>
<td>smean</td>
<td>-0.06</td>
<td>negative</td>
</tr>
<tr>
<td>ct_srv_dst</td>
<td>0.24</td>
<td>positive</td>
</tr>
<tr>
<td>dbytes</td>
<td>-0.47</td>
<td>negative</td>
</tr>
<tr>
<td>ct_dst_src_ltm</td>
<td>0.29</td>
<td>positive</td>
</tr>
<tr>
<td>sttl</td>
<td>0.66</td>
<td>positive</td>
</tr>
<tr>
<td>dmean</td>
<td>-0.52</td>
<td>negative</td>
</tr>
</tbody>
</table>

Complexity Analysis. It can be observed that the key process of attribute mapping is to select the most relevant attributes for each network attack based on XGBoost. With the number of trees denoted by \( n_T \), the maximum tree depth by \( h \) and the size of training data by \( n \), the complexity of XGBoost can be represented by \( O(n_T \times h \times n \times \log(n)) \) \[51\]. As a result, supposing that \( m \) denotes the total number of types of network attacks, we can represent the complexity of attribute mapping by \( O(m \times n_T \times h \times \bar{n} \times \log(\bar{n})) \). For metrics construction, the complexity of Pearson correlation calculation can be represented by \( O(\bar{n}) \). Suppose that the size of all data is denoted by \( n \) and the number of attributes by \( n_A \), then the complexity of metrics construction can be represented by \( O(n_A \times (n + \bar{n})) \). Therefore, without loss of generality, the time complexity of constructing risk metrics of CIA can be represented by \( O(m \times n_T \times h \times \bar{n} \times \log(\bar{n}) + n_A \times (n + \bar{n})) \).

4.1.2. Risk Metrics of Statistical Anomaly

Since NID can be considered as a task of anomaly detection, many effective machine learning approaches for anomaly detection can be naturally exploited for risk measurement. In this work, we use Local Outlier Factor (LOF), Isolation Forest (IF) and OneClassSVM to quantify an activity\’s degree of anomaly.
It can be observed that if a point’s LOF score is less than 1, it is in a relatively dense area. However, if its LOF score is much greater than 1, it is usually distant from other points, thus very likely to be an outlier.

**Isolation Forest.** IF defines anomalies as the outliers that can be easily isolated \([11]\), i.e., the points that are sparsely distributed and far away from the dense clusters of points. It isolates points by constructing a tree structure. Since anomalies are easier to be isolated, they tend to have a shorter path length to the root. In contrast, normal points can only be isolated at the deepest parts of the tree. Consider the example from CICIDS2017 dataset as shown in Figure 6: the normal point \(r_1\) requires 6 random partitions for its isolation, while the abnormal point \(r_2\) can be isolated with only 2 random partitions.

![Figure 6](image)

Figure 6: The random partitioning visualization of a normal point versus an abnormal one.

Given a point \(r_i\) in the set of \(R\), its isolation length, denoted by \(IL(r_i)\), refers to the number of edges traversed from the root to its corresponding leaf node in an isolation tree. Formally, the IF score of \(r_i\), \(IF(r_i)\), is measured by

\[
IF(r_i) = 2^{\frac{E(IL(r_i))}{IL(R)}},
\]

in which \(E(IL(r_i))\) denotes the expectation of \(IL(r_i)\) based on a collection of isolation trees, and \(IL(R)\) denotes the average isolation length of the points in \(R\). According to the IF measurement, if a point’s IF score is close to 1, it is highly likely to be abnormal; in contrast, a score less than 0.5 usually means that it can be safely regarded as a normal point.

**SVM.** SVM is another popular technique for anomaly detection. For risk measurement, we leverage OneClassSVM \([12]\), which maps points into a feature space using a kernel function, and then constructs the smallest hyperplane to capture most normal points. Provided with a new point \(r_i\), it determines which side of the hyperplane it falls by a decision function in the feature space. Formally, the decision function of \(DF(r_i)\) is defined by

\[
DF(r_i) = \text{sgn}((\omega \cdot \Phi(r_i)) - \rho),
\]

where \(\omega\) and \(\rho\) denote model parameters and \(\Phi(r_i)\) is the feature map of \(r_i\). Learning from training data in the form of soft margins, OneClassSVM considers a new point that falls outside the learned boundary as an anomaly. We measure risk by OneClassSVM output probability.

**Complexity Analysis.** The time complexity of generating statistical risk metrics is provided as as follows:

- **Complexity of LOF.** Let \(n\) denote the dataset size. The process of LOF consists of two steps: the first one finds the MinPtsUB-nearest neighbors, while the second one computes the LOF scores. According to the analysis provided in \([10]\), the first step has the time complexity of \(O(n \times \log(n))\), and the second step has the time complexity of \(O(n)\). Hence, the total time complexity of LOF can be represented by \(O(n \times \ln(n))\).

- **Complexity of IF.** The process of IF consists of two steps: the first one builds a forest of \(n_F\) trees, in each of which \(\psi\) points are uniformly sampled from a workload, while the second one calculates the anomaly score of
each point. According to the analysis provided in [11], with the dataset size denoted by $n$, the time complexities of the two steps can be represented by $O(n_T \times \psi \times \ln(\psi))$ and $O(n_T \times n \times \ln(\psi))$ respectively. Hence, the total time complexity of IF can be represented by as $O(n_T \times (n + \psi) \times \ln(\psi))$. Note that in the default setting of our scikit-learn implementation [1] the parameters of $n_T$ and $\psi$ are set to be 100 and 256 respectively. As a result, the time complexity can be considered to be approximately linear.

- **Complexity of SVM [52]:** We have implemented OneClassSVM based on scikit-learn. According to [52], it has a linear complexity w.r.t dataset size.

4.1.3. Risk Feature Generation.

With the risk metrics of domain knowledge and statistical anomaly as well as all the attribute values of activity records, our solution generates risk features by one-sided decision trees as presented in [8]. Each valid leaf node in the resulting one-sided decision trees corresponds to a risk feature, which is highly indicative of one class label over the other.

In the construction of traditional two-sided decision trees, a partition operation divides a set $R$ of records into two separate subsets, $R_l$ and $R_r$, both of which consist of mostly normal or abnormal records. The label purity of a partition is usually measured by the metric of *Gini* [53] as follows:

$$G(R) = \frac{|R_l|}{|R|} \times G(R_l) + \frac{|R_r|}{|R|} \times G(R_r),$$

in which $G(R_l)$ and $G(R_r)$ denote the *Gini* values of two subsets.

In contrast, the process of risk feature generation is one-sided. A partition only needs to extract a subset $R_i$ from $R$ such that most of the records in $R_i$ have the same label. The remaining records in the subset of $R' = R - R_i$ can instead have mixed labels. The partition operation is supposed to be iteratively applied on $R'$. The label purity of a one-sided partition is measured by the metric of one-sided *Gini* index as follows:

$$\hat{G}(R) = \min \left( \frac{\lambda}{|R_l|} + (1 - \lambda) \times G(R_l), \frac{\lambda}{|R_r|} + (1 - \lambda) \times G(R_r) \right),$$

in which $\lambda$ is the weight parameter to balance the influence of set size and label impurity. A large $\lambda$ means we prefer set size over label purity. Since the extracted subset needs to be highly discriminating, we suggest to set the value of $\lambda$ at low. A one-sided partition would generate two subsets with the minimum value of one-sided *Gini* index. By this optimization objective, the construction process ensures that each partition would produce a highly pure leaf node regardless of the purity of the remaining records.

**Complexity Analysis.** As shown in [8], with the size of training data denoted by $\bar{n}$, the number of risk metrics by $m$, and the pre-specified maximum depth of decision trees by $h$, the total time complexity of risk feature generation can be represented by $O(h \times (2m)^h \times \bar{n} \times \ln(\bar{n}))$. It is worthy to point out that for NID, the number of risk metrics (i.e., $m$) is usually limited (e.g., dozens); to ensure interpretability, the maximum depth of decision trees (i.e., $h$) is also usually set to a small value (e.g., $h \leq 4$ in our implementation). Therefore, the algorithm of risk feature generation can be executed efficiently in practice.

4.2. Risk Model Training

As in previous work [8], we employ the metric of Value at Risk (VaR) to quantify misclassification risk. Given an activity record, $r_i$, we denote its intrusion probability by $p_i$ and the inverse of its cumulative distribution function by $F_i^{-1}(\cdot)$, i.e., the quantile function. In other words, if a deep model labels $r_i$ as *Normal*, the probability of $r_i$ being mislabeled is equal to $p_i$. Accordingly, given the confidence level of $\theta$, the VaR of $r_i$ is measured by the maximum value of $z = p_i$ after excluding the $1 - \theta$ worse cases where $p_i$ is from $F_i^{-1}(\theta)$ to $+\infty$. Formally, the VaR value of a record with the label of *Normal* can be represented by

$$\text{VaR}_\theta(r_i) = F_i^{-1}(\theta; \mu, \sigma_i^2).$$

Similarly, if \( r_i \) is labeled as Abnormal, its VaR value can be represented by

\[
\text{VaR}_d(r_i) = 1 - F^{-1}_r(1 - \theta; \mu_i, \sigma_i^2). \tag{14}
\]

The risk model is supposed to be trained on labeled validation data by a learning-to-rank objective \cite{54}. In general, it aims to rank a mislabeled record before a correctly labeled record in terms of mislabeling risk. The learnable parameters include the weights of risk features, \( w_i \), and the variances of their distribution, \( \sigma_i^2 \). We employ the logistic function to map a risk value to a posterior probability by

\[
p_{ij} = \frac{e^{v_i - v_j}}{1 + e^{v_i - v_j}}, \tag{15}
\]

and define its target probability by

\[
\tilde{p}_{ij} = 0.5 \times (1 + \hat{g}_i - \hat{g}_j), \tag{16}
\]

where \( \hat{g}_j \in \{0, 1\} \) denotes the risk label of a record. If a record \( r_i \) is mislabeled, \( \hat{g}_i = 1 \); otherwise, \( \hat{g}_j = 0 \). Based on the definitions of posterior probability and its target probability, we define the loss function of risk model training by

\[
L(R^v) = \sum_{r_i, r_j \in R^v} -\tilde{p}_{ij} \cdot \log(p_{ij}) - (1 - \tilde{p}_{ij}) \cdot \log(1 - p_{ij}). \tag{17}
\]

In the implementation, we have employed the Adam optimizer \cite{55} to find the optimal parameters by gradient descent.

5. Adaptive Deep Learning for NID

**Procedure 1** Risk-based Adaptive Training for NID

**Input**: A NID task \( R \) consisting of \( R^s \), \( R^r \) and \( R^t \), and a NID model, \( g(w) \);

**Output**: A learned classifier \( g(w^*) \).

1: \( w_0 \leftarrow \) Initialize \( w \) with random values;
2: \textbf{for} \( k = 0 \) to \( m - 1 \) \textbf{do}
3: \hspace{1em} \( w_{k+1} \leftarrow w_k - \alpha \times \nabla_{w_k} L_{\text{train}}(w_k) \);
4: \textbf{end for}
5: Select the best model, \( g(w_*) \), based on \( R^v \);
6: \( w_0 \leftarrow w_* \);
7: \textbf{for} \( k = m \) to \( m + n - 1 \) \textbf{do}
8: \hspace{1em} Update the risk model based on \( R^v \) and \( g(w_k) \);
9: \hspace{1em} \( w_{k+1} \leftarrow w_k - \alpha \times \nabla_{w_k} L_{\text{risk}}^{tik}(w_k) \);
10: \textbf{end for}
11: Select the last trained model, \( g(w_*) \leftarrow g(w_{k+1}) \).
12: \textbf{return} \( g(w_*) \).

As shown in Figure 3, risk-based adaptive training for NID consists of two phases: the traditional training phase and the following adaptive training phase. In the first phase, it fine-tunes a deep model based on labeled training data in the traditional way. In the second phase, it iteratively executes: 1) using LearnRisk to learn a risk model based on the current deep model; 2) fine-tuning the current deep model by leveraging the learned risk model to minimize its misprediction risk on a target workload.

In the adaptive training phase, we employ the loss function of

\[
L_{\text{risk}}^{tik}(w) = \frac{1}{n_R} \sum_{i=1}^{n_R} [-[1 - \text{VaR}^r(r_i)] \times \ln(g(x_i^t; w)) - [1 - \text{VaR}^r(r_i)] \times \ln(1 - g(x_i^t; w))], \tag{18}
\]
in which \( n_{Rt} \) denotes the total number of records in the target workload of \( R \), \( g(x^t_t; w) \) denotes the label probability of a record in \( R \) as predicted by the current model, and \( \text{VaR}^+ (r_t) \) (resp. \( \text{VaR}^- (r_t) \)) denotes its estimated misprediction risk if it is labeled as Abnormal (resp. Normal).

We have sketched the process of risk-based adaptive training in Procedure[1]. In the first phase, the model is pre-trained based on labeled training data, and then the model with the best performance on validation data is selected. In the second phase, the parameters of the pre-trained model is iteratively fine-tuned by minimizing the loss of \( L_{\text{risk}}^{\text{test}} (w) \). In the implementation, as usual we update model parameters by gradient descent as follows:

\[
w_{k+1} = w_k - \alpha \times \nabla w_k L_{\text{risk}}^{\text{test}} (w_k),
\]

in which \( \alpha \) denotes the learning rate.

It can be observed that the adaptive training approach leverages the risk estimations of LearnRisk to fine-tune a classifier by minimizing its misprediction risk on a target workload. However, it does not use the ground-truth label of any target record. In the scenario of NID, it is reasonable to suppose that unlabeled target activity records are incrementally available for classifier fine-tuning. Therefore, the adaptive training approach is applicable to the task of NID. Our previous work on adaptive deep learning for the task of Entity Resolution (ER) has also provided theoretical explanation for its efficacy [44]. Specifically, it has been theoretically established that under the assumption of Identicalness of Risk Feature Distributions, risk-based fine-tuning has a fairly good chance to correctly flip the label of a false negative or positive to its ground-truth label. Those theoretical analytic results are similarly applicable to the task of NID. The detailed theoretical analysis is omitted here; but they can be found at [44].

6. Empirical Evaluation

![Figure 7: Comparative evaluation: Risk Analysis](image)
This section empirically evaluates the performance of the proposed solutions by a comparative study. It is organized as follows: Subsection 6.1 describes the experimental setup. Subsection 6.2 evaluates the proposed solution of risk analysis. Subsection 6.3 evaluates the proposed solution of risk-based adaptive training.

6.1. Experimental Setup

Our experiments have been conducted on three benchmark datasets: KDD99\(^2\) [27] and UNSW-NB15\(^3\) [28]. KDD99 is a classic NID dataset containing 22 types of attacks and 41 traffic features. CICIDS2017 is a more recent dataset, which contains 14 types of attacks and 78 traffic features. Compared with KDD99 and CICIDS2017, UNSW-NB15 contains richer types of attacks, i.e., 9 different types of contemporary attacks including Fuzzers, Reconnaissance, Shellcode, Analysis, Backdoors, DoS, Exploits, Generic and Worms. UNSW-NB15 contains 42 traffic features, which can be grouped into Flow features, Basic features, Content features, Time features and Additional generated features.

In order to simulate the real network scenario where there are a limited number of labeled activities but a lot of target activities to be labeled, we divide each dataset such that its test data subset has large size while its training and validation data subsets have relatively smaller sizes. On KDD99, the training data has 20,480 records, the validation data has 10,480 records while the test data has 40,960 records. On CICIDS2017, the training data has 179,200 records, the validation data has 10,480 records while the test data has 40,960 records. On UNSW-NB15, the training data has 40,960 records, the validation data has 26,624 records while the test data has 81,920 records.

For the evaluation of risk analysis, we compare the proposed solution, denoted by LearnRisk, with the following alternatives:

- **Baseline** [27]. It is a baseline approach that simply measures risk by the ambiguity of classifier output. In the case of NID, the network records with more ambiguous classifier outputs (i.e., close to 0.5) are supposed to be at higher risk of being mislabeled than those with the more extreme outputs (i.e., close to 0 or 1).
- **Uncertainty** [28]. It is a bootstrap-based approach originally proposed for active learning. It first generates different training sets by bootstrap resampling to train multiple classifiers, then estimates a record’s label probability \(p\) based on the classifier outputs, and finally uses the metric of \(p \times (1 - p)\) to measure risk.
- **TrustScore** [30]. It is a clustering-based approach. TrustScore first uses labeled training data to construct separate clusters to represent each class. For a target record \(r\), let \(\rho_Y\) represent the distance of \(r\) to the cluster with the same label, and \(\rho_N\) represent the distance of \(r\) to the nearest cluster with a different label. Then, the TrustScore of \(r\) is measured by \(\frac{\rho_Y}{\rho_N}\).

For Uncertainty, we train 10 deep learning models with different random seeds for each dataset. For TrustScore, we generate the input features of 12-dimension vectors based on deep models. For LearnRisk, we set the VaR confidence of the risk model at 0.9 and the number of epochs for parameter optimization at 20.

For the evaluation of risk-based adaptive training, we compare the adaptive approach, denoted by Risk, with a baseline deep model, denoted by Traditional. To simulate different sufficiency levels of training data, we randomly select 25\%, 50\%, 75\% and 100\% of the split set of training data on UNSW-NB15 and KDD99, and select 10,240, 20,480, 30,720, 40,960 and 179,200 records from the training data on CICIDS2017 while fixing the sets of validation and test data. In order to overcome the randomness caused by model initialization, we conduct 5 training sessions with different random seeds and report the mean and standard deviation of \(F_1\) score on test data.

We have implemented the Time-related Network Intrusion Detection Model [7], the state-of-the-art deep learning solution for NID, as the baseline classifier, and built the proposed solutions of risk analysis and adaptive training upon it. For Traditional, we run 20 iterations in each training session on UNSW-NB15, 100 iterations on KDD99 and 200 iterations on CICIDS2017. We have observed that after that, the performance of the deep classifier flattens out and only fluctuates very marginally. For Risk, the adaptive training phase consists of 40 iterations on UNSW-NB15, 100 iterations on KDD99 and 100 iterations on CICIDS2017. Similarly, the performance of Risk flattens out after that.

In the adaptive training phase, our implementation re-trains the risk model based on the output of the current deep classifier, and re-assesses labeling risk in each iteration. We have set the learning rates of the Traditional and the Risk between 0.001 and 0.02. For comparative evaluation, all the learning rates are set to be 0.01. Our sensitivity evaluation results presented in Subsection 6.3 show that the proposed approach consistently outperforms the baseline deep classifier, and its performance is robust w.r.t the learning rate.

6.2. Evaluation: Risk Analysis

The detailed evaluation results have been presented in Figure 7. It can be observed that LearnRisk consistently outperforms its alternatives by considerably margins on all the test datasets with various sufficiency levels of training data. In particular, in terms of AUROC, LearnRisk outperforms Baseline by more than 25% on UNSW-NB15. The major reason for the unsatisfactory performance of Baseline is that deep models are usually prone to over-confident predictions. For instance, as shown in Figure 8, most of the classifier outputs on UNSW-NB15 are extreme values (i.e., very close to 0 or 1), some of which are unfortunately misleading. It can also be observed that Uncertainty achieves highly regular ROC curves, but its performance is not even competitive with Baseline. TrustScore performs better than Baseline on UNSW-NB15, but worse than Baseline on CICIDS2017 and KDD99. Our experiments have clearly demonstrated the efficacy of LearnRisk.

Sensitivity evaluation w.r.t size of validation data. Since LearnRisk depends on labeled validation data, we evaluate its performance sensitivity w.r.t the size of validation data. In real scenarios, validation data are necessary for hyperparameter tuning and model selection to ensure that a trained deep model can generalize well. However, due to labeling cost, it is usually desirable to keep the size of validation data small. For sensitivity evaluation, on all three datasets, we fix the size of training data and test data while reducing the size of validation data to 1024 records, which is the smallest batch size of the deep classifier for NID. The detailed evaluation results have been shown in Figure 9. It can be observed that the performance of LearnRisk only fluctuates marginally. This observation means that LearnRisk is very robust w.r.t the size of validation data.

Sensitivity evaluation w.r.t the parameter of $\lambda$. Since LearnRisk uses the parameter of $\lambda$ as shown in Eq. 12 to balance the influence of set size and label impurity, we also evaluate its performance sensitivity w.r.t the value of $\lambda$. We suggest that the value of $\lambda$ is set to be less than 0.5 to ensure that label impurity weighs higher than set size. The evaluation results of LearnRisk on CICIDS2017 with $\lambda = 0.1$, 0.2 or 0.3 have been presented in Table 4. The results on other datasets are similar, thus omitted here. It can be observed that the performance of LearnRisk is robust w.r.t the parameter of $\lambda$ provided that its value is set within a reasonable range (e.g., between 0.1 and 0.3).
6.3. Evaluation: Risk-based Adaptive Training

We have evaluated the performance of risk-based adaptive training in both offline and online settings. In the offline setting, the whole target workload is supposed to be available for classifier training. However, in real scenarios, NID systems usually need to perform intrusion detection in a real-time manner. Therefore, we have also conducted the evaluation in the online setting, where a target workload is supposed to be streamed in, or only incrementally available for model training. To this end, inspired by the sliding window algorithm, we simulate network streaming traffic by splitting a target workload into fixed-sized windows (e.g., 2048 records). A deep classifier is iteratively fine-tuned based on sliding windows. In other words, in each iteration, we only leverage the 2048 records within the current window for adaptive training.

**Evaluation of adaptive deep learning: the offline setting.** The detailed evaluation results have been presented in Figure 10 where the mean and standard deviation of F1 are reported. It can be observed that Risk consistently outperforms Traditional on all the three datasets with different sufficiency levels of training data. In particular, the performance margins between Risk and Traditional tend to become larger when less training data are available. For example, with only 10,240 training records on CICIDS2017, Risk beats Traditional by around 17% (i.e., 87.3% vs. 70.2%). However, with 17,920 training records, their performance difference shrinks to only 2% (i.e., 85.9% vs. 83.1%). However, this observation should not be surprising because provided with insufficient training data, deep models usually underperform; risk analysis can thus more easily leverage the characteristics of target data for improved performance.

**Evaluation of adaptive deep learning: the online setting.** To simulate the streaming traffic, we split a test dataset into fixed-size (e.g., 2048) disjoint subsets. Then, instead of using all the records in the test dataset at once, we only leverage one subset for risk-based adaptive training at a time. A newly fine-tuned deep model for NID is iteratively employed to predict the next subset. The detailed evaluation results have been presented in Figure 10 in which StreamRisk denotes the streaming approach of risk-based adaptive training. It can be observed that even though the performance of StreamRisk fluctuates slightly due to distribution shifting, it consistently performs better than Traditional and even beats the offline Risk in some cases. For instance, on CICIDS2017, with the split set of training data at 20,480, the performance of StreamRisk outperforms Risk by around 9% (i.e., 88.8% vs. 79.4%).

<table>
<thead>
<tr>
<th></th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>CICIDS2017(10240)</td>
<td>0.8752</td>
<td>0.8962</td>
<td>0.8830</td>
</tr>
<tr>
<td>CICIDS2017(20480)</td>
<td>0.7855</td>
<td>0.7722</td>
<td>0.8092</td>
</tr>
<tr>
<td>CICIDS2017(30720)</td>
<td>0.8720</td>
<td>0.8795</td>
<td>0.8720</td>
</tr>
<tr>
<td>CICIDS2017(40960)</td>
<td>0.8939</td>
<td>0.8875</td>
<td>0.9082</td>
</tr>
</tbody>
</table>
**Illustrative examples.** We illustrate the efficacy of adaptive training by the example of CICIDS2017 with 10240 labeled training data. Since the adaptive training phase consists of 100 iterations, we report the performance of the fine-tuned deep model after every 20 iterations. The detailed evaluation results have been presented in Figure 10. It can be observed that despite some fluctuations, overall speaking, adaptive training can continuously improve the performance of the deep model until its performance finally flattens out. This observation clearly indicates that risk-based fine-tuning can effectively improve classification accuracy. Note that similar observations can also be observed on other datasets.

Furthermore, we have provided with the statistics of label flips after the first 20 iteration on CICIDS2017 in Table 5. It can be observed that adaptive training can correctly flip many false positives and negatives while flipping a smaller number of true positives and negatives. The combined result are the performance improvements as shown in Figure 11. The major reason for mis-flipping some true positives and negatives is distribution misalignment of risk features between labeled training/validation data and unlabeled test data. These experimental results are consistent with the theoretical analysis presented in our previous work [44].

**Sensitivity evaluation w.r.t the parameter of \( \alpha \).** Since our proposed approach uses the learning rate of \( \alpha \) as shown in Eq. [19] to update model parameters in its adaptive training phase, we also evaluate its performance sensitivity w.r.t the value of \( \alpha \). We vary the learning rates of Traditional and Risk from 0.001 to 0.02. The detailed evaluation results on UNSW-NB15 (with 25% training data) and CICIDS2017 (with 10240 training data) have been presented in Figure 12. The results on other settings are similar, thus omitted here. It can be observed that the performance of both Traditional and Risk initially improves with the increasing value of \( \alpha \) but flattens out or even becomes a litter worse.
Table 5: The statistics of label flips after the first 20 iterations of risk-based fine-tuning on CICIDS2017.

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>Flipped</th>
</tr>
</thead>
<tbody>
<tr>
<td>False Positives</td>
<td>1605</td>
<td>653</td>
</tr>
<tr>
<td>True Positives</td>
<td>5193</td>
<td>265</td>
</tr>
<tr>
<td>False Negatives</td>
<td>2878</td>
<td>1106</td>
</tr>
<tr>
<td>True Negatives</td>
<td>31284</td>
<td>841</td>
</tr>
</tbody>
</table>

after 0.01. Therefore, in real implementations, we suggest to set the learning rate at 0.01. It is noteworthy that even though the performance of the proposed approach may fluctuate with the learning rate, it consistently outperforms the baseline deep classifier. Our experimental results clearly demonstrate the efficacy of the proposed approach.

![Graph](a) UNSW-NB15 (with 25% training data)  (b) CICIDS2017 (with 10240 training data)

Figure 12: Performance sensitivity evaluation w.r.t the learning rate ($\alpha$)

7. Conclusion

In this paper, we have proposed a solution of interpretable risk analysis for NID and demonstrated how to leverage it for adaptive deep learning. Our extensive experiments have validated the efficacy of the proposed solutions. With respect to future work, it can be observed that even though risk features in the form of rules are interpretable and effective, their efficacy is to some extent limited by the expressive power of rules. Therefore, it is interesting to explore risk features for NID beyond rules. On the other hand, in principle, the framework of LearnRisk and the risk-based adaptive training approach are applicable to other classification tasks; the technical solutions however need to be further investigated.

Acknowledgements

The work is supported by the National Key Research and Development Program of China under Grant No. 2018YFB1003400, the National Natural Science Foundation of China under Grant No. 62172335, No. 61972317, No. 61732014, and No. 61672432, the Fundamental Research Funds for the Central Universities of China under Grant No.3102019DX1004.
References
