Neurocomputing 493 (2022) 46–58

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Adaptive deep learning for network intrusion detection by risk analysis

Lijun Zhang [a,b], Xingyu Lu [a,b], Zhaoqiang Chen [a,b], Tianwei Liu [a,b], Qun Chen [a,b,*], Zhanhuai Li [a,b]

[a] School of Computer Science, Northwestern Polytechnical University, Xi'an, China
[b] Key Laboratory of Big Data Storage and Management, Northwestern Polytechnical University, Ministry of Industry and Information Technology, Xi'an, China

ARTICLE INFO

Article history:
Received 1 July 2021
Revised 28 March 2022
Accepted 9 April 2022
Available online 12 April 2022
Communicated by Zidong Wang

Keyword:
Network intrusion detection
Risk analysis
Adaptive deep learning

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 deep models may still mislabel many activities in real networks. 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 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.

1. Introduction

With increasing connectedness, network intrusion has become a critical security concern for modern information systems. Various network intrusion incidents have been reported in recent years and they have brought about considerable financial and societal damage [1,2]. The purpose of Network Intrusion Detection (NID) is to identify malicious activities in network traffic. As shown in Fig. 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 r_2 uses the udp protocol and the dns service with fixed-size source-to-destination bytes. Since r_2 matches the characteristics of a generic attack, it is considered to be a network intrusion. However, the record of r_1 represents a normal activity. NID is usually considered as a binary classification problem tasked with labeling each activity as *Normal* or *Abnormal*. Even though NID can be performed based on expert rules and data mining [3,4], its state-of-the-art performance has been achieved by deep learning [5–7].

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 networks. It is noteworthy that 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. Toward this end, we propose a novel solution of risk analysis for NID in this paper. Built upon on 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 Fig. 2, it consists of three steps: risk feature generation, risk model construction and risk model training. To ensure that risk features are interpretable and discriminative, we construct various risk metrics from differ-

* Corresponding author at: School of Computer Science, Northwestern Polytechnical University, Xi'an, China.
E-mail address: chenbenben@nwpu.edu.cn (Q. Chen).

https://doi.org/10.1016/j.neucom.2022.04.061
0925-2312/

ID	dur	proto	service	state	sbytes	dbytes	.....
r_1	1.681642	tcp	ftp	FIN	628	770	.....
r_2	0.000003	udp	dns	INT	114	0	.....
.....							

Fig. 1. A NID running example.

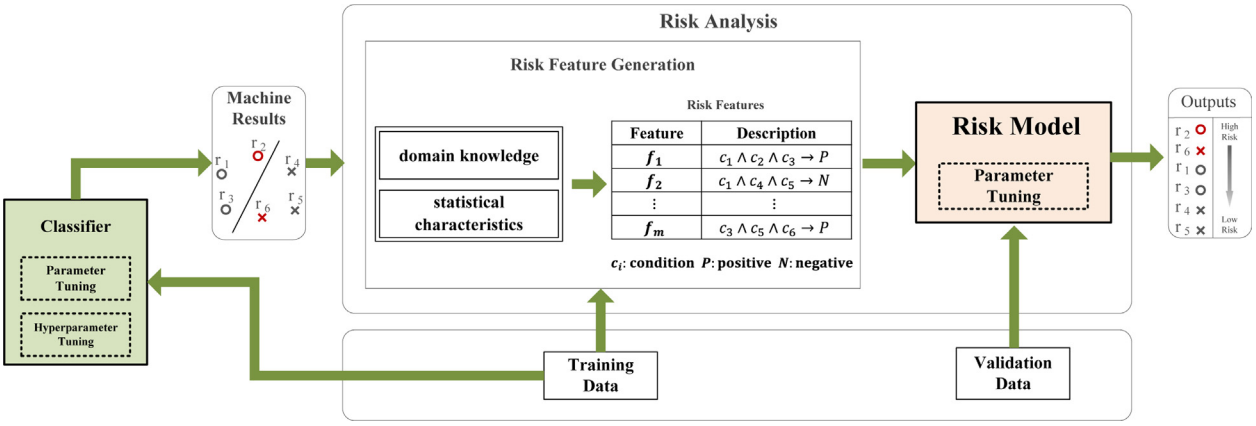


Fig. 2. The LearnRisk framework for NID.

ent 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 [10–12], 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 machine 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 risk analysis. As shown in Fig. 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 their 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 extensive experiments have shown that the proposed solution of risk analysis can identify the mislabeled activities 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.

2. Related Work

We review related work from the mutually orthogonal perspectives of network intrusion detection, risk analysis and model training.

Network Intrusion Detection. Extensively studied in the literature [13–15], NID has been a research hotspot. 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 [20–22] has 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 been empirically shown that the combination of a stacked sparse autoencoder and a variant of RNN could achieve promising performance for NID [7]. However, the efficacy of these deep models depends on large quantities of labeled training data, which may not be sufficiently available in real scenarios.

Risk Analysis. It has been well recognized that many classification tasks, including NID, are so challenging in real scenarios that even a well trained deep model is prone to prediction errors. Therefore, in recent years, there has been a growing interest in the study of risk analysis [27–32,8], alternatively called *confidence ranking* or *trust scoring* in the literature, which aimed to analyze misprediction risk of a machine classifier. Targeting image recognition, the authors of [27] presented a simple baseline that employed the probabilities from softmax distributions for risk measurement. The authors of [30] instead proposed to measure prediction risk by comparing the distances of an instance to the cluster with the same

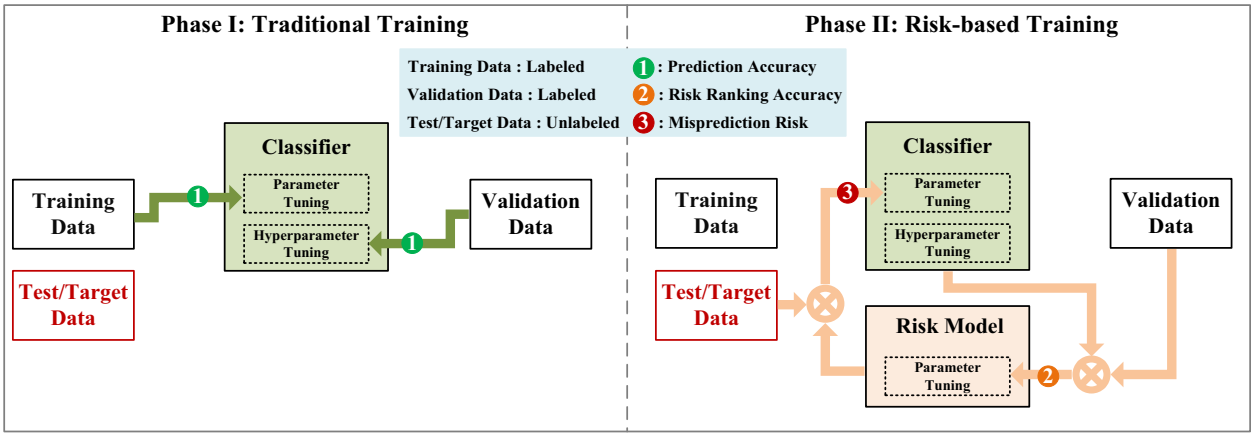


Fig. 3. Risk-based adaptive training for NID.

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 was to leverage validation data for model selection (e.g., cross validation [33]). An alternative approach was by regularization [34,35], which could 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 transfer learning [36–38] and adaptive representation learning [39–43]. 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 the domain-invariant features shared among different 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 transfer learning and adaptive representation learning, it could 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]: 1) unlike the traditional labeling functions, the purpose of LearnRisk is to estimate the misclassification risk of an instance as predicted by a machine classifier; 2) more importantly, the ensemble approach trains multiple models and tunes them based on training data while 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 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:

$$\mathbf{F1} = \frac{2 \times precision \times recall}{precision + recall}. \tag{1}$$

As usual, we suppose that an NID task of R consists of a set of labeled training data, $R^s = \{(x_i^s, y_i^s)|i\}$, where each (x_i^s, y_i^s) denotes a training instance with its feature vector of x_i^s and ground-truth label of y_i^s, a set of labeled validation data, $R^v = \{(x_i^v, y_i^v)|i\}$, and a set of unlabeled target data, $R^t = \{(x_i^t, ?)|i\}$. Note that the validation data, R^v, are provided as a proxy workload of R^t. In this paper, we consider both offline and online settings. In the offline setting, the entire target workload of R^t is supposed to be available for classifier training, while in the online setting, the workload of R^t is only provided incrementally. Given a task of R consisting of R^s, R^v and R^t, NID aims to learn an optimal classifier, $\mathscr{M}$, such that the performance of $\mathscr{M}$ on R^t in terms of **F1**, or $\mathbf{F1}(\mathscr{M}, R^t)$, 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. 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 thresholds. 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 and the number of false negatives by n_-^F. Then, we measure the true positive rate, *TPR*, by $\frac{n_+^T}{n_+^T + n_-^F}$, and the false positive rate, *FPR*, by $\frac{n_+^F}{n_+^F + n_-^T}$. 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

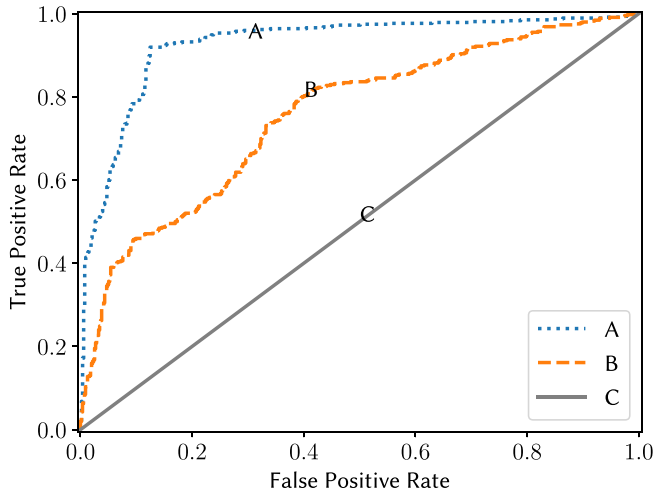


Fig. 4. The examples of ROC curve: C denotes a trivial baseline model, and A is better than B.

higher score to a randomly chosen positive than a randomly chosen negative. Therefore, a model with a higher AUROC achieves better performance. The illustrative examples of ROC have been shown in Fig. 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^s, R^v and R^t, and a classifier of $\mathscr{M}$ has labeled each activity in R^t. The task of risk analysis is to quantitatively measure the mislabeling risk of each activity 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 spans over a large subpopulation of the workload. As opposed to the traditional labeling functions, which classify activities 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:

$$r_i[\text{sttl}] > 61.0 \wedge r_i[\text{dpkts}] > 135.0 \rightarrow \text{abnormal}(r_i), \tag{2}$$

where r_i denotes an activity, 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 the longer time and more packets a network activity takes, the more likely it is an intrusion. With this knowledge, an activity predicted as *Normal*, whose *sttl* value is however greater than 61.0 and *dpkts* value 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 with human-friendly explanations. Towards this end, LearnRisk, drawing inspiration from investment theory, models the label probability distribution (portfolio reward) of an activity as the aggregation of the distributions of its compositional features (stocks rewards).

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

$$\mu_i = z_i(w \circ \mu_{\mathbf{F}}), \tag{3}$$

and

$$\sigma_i^2 = z_i(w^2 \circ \sigma^2_{\mathbf{F}}). \tag{4}$$

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 θ, the metric of VaR represents the maximum loss after excluding the worst outcomes whose combined probability is at most 1-θ.

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 (σ_i^2) of risk features as well as their weights (w_i). As for their expectations (μ_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 misclassification risk on a target workload labeled by a classifier.

4. Risk Analysis for NID

In this section, we first present how to generate risk features, and then describe how to train a risk model.

4.1. Risk Feature Generation

As shown in Fig. 5, we construct basic risk metrics for NID based on both intrusion domain knowledge and statistical abnormalcy.

4.1.1. Risk Metrics of Domain Knowledge

It has been widely recognized that domain knowledge can improve a NID classifier's generalizability to unknown attacks as well as its interpretability. In this work, we leverage the classical 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.

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, $\{t_1, t_2, \ldots, t_n \in T\}$.

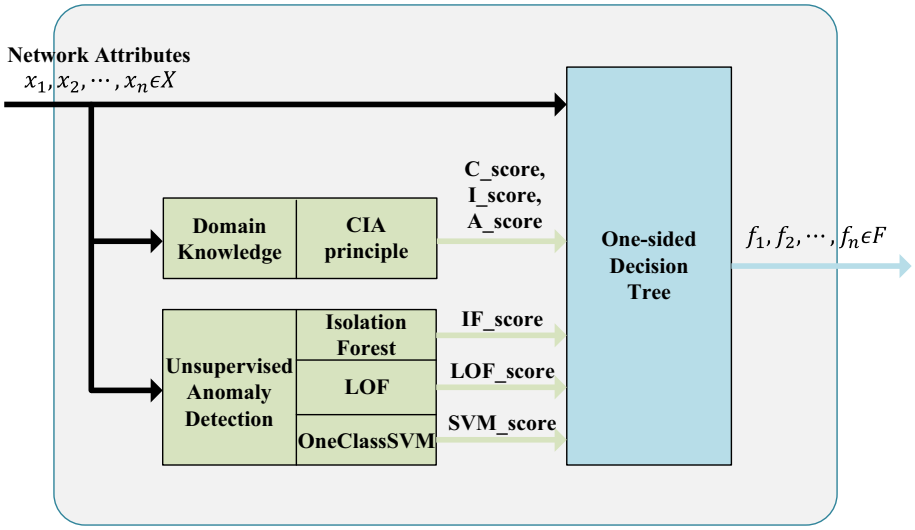


Fig. 5. Risk feature generation for NID.

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 can destroy data and harm confidentiality, integrity, and availability, it is related to all the three properties of CIA. see Table 1.

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. Some examples of attributes and their estimated correlation coefficients have been shown in Table 3.

Given an activity, r, we represent its CIA attribute vectors by T_C, T_I and T_A, which correspond to the three properties of CIA respectively. Note that the CIA attribute vectors can be simply computed based on the mapping between attributes and CIA as shown in Table 2. Then, we generate the three risk metrics of r, *C_score(r)*, *I_score(r)*, and *A_score(r)*, by aggregating its attribute vectors of CIA and the correlation coefficient vectors V_i as follows:

$$\text{C\_score}(\text{r}) = T_C \cdot V_C^{\text{T}}, \tag{5}$$

$$\text{I\_score}(\text{r}) = T_I \cdot V_I^{\text{T}}, \tag{6}$$

$$\text{A\_score}(\text{r}) = T_A \cdot V_A^{\text{T}}. \tag{7}$$

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 $\bar{n}$, the complexity of XGBoost can be represented by $\mathbf{O}(n_T \times h \times \bar{n} \times ln(\bar{n}))$ [51]. As a result, supposing that m denotes the total number of types of network attacks, we can represent

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

Attack	Related CIA components
Fuzzers	A
Analysis	C,I,A
Backdoors	C
DOS	A
Exploits	A
Generic	C
Reconnaissance	C
Shellcode	C,I,A
Worms	C,I,A

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

Attribute	Related attacks: the top-3 attributes	Related CIA component
sbytes	Analysis, Backdoors, DOS, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode, Worms	C,I,A
sload	Analysis, Backdoors	C,I,A
ct_srv_src	Analysis	C,I,A
ct_dst_sport_ltm	Backdoors	C
smean	DOS, Generic, Reconnaissance, Shellcode, Worms	C,I,A
ct_srv_dst	DOS, Exploits	A
dbytes	Exploits	A
ct_dst_src_ltm	Fuzzers	A
sttl	Fuzzers, Worms	C,I,A
dmean	Generic, Reconnaissance, Shellcode	C,I,A

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

Attribute	Correlation coefficient	Influence on label status
sbytes	−0.34	negative
sload	0.30	positive
ct_srv_src	0.24	positive
ct_dst_sport_ltm	0.46	positive
smean	−0.06	negative
ct_srv_dst	0.24	positive
dbytes	−0.47	negative
ct_dst_src_ltm	0.29	positive
sttl	0.66	positive
dmean	−0.52	negative

the complexity of attribute mapping by $\mathbf{O}(m \times n_T \times h \times \bar{n} \times ln(\bar{n}))$. For metrics construction, the complexity of Pearson correlation calculation can be represented by $\mathbf{O}(\bar{n})$. 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 $\mathbf{O}(n_A \times (n + \bar{n}))$. Therefore, without loss of generality, the time complexity of constructing risk metrics of CIA can be represented by $\mathbf{O}(m \times n_T \times h \times \bar{n} \times ln(\bar{n}) + n_A \times (n + \bar{n}))$.

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.

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 $k\text{-}distance(r_i)$. Accordingly, the reachability distance from r_i to another point r_j, denoted by $reach\text{-}dist(r_i, r_j)$, is defined by the maximum of $k\text{-}distance(r_i)$ 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:

$$LOF(r_i) = \frac{\sum_{r_j \in N_{r_i}} \frac{LRD(r_j)}{LRD(r_i)}}{\mid N_{r_i} \mid}, \tag{8}$$

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.

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. 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 examples from CICIDS2017 as shown in Fig. 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.

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

$$IF(r_i) = 2^{-\frac{E(IL(r_i))}{IL(R)}}, \tag{9}$$

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 definition, if a point's IF score is close to 1, it is very 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 approach for anomaly detection. For risk measurement, we leverage OneClassSVM [12], which first 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, $DF(r_i)$, is defined by

$$DF(r_i) = sgn((\omega \cdot \Phi(r_i)) - \rho), \tag{10}$$

where ω and ρ denote model parameters and $\Phi(r_i)$ denotes 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 quantify risk by OneClassSVM output probability.

Complexity Analysis. The time complexities of generating statistical risk metrics are 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 $\mathbf{O}(n \times log(n))$, and the second one has the time complexity of $\mathbf{O}(n)$. Hence, the total time complexity of LOF can be represented by $\mathbf{O}(n \times ln(n))$.
- Complexity of IF. The process of IF consists of two steps: the first one builds a forest of n_T trees, in each of which ψ 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 $\mathbf{O}(n_T \times \psi \times ln(\psi))$ and $\mathbf{O}(n_T \times n \times ln(\psi))$ respectively. Hence, the total time complexity of IF can be represented by as $\mathbf{O}(n_T \times (n + \psi) \times ln(\psi))$. Note that in the default setting of our scikit-learn implementation [1], the parameters of n_T and ψ 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.

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

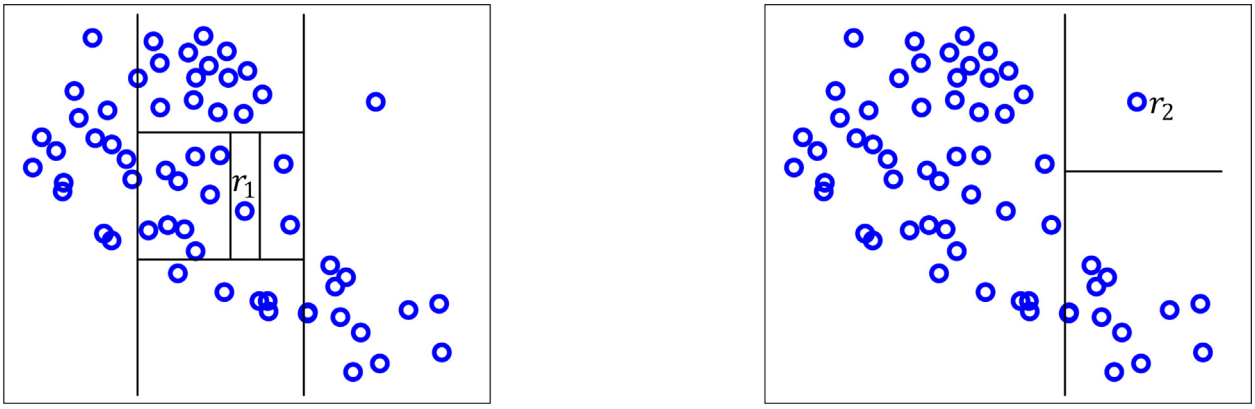
$$G(R) = \frac{\mid R_l \mid}{\mid R \mid} \times G(R_l) + \frac{\mid R_r \mid}{\mid R \mid} \times G(R_r), \tag{11}$$

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 activities in R_i have the same label. The remaining activities 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:

$$\widehat{G}(R) = min\left(\frac{\lambda}{\mid R_l \mid} + (1 - \lambda) \times G(R_l), \frac{\lambda}{\mid R_r \mid} + (1 - \lambda) \times G(R_r)\right), \tag{12}$$

[1] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html



(a) Isolationg r_1 (b) Isolationg r_2

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

in which λ is the weight parameter to balance the influence of set size and label impurity. A large λ means we prefer set size over label purity. Since the extracted subset needs to be highly discriminative, we suggest to set the value of λ 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 activities.

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 $\mathbf{O}\left(h \times (2m)^h \times \bar{n} \times ln(\bar{n})\right)$. 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., $h \leqslant 4$ 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, 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 θ, 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 an activity with the label of *Normal* can be represented by

$$VaR_\theta(r_i) = F_i^{-1}\left(\theta; \mu_i, \sigma_i^2\right). \tag{13}$$

Similarly, if r_i is labeled as *Abnormal*, its VaR value can be represented by

$$VaR_\theta(r_i) = 1 - F_i^{-1}\left(1-\theta; \mu_i, \sigma_i^2\right). \tag{14}$$

The risk model is supposed to be trained on labeled validation data by a learning-to-rank objective [54]. In general, it aims to rank a mislabeled activity before a correctly labeled one in terms of mislabeling risk. The learnable parameters include the weights of risk features, w_i, and the variances of their distribution, σ^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

$$\bar{p}_{ij} = 0.5 * (1 + \hat{g}_i - \hat{g}_j), \tag{16}$$

where $\hat{g}_j \in \{0, 1\}$ denotes the risk label of an activity. If an activity 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

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

In the implementation, we have employed the Adam optimizer [55] to find the optimal parameters by gradient descent.

5. Adaptive Deep Learning for NID

As shown in Fig. 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. Then, 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

$$\begin{aligned}\mathscr{L}_{test}^{risk}(w) = \frac{1}{n_{R^t}} \sum_{i=1}^{n_{R^t}} \{ &-[1 - VaR^+(r_i)] \times ln\left(g(x_i^t; w)\right) \\ &- [1 - VaR^-(r_i)] \times ln\left(1 - g(x_i^t; w)\right)\},\end{aligned} \tag{18}$$

in which n_{R^t} denotes the total number of activities in the target workload of R^t, $g(x_i^t; w)$ denotes the label probability of an activity in R^t as predicted by the current model, and $VaR^+(r_i)$ (resp. $VaR^-(r_i)$) 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 $\mathscr{L}_{test}^{risk}(w)$. In the implementation, as usual we update model parameters by gradient descent as follows:

$$w_{k+1} = w_k - \alpha \times \nabla_{w_k} \mathscr{L}^{risk}_{test}(w_k), \quad (19)$$

in which α denotes the learning rate.

Procedure 1: Risk-based Adaptive Training for NID

Input: A NID task R consisting of R^s, R^v and R^t, and a NID model, $g(w)$;
Output: A learned classifier $g(w_*)$.
1: $w_0 \leftarrow$ Initialize w with random values; **do**
2: **for** $k = 0$ to $m - 1$
3: $w_{k+1} \leftarrow w_k - \alpha \times \nabla_{w_k} \mathscr{L}_{train}(w_k)$;
4: **end for**
5: Select the best model, $g(w_*)$, based on R^v;
6: $w_m \leftarrow w_*$;
7: **for** $k = m$ to $m + n - 1$ **do**
8: Update the risk model based on R^v and $g(w_k)$;
9: $w_{k+1} \leftarrow w_k - \alpha \times \nabla_{w_k} \mathscr{L}^{risk}_{test}(w_k)$;
10: **end for**
11: Select the last trained model, $g(w_*) \leftarrow g(w_{k+1})$.
12: **return** $g(w_*)$

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 activity. In the scenario of NID, it is reasonable to suppose that unlabeled target activities 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 it can be found at [44].

6. Empirical Evaluation

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], CICIDS2017[3] and UNSW-NB15[4]. 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. It has 42 traffic features, which can be grouped into Flow features, Basic features, Content features, Time features and Additional generated features.

[2] available at http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[3] available at https://www.unb.ca/cic/datasets/ids-2017.html
[4] available at https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/

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 subset has large size while its training and validation subsets have relatively smaller sizes. On KDD99, training data have 20,480 activity records, validation data have 10,480 records while test data have 40,960 records. On CICIDS2017, training data have 179,200 records, validation data have 10,480 records while test data have 40,960 records. On UNSW-NB15, training data have 40,960 records, validation data have 26,624 records while test data have 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 activities 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]. Originally proposed for active learning, the bootstrap-based approach first generates different training sets by bootstrap resampling to train multiple classifiers, then estimates an activity's label probability p based on the classifier outputs, and finally uses the metric of $p \times (1 - p)$ to measure risk.
- **TrustScore** [30]. The clustering-based approach first uses labeled training data to construct separate clusters to represent each class. Given a target activity, r_i, let ρ_Y represent the distance of r_i to the cluster with the same label, and ρ_N represent the distance of r_i to the nearest cluster with a different label. Then, TrustScore measures the misclassification risk of r_i 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 **F1** score on test data.

We have implemented the Time-related Network Intrusion Detection Model [7], the state-of-the-art deep model 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

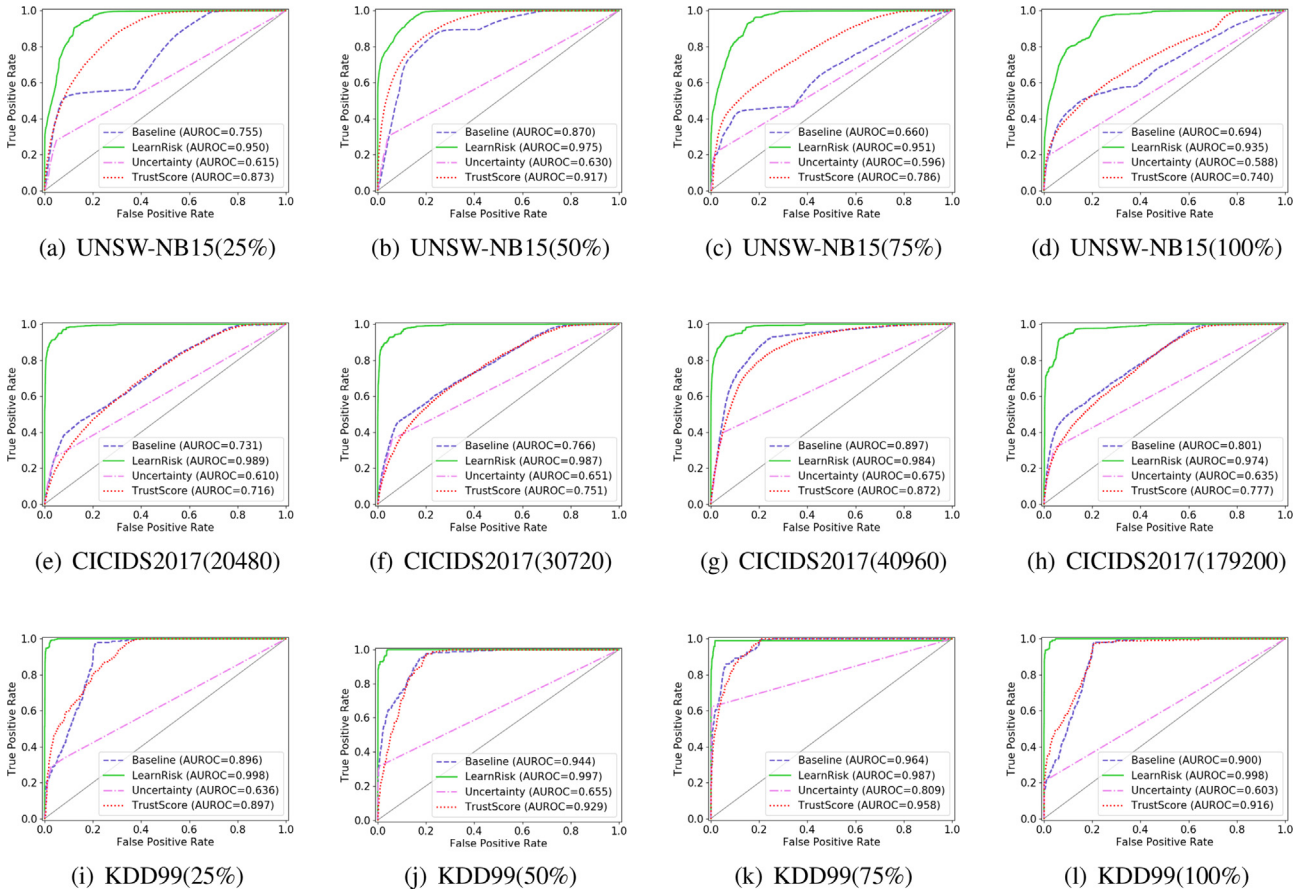


Fig. 7. Comparative evaluation: Risk Analysis.

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. The implementations as well as the benchmark datasets have been open sourced on our website [5].

6.2. Evaluation: Risk Analysis

The detailed evaluation results have been presented in Fig. 7. It can be observed that **LearnRisk** consistently outperforms its alternatives by considerable margins on all the test scenarios. In particular, in terms of AUROC, **LearnRisk** outperforms **Baseline** by more than 20% 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 Fig. 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 Fig. 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.

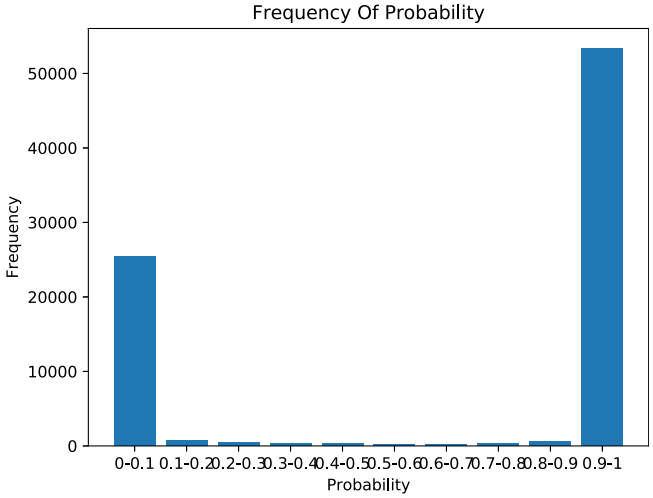
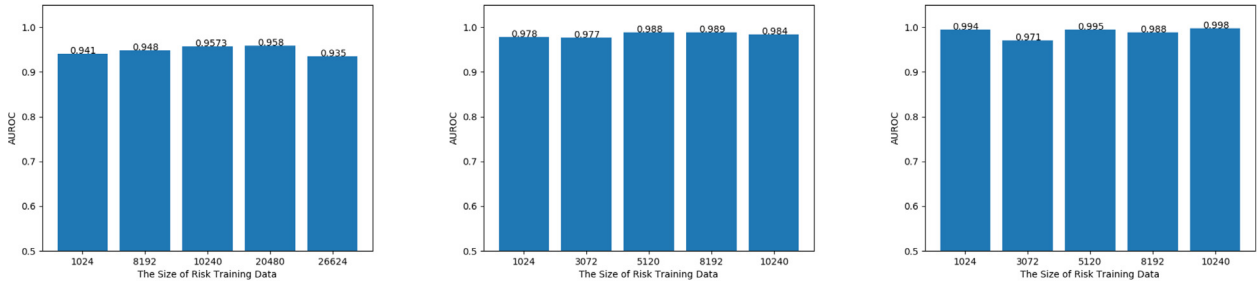


Fig. 8. The distribution of classifier outputs on UNSW-NB15.

Sensitivity evaluation w.r.t the parameter of λ. Since LearnRisk uses the parameter of λ as shown in Eq. 12 to balance the

[5] https://chenbenben.org/adaptive-training-NID.html



(a) UNSW-NB15 (b) CICIDS2017 (c) KDD99

Fig. 9. Performance sensitivity evaluation of LearnRisk w.r.t the size of validation data.

influence of set size and label impurity, we also evaluate its performance sensitivity w.r.t the value of λ. We suggest that the value of λ 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 λ 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

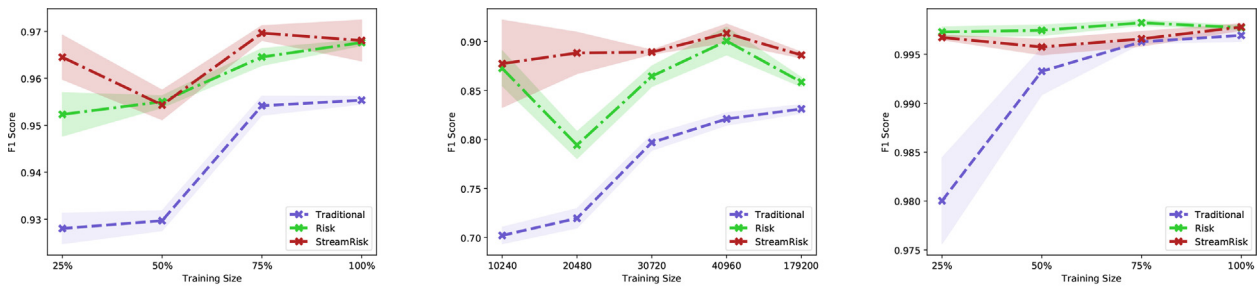
We have evaluated the performance of risk-based adaptive training in both offline and online settings. In the offline setting, the entire target workload is supposed to be available for classifier training. However, in real scenarios, NID systems usually need to perform intrusion detection in a real-time 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 many fixed-sized windows (e.g., 2048 activities). A deep classifier is iteratively fine-tuned based on sliding windows. In other words, in each iteration, we only leverage the 2048 activities within the current window for adaptive training.

Evaluation of adaptive deep learning: the offline setting. The detailed evaluation results have been presented in Fig. 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 activities on CICIDS2017, **Risk** beats **Traditional** by around 17% (i.e., 87.3% vs. 70.2%). However, with 17,9200 training activities, their performance difference narrows 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; thus, risk analysis can more easily leverage the characteristics of target data for improved performance.

Evaluation of adaptive deep learning: the online setting. To simulate streaming traffic, we split a test dataset into fixed-size (e.g., 2048) disjoint subsets. Then, instead of using all the activities 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 Fig. 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

Table 4
Performance sensitivity evaluation of LearnRisk w.r.t the parameter of λ

	0.1	0.2	0.3
CICIDS2017(10240)	0.8752	0.8962	0.8830
CICIDS2017(20480)	0.7855	0.7722	0.8092
CICIDS2017(30720)	0.8720	0.8795	0.8720
CICIDS2017(40960)	0.8939	0.8875	0.9082



(a) UNSW-NB15 (b) CICIDS2017 (c) KDD99

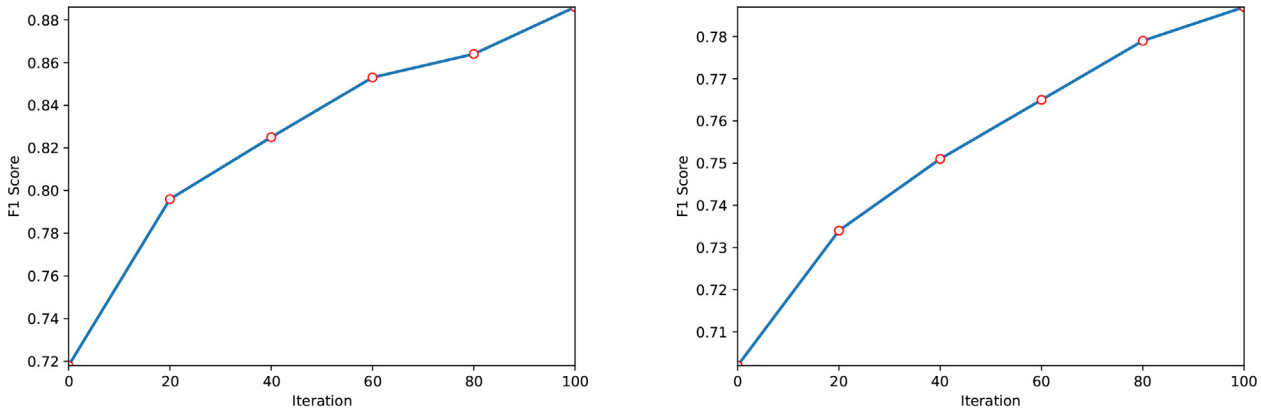
Fig. 10. Comparative evaluation: Risk-based Adaptive Training.

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%).

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 Fig. 11. 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 false negatives while flipping a smaller number of true positives and true negatives. The combined result are the performance improvements as shown in 11. The major reason for misflipping some true positives and true 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 α. Since our proposed approach uses the learning rate of α as shown in Eq. 19 to update model parameters in its adaptive training phase, we also

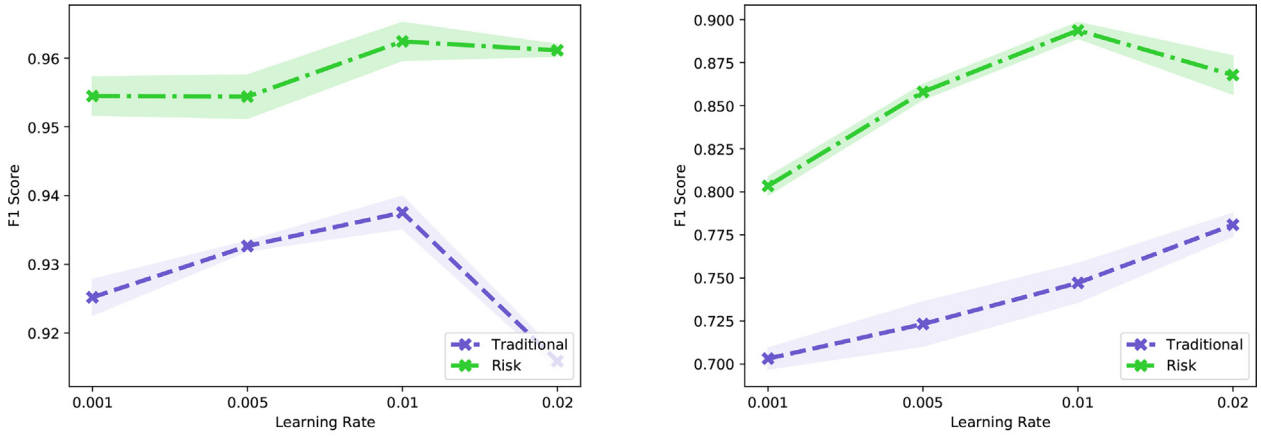


(a) The size of training data at 10240 (b) The size of training data at 20480

Fig. 11. The performance of **Risk** after every 20 iterations of risk-based fine-tuning on CICIDS2017.

Table 5
The statistics of label flips after the first 20 iterations of risk-based fine-tuning on CICIDS2017.

	#	# Flipped
False Positives	1605	653
True Positives	5193	265
False Negatives	2878	1106
True Negatives	31284	841



(a) UNSW-NB15 (with 25% training data) (b) CICIDS2017 (with 10240 training data)

Fig. 12. Performance sensitivity evaluation w.r.t the learning rate (α).

evaluate its performance sensitivity w.r.t the value of α. 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 Fig. 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 α but flattens out or even becomes a litter worse 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.

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. For 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, the framework of LearnRisk and the risk-based adaptive training approach are in principle applicable to other classification tasks; the technical solutions however need to be further investigated.

CRediT authorship contribution statement

Lijun Zhang: Conceptualization, Methodology, Validation, Writing - review & editing. **Xingyu Lu:** Methodology, Software, Investigation, Writing - original draft. **Zhaoqiang Chen:** Writing - original draft, Visualization. **Tianwei Liu:** Software. **Qun Chen:** Conceptualization, Writing - review & editing, Supervision. **Zhanhuai Li:** Resources.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

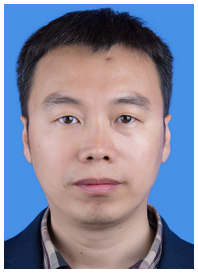
Acknowledgements

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.

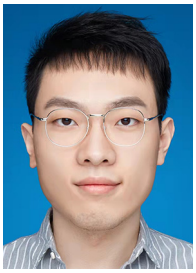
References

[1] Symantec internet security threat report, URL:http://www.symantec.com/, 2020. Accessed March 1, 2021.
[2] Verizon's data breach investigation report 2014, URL:http://www.verizonenterprise.com/DBIR/2014/, 2014. Accessed March 1, 2021.
[3] T. Abbes, A. Bouhoula, M. Rusinowitch, Efficient decision tree for protocol analysis in intrusion detection, International Journal of Security and Networks 5 (2010) 220–235.
[4] W. Zhi, Fault diagnosis for wireless sensor network based on genetic-support vector machine, in: Proceedings of 2011 International Conference on Computer Science and Network Technology, volume 4, IEEE, 2011, pp. 2691–2694.
[5] M.-J. Kang, J.-W. Kang, Intrusion detection system using deep neural network for in-vehicle network security, PloS one 11 (2016) e0155781.
[6] B. Roy, H. Cheung, A deep learning approach for intrusion detection in internet of things using bi-directional long short-term memory recurrent neural network, in: Proceedings of the 28th International Telecommunication Networks and Applications Conference (ITNAC), IEEE, 2018, pp. 1–6.
[7] Y. Lin, J. Wang, Y. Tu, L. Chen, Z. Dou, Time-related network intrusion detection model: A deep learning method, in: Proceedings of the 2019 Global Communications Conference (GLOBECOM), IEEE, 2019, pp. 1–6.
[8] Z. Chen, Q. Chen, B. Hou, Z. Li, G. Li, Towards interpretable and learnable risk analysis for entity resolution, in: Proceedings of the 2020 ACM International Conference on Management of Data (SIGMOD), ACM, 2020, pp. 1165–1180.
[9] M. Bishop, Introduction to Computer Security, Addison Wesley, 2004.
[10] M.M. Breunig, H.-P. Kriegel, R.T. Ng, J. Sander, LOF: identifying density-based local outliers, in: Proceedings of the ACM International Conference on Management of Data (SIGMOD), ACM, 2000, pp. 93–104.
[11] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation forest, in: Proceedings of the 8th International Conference on Data Mining, IEEE, 2008, pp. 413–422.
[12] B. Schölkopf, R.C. Williamson, A.J. Smola, J. Shawe-Taylor, J.C. Platt, Support vector method for novelty detection, Advances in Neural Information Processing Systems (2000) 582–588.
[13] M. Ahmed, A.N. Mahmood, J. Hu, A survey of network anomaly detection techniques, Journal of Network and Computer Applications 60 (2016) 19–31.
[14] R. Singh, H. Kumar, R. Singla, An intrusion detection system using network traffic profiling and online sequential extreme learning machine, Expert Systems with Applications 42 (2015) 8609–8624.
[15] G.X. Gu, C.-T. Chen, M.J. Buehler, De novo composite design based on machine learning algorithm, Extreme Mechanics Letters 18 (2018) 19–28.
[16] L. Xiao, Y. Chen, C.K. Chang, Bayesian model averaging of bayesian network classifiers for intrusion detection, in: Proceedings of the 38th International Computer Software and Applications Conference Workshops, IEEE, 2014, pp. 128–133.
[17] H.M. Anwer, M. Farouk, A. Abdel-Hamid, A framework for efficient network anomaly intrusion detection with features selection, in: Proceedings of the 9th International Conference on Information and Communication Systems (ICICS), IEEE, 2018, pp. 157–162.
[18] J. Zhang, M. Zulkernine, A hybrid network intrusion detection technique using random forests, in: Proceedings of the First International Conference on Availability, Reliability and Security (ARES'06), IEEE, 2006, pp. 262–269.
[19] J. Yang, J. Deng, S. Li, Y. Hao, Improved traffic detection with support vector machine based on restricted boltzmann machine, Soft Computing 21 (2017) 3101–3112.
[20] E. Hodo, X. Bellekens, A. Hamilton, P.-L. Dubouilh, E. Iorkyase, C. Tachtatzis, R. Atkinson, Threat analysis of IoT networks using artificial neural network intrusion detection system, in: Proceedings of the 2016 International Symposium on Networks, Computers and Communications (ISNCC), IEEE, 2016, pp. 1–6.
[21] N. Shone, T.N. Ngoc, V.D. Phai, Q. Shi, A deep learning approach to network intrusion detection, IEEE Transactions on Emerging Topics in Computational Intelligence 2 (2018) 41–50.
[22] A. Javaid, Q. Niyaz, W. Sun, M. Alam, A deep learning approach for network intrusion detection system, in: Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies, 2016, pp. 21–26.
[23] M.Z. Alom, V. Bontupalli, T.M. Taha, Intrusion detection using deep belief networks, in: Proceedings of the 2015 National Aerospace and Electronics Conference (NAECON), IEEE, 2015, pp. 339–344.
[24] M. Sun, Z. Song, X. Jiang, J. Pan, Y. Pang, Learning pooling for convolutional neural network, Neurocomputing 224 (2017) 96–104.
[25] Y. Chen, J. Yang, J. Qian, Recurrent neural network for facial landmark detection, Neurocomputing 219 (2017) 26–38.
[26] J. Qiao, G. Wang, X. Li, W. Li, A self-organizing deep belief network for nonlinear system modeling, Applied Soft Computing 65 (2018) 170–183.
[27] D. Hendrycks, K. Gimpel, A baseline for detecting misclassified and out-of-distribution examples in neural networks, in: Proceedings of the 5th International Conference on Learning Representations (ICLR), 2017, pp. 1–12.
[28] B. Mozafari, P. Sarkar, M. Franklin, M. Jordan, S. Madden, Scaling up crowd-sourcing to very large datasets: a case for active learning, Proceedings of the VLDB Endowment 8 (2014) 125–136.
[29] D. Hendrycks, M. Mazeika, T.G. Dietterich, Deep anomaly detection with outlier exposure, in: Proceedings of the 7th International Conference on Learning Representations (ICLR), 2019, pp. 1–18.
[30] H. Jiang, B. Kim, M. Guan, M. Gupta, To trust or not to trust a classifier, in: Advances in Neural Information Processing Systems, volume 31, 2018, pp. 5541–5552.
[31] P. Zhang, J. Wang, A. Farhadi, M. Hebert, D. Parikh, Predicting failures of vision systems, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, IEEE, 2014, pp. 3566–3573.
[32] Z. Chen, Q. Chen, B. Hou, M. Ahmed, Z. Li, Improving machine-based entity resolution with limited human effort: A risk perspective, in: Proceedings of the International Workshop on Real-Time Business Intelligence and Analytics, 2018, pp. 1–5.
[33] R. Kohavi et al., A study of cross-validation and bootstrap for accuracy estimation and model selection, in: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI), volume 2, Montreal, Canada, 1995, pp. 1137–1145.
[34] B. Neyshabur, S. Bhojanapalli, D. McAllester, N. Srebro, Exploring generalization in deep learning, Advances in Neural Information Processing Systems (2017) 5947–5956.

[35] C. Zhang, S. Bengio, M. Hardt, B. Recht, O. Vinyals, Understanding deep learning requires rethinking generalization, in: Proceedings of the 5th International Conference on Learning Representations (ICLR), 2017, pp. 1–11.
[36] S.J. Pan, Q. Yang, A survey on transfer learning, IEEE Transactions on Knowledge and Data Engineering (TKDE) 22 (2010) 1345–1359.
[37] Y. Wei, Y. Zhang, J. Huang, Q. Yang, Transfer learning via learning to transfer, in: Proceedings of the 35th International Conference on Machine Learning (ICML), volume 80, 2018, pp. 5072–5081.
[38] N. Houlsby, A. Giurgiu, S. Jastrzkebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, S. Gelly, Parameter-efficient transfer learning for NLP, in: Proceedings of the 36th International Conference on Machine Learning (ICML), volume 97, 2019, pp. 2790–2799.
[39] M. Long, G. Ding, J. Wang, J. Sun, Y. Guo, P.S. Yu, Transfer sparse coding for robust image representation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013, pp. 407–414.
[40] M. Long, Y. Cao, J. Wang, M.I. Jordan, Learning transferable features with deep adaptation networks, in: Proceedings of the 32nd International Conference on Machine Learning (ICML), volume 37, 2015, pp. 97–105.
[41] H. Zhao, R.T. des Combes, K. Zhang, G.J. Gordon, On learning invariant representations for domain adaptation, in: Proceedings of the 36th International Conference on Machine Learning (ICML), volume 97, 2019, pp. 7523–7532.
[42] Z. Wu, X. Wang, J.E. Gonzalez, T. Goldstein, L.S. Davis, Ace: Adapting to changing environments for semantic segmentation, in: Proceedings of the IEEE International Conference on Computer Vision (CVPR), 2019, pp. 2121–2130.
[43] T. Kim, M. Jeong, S. Kim, S. Choi, C. Kim, Diversify and match: A domain adaptive representation learning paradigm for object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 12456–12465.
[44] Q. Chen, Z. Chen, Y. Nafa, T. Duan, Z. Li, Adaptive deep learning for entity resolution by risk analysis, CoRR abs/2012.03513 (2020) 1–31.
[45] Z.-H. Zhou, Ensemble learning, Encyclopedia of biometrics 1 (2009) 270–273.
[46] O. Sagi, L. Rokach, Ensemble learning: A survey, Wiley Interdisciplinary Reviews, Data Mining and Knowledge Discovery 8 (2018) e1249.
[47] T. Fawcett, An introduction to ROC analysis, Pattern Recognition Letters 27 (2006) 861–874.
[48] G. Tardivo, Value at risk (var): The new benchmark for managing market risk, Journal of Financial Management & Analysis 15 (2002) 16–26.
[49] S.R. Islam, W. Eberle, S.K. Ghafoor, A. Siraj, M. Rogers, Domain knowledge aided explainable artificial intelligence for intrusion detection and response, CoRR abs/1911.09853 (2019) 1–11.
[50] A. Husain, A. Salem, C. Jim, G. Dimitoglou, Development of an efficient network intrusion detection model using extreme gradient boosting (XGBoost) on the UNSW-NB15 dataset, in: Proceedings of the 2019 International Symposium on Signal Processing and Information Technology (ISSPIT), IEEE, 2019, pp. 1–7.
[51] T. Chen, G. Carlos, XGBoost: A scalable tree boosting system, in: KDD, ACM, 2016, pp. 785–794.
[52] Novelty and outlier detection, URL:https://scikit-learn.org/stable/modules/outlier_detection.html#outlier-detection, 2021. Accessed September 28, 2021.
[53] A. Trendowicz, R. Jeffery, Classification and regression trees, in: Software Project Effort Estimation, Springer, 2014, pp. 295–304.
[54] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, G. Hullender, Learning to rank using gradient descent, in: Proceedings of the 22nd international conference on Machine learning, 2005, pp. 89–96.
[55] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Proceedings of the 3rd International Conference on Learning Representations (ICLR), 2015, pp. 1–11.



Lijun Zhang received the B.E. degree in computer science and technology from Jilin University, China in 2000, and the M.S. and Ph.D. degrees in computer science and technology from Northwestern Polytechnical University, China in 2003 and 2013 respectively. He is currently an associate professor with the School of Computer Science, Northwestern Polytechnical University. From 2019 to 2020, he was a visiting scholar at RMIT University, Australia. His research interests include data analysis and mining, machine learning, database management system, and distributed database technology.



Xingyu Lu is a master student in the School of Computer Science, Northwestern Polytechnical University. His research interests include artificial intelligence and network security.

Zhaoqiang Chen is a Ph.D. student in the School of Computer Science, Northwestern Polytechnical University. His research interests include data quality and risk analysis for artificial intelligence.

Tianwei Liu is a master student in the School of Computer Science, Northwestern Polytechnical University. His research interests include artificial intelligence and Computer Vision.

Qun Chen received his bachelor degree from Tsinghua University, China, and his Ph.D. from National University of Singapore, Singapore. He is currently a professor in the School of Computer Science in Northwestern Polytechnical University. His current research interests focus on gradual machine learning and risk analysis for AI.

Zhanhuai Li is a professor in the School of Computer Science in Northwestern Polytechnical University. His research interests include data storage and management. He has served as Program Committee Chair or Member in various conferences and committees.