Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH Open Access

Robust and accurate performance anomaly detection and prediction for cloud applications: a novel ensemble learning-based framework

Ruyue Xin[1], Hongyun Liu[1], Peng Chen[2*] and Zhiming Zhao[1*]

Abstract

Effectively detecting run-time performance anomalies is crucial for clouds to identify abnormal performance behavior and forestall future incidents. To be used for real-world applications, an effective anomaly detection framework should meet three main challenging requirements: high accuracy for identifying anomalies, good robustness when application patterns change, and prediction ability for upcoming anomalies. Unfortunately, existing research about performance anomaly detection usually focuses on improving detection accuracy, while little research tackles the three challenges simultaneously. We conduct experiments for existing detection methods on multiple application monitoring data, and results show that existing detection methods usually focus on different features in data, which will lead to their diverse performance on different data patterns. Therefore, existing anomaly detection methods have difficulty improving detection accuracy and robustness and predicting anomalies. To address the three requirements, we propose an Ensemble Learning-Based Detection (ELBD) framework which integrates existing well-selected detection methods. The framework includes three classic linear ensemble methods (maximum, average, and weighted average) and a novel deep ensemble method. Our experiments show that the ELBD framework realizes better detection accuracy and robustness, where the deep ensemble method can achieve the most accurate and robust detection for cloud applications. In addition, it can predict anomalies in the next four minutes with an F1 score higher than 0.8. The paper also proposes a new indicator *ARP_score* to measure detection accuracy, robustness, and multi-step prediction ability. The *ARP_score* of the deep ensemble method is 5.1821, which is much higher than other detection methods.

Keywords Performance anomaly detection, Algorithm robustness, Anomaly prediction, Ensemble learning, Deep ensemble

Introduction

The run-time status of cloud applications can be continuously monitored through system-related metrics, e.g., CPU and memory usage [1]. Performance anomaly detection plays a vital role in operating cloud services, and applications [2, 3]. Cloud performance anomalies such as degraded response time, often caused by underlying system resource shortages, may severely affect the quality of an application's user experience (QoE) and service (QoS). Therefore, effectively analyzing patterns of monitoring system-related metrics and identifying abnormal performance in real-time is crucial for continuously delivering the business value of a cloud application. In this context, we can highlight three challenging requirements for a performance anomaly detection framework. First, the detection must achieve high accuracy to ensure anomalies can be found as accurately

*Correspondence:
Peng Chen
chenpeng@mail.xhu.edu.cn
Zhiming Zhao
z.zhao@uva.nl
[1] Multiscale Networked Systems (MNS) research group, University of Amsterdam, Amsterdam, Netherlands
[2] School of Computer and Software Engineering, Xihua University, Chengdu, China

as possible. Second, detection algorithm robustness is essential. Different data distributions exist in multiple monitoring data, which requires a robust algorithm to meet changes in data patterns and maintain performance consistency. Finally, to prevent potential application violations effectively, it is vital to make a multi-step prediction of future anomalies.

Existing anomaly detection methods have often been developed using statistics [4] or machine learning [5, 6] based methods. Most methods focus on improving detection accuracy. For example, Audibert et al. [7] developed the USAD based on an adversely trained AutoEncoder and achieved the best detection accuracy. Studies on improving the robustness of detection methods usually use adversarial training, which needs to make a trade-off between robustness and accuracy [8], rather than simultaneously improving accuracy and robustness. In addition, research on anomaly prediction mainly focuses on one-step prediction [9], which has limited effect in preventing potential performance anomalies. Existing research explores different aspects of the three challenging requirements, but few studies simultaneously tackle the challenges of accuracy, robustness, and multi-step prediction ability. Besides, there are also no effective indicators to measure the combination of the three requirements.

Moreover, the development of performance anomaly detection has to handle two data challenges.

- Missing data labels. Most of the monitoring data does not contain labels that can be immediately used for training a machine learning-based model, and labeling time-series data is often manual and time-consuming.
- Data noise. Monitoring data collected from a distributed network often contain noises, which can significantly influence the performance of the anomaly detection methods and increase the false-positive detection.

Thus, for performance anomaly detection, we define our research question as *"how to effectively detect and predict performance anomalies with high accuracy and good robustness?"*. To address the two data challenges, we focus on unsupervised and weakly supervised detection methods and provide feature extraction to filter noise in data. To answer our research question, we first explore existing unsupervised anomaly detection methods and observe their detection performance on different datasets. Then, to improve detection accuracy, robustness, and prediction ability, we develop an Ensemble Learning-Based Detection (ELBD) framework that incorporates classic detection methods rather than enhances a single model. The contributions in this paper mainly include:

- We characterize four typical base detection methods on different datasets, and the results show that their detection performance is not good for detection accuracy, robustness, and prediction.
- Based on base detection methods, we propose an ELBD framework including three classic linear ensemble methods (maximum, average, and weighted average) and a deep ensemble method.
- We propose *ARP_score* to evaluate detection performance in terms of accuracy, robustness, and multi-step prediction.
- We evaluated the methods in the ELBD framework on different datasets, and the results show that the deep ensemble method achieves the highest *ARP_score* 5.1821.

The rest of the paper is organized as follows. In Related works section, we review existing performance anomaly detection methods, specifically ensemble learning. In Base performance anomaly detection methods section, we provide base detection methods and an evaluation of their performance. In Ensemble learning-based detection framework section, we propose the ELBD framework and evaluate detection accuracy, robustness, and prediction ability. Finally, discussion and conclusion are provided in Discussion and Conclusion and future work sections.

Related works

Performance anomaly detection is a process of detecting abnormal performance phenomena and predicting anomalies to forestall future incidents [10]. Research about performance anomaly detection is ongoing rapidly, and machine learning methods are widely applied [11]. This section will briefly review machine learning-based anomaly detection methods and specifically highlight ensemble learning.

Machine learning-based anomaly detection methods

Machine learning-based anomaly detection methods can be reviewed in terms of supervised, semi-supervised, and unsupervised learning. Supervised learning methods have high accuracy [5], but they are ineffective for application monitoring data because data labels are usually missing in reality and manually labeling data manually is time-consuming. Semi-supervised learning methods are developed when fewer labels exist, and unsupervised learning methods are used when no labels exist. Semi-supervised methods typically outperform unsupervised methods, but unsupervised methods are better suited for actual industrial scenarios [12].

In Table 1, we provide a classification of unsupervised performance anomaly detection methods. The table includes traditional methods such as tree-based,

Table 1 A classification of classic unsupervised performance anomaly detection methods

Type	Method	Description
Density-based	LOF [6]	Local Outlier Factor
	COF [14]	Connectivity-based Outlier Factor
	LOCI [15]	Local Correlation Integral
Distance-Based	KNN [16]	K-Nearest Neighbors
	LDOF [17]	Local Distance-based Outlier Factor
Kernel-based	OCSVM [18]	One-Class Support Vector Machines
	RSVM [19]	Robust Support Vector Machines
Tree-based	IForest [20]	Isolation Forest
Deep learning	AutoEncoder [21]	Fully connected AutoEncoder
	VAE [22]	Variational AutoEncoder

kernel-based, distance-based, and density-based. They usually focus on different features in data, and their performance varies for different datasets, which will be verified in Base performance anomaly detection methods section. Deep learning methods are also developing rapidly recently. For example, Su et al. [13] provide a stochastic recurrent neural network named OmniAnomaly for multivariate time series anomaly detection. Deep learning methods can achieve high detection accuracy, but model training is usually time-consuming.

Researchers usually improve algorithm robustness through adversarial training, which uses deep learning methods to defend generated adversarial examples [23]. For example, Hashemi et al. [24] enhance the robustness of an intrusion detection system in the presence of adversarial examples by utilizing denoising autoencoders. However, there is usually a trade-off between model accuracy and robustness [8], which makes it a challenge to improve model robustness and accuracy simultaneously. In addition, research on anomaly prediction usually focuses on univariate data and one-step prediction. For example, Wu et al. [9] provide a prediction-driven anomaly detection method that relies on Long Short Term Memory (LSTM) with univariate time-series data.

In conclusion, machine learning methods, especially semi-supervised and unsupervised methods, can be considered for performance anomaly detection because fewer labels exist. While different methods usually focus on different data features, we can consider integrating existing methods, for example, LOF, KNN, OCSVM, and IForest, in Table 1. To improve detection accuracy and robustness simultaneously, ensemble learning instead of adversarial training to integrate existing detection methods can be considered. We will introduce related work to ensemble learning next. In addition, we provide a multi-step prediction based on multi-variate metrics for performance anomalies in this paper.

Ensemble learning

Ensemble learning is proposed to improve the accuracy and reduce the variance of an automated decision-making system [25]. The primary assumption of ensemble learning is that by combining several base models, the errors of a single model will likely be compensated by other models [26]. For anomaly detection, the ensemble of anomaly scores by taking the maximum, and average actions can be found in [27]. Research about ensemble learning can be reviewed based on supervised classification, semi-supervised and unsupervised clustering ensemble.

Some research already focuses on ensemble learning with machine learning methods. As for *supervised ensemble learning*, Tyralis et al. [28] propose an ensemble learning method by combining ten machine learning algorithms and estimating the weights through a k-fold cross-validation procedure. Tama et al. [29] propose a stacked ensemble, which uses three classifiers (random forest, gradient boosting machine, and XGBoost) and provides a generalized linear model (GLM) as a combiner. Adeyemo et al. [30] focus on network intrusion detection and implement two ensemble methods and a deep learning method (LSTM). The two ensemble methods include a homogeneous method that uses an optimized bagged random forest algorithm and a heterogeneous method that is an averaged probability method of a voting ensemble for four standard classifiers. These studies of ensemble learning mainly focus on weight calculations or linear combinations of different base models.

Semi-supervised ensemble learning mainly focuses on expanding the labeled training set and utilizing the expanded training set to do classification or regression [31]. For example, Jian et al. [32] present a sample information-based synthetic minority oversampling technique to balance the labeled dataset and use variable weighted voting for integrating base models. This research focuses on the data label issue with semi-supervised learning, but the ensemble is linear. *Unsupervised ensemble learning*, also known as consensus clustering, is to find the optimal combination strategy of individual clustering. Ensemble clustering can be classified into three categories, pairwise co-occurrence based methods [33], graph partitioning based methods [34] and median partition-based methods [35]. Unlu et al. [36] provide a weighting policy based on internal clustering quality measures, which gives different importance to individual clustering. This research provides a weighted policy to integrate individual models, but it works linearly.

For ensemble learning, we can see that combining different methods is challenging, and many methods focus on linear combinations, which limits information extraction and detection performance improvement. Therefore, to get an effective model that improves detection accuracy and robustness and makes prediction of performance anomaly detection, the nonlinear combination of different base detection methods is further investigated in this paper.

Base performance anomaly detection methods

For performance anomaly detection of cloud applications, we provide feature extraction for original data and explore the performance of four base detection methods in this section.

Problem definition

Multivariate time-series data are timestamped data points sequences and can be represented as D. Then each data point will be D_i^t ($i = [1, ..., n]$ is the index of resource metrics. n is the number of resource metrics. $t \in N^*$ is the index of timestamps). Multivariate time-series data anomaly detection is to learn the characteristics of data D and determine whether an observation D_{n+1} is anomalous or not. For multi-step anomaly prediction, we will use data D for training, and determine whether $D_{n+1}, D_{n+2}, ..., D_{n+p}$ is anomalous.

In this paper, we first provide the performance of classic detection methods. Then we propose an ELBD framework, which is developed based on ensemble learning and aims to improve detection accuracy and robustness by integrating information extracted by classic detection methods non-linearly. In addition, we implement multi-step prediction ability in the deep ensemble method in ELBD framework.

Feature extraction

Multivariate data usually contains noise, which can induce unnecessary variance in a model. Therefore, preprocessing data through feature extraction to remove redundant information and reduce data dimension is needed. For feature extraction, Principal Components Analysis (PCA) [37] is a classic and most used method. PCA is an unsupervised method that uses eigenvalue decomposition to compress and denoise data, which is suitable as the feature extraction method considering there are no labels or fewer labels in reality.

PCA is a method to transform a dataset with lots of variables into a smaller one containing most of the original information. The process steps of PCA are: 1) getting the covariance matrix of original features; 2) calculating eigenvectors and eigenvalues of the covariance matrix to identify principal components; 3) sorting eigenvalues and selecting eigenvectors with high eigenvalues as feature vectors; 4) recasting original data based on feature vectors. In step 3, the number of selected eigenvectors determines the data dimensions after reduction. In practice, we set the reduction dimension based on a calculated percentage of variance [38]. According to these calculations, PCA achieves principal feature selection and data dimension reduction. Finally, we apply PCA to all monitoring data and use the data with low dimensions as the input of anomaly detection methods.

Base detection methods

Different anomaly detection methods usually focus on different features in data, such as density-based and distance-based, and result in diverse performance on data. Therefore, to comprehensively understand the characteristics of monitoring data, we select four classic methods (IForest, KNN, LOF, OCSVM) in Table 1 as base detection methods.

IForest is based on the Decision Tree algorithm [39]. Many isolation trees make up an isolation forest to make anomaly detection results more credible. KNN is a distance-based algorithm [16]. It calculates each point's distance (such as Euclidean, Manhattan) with k nearest neighbors and sets the distance as an anomaly score. LOF is a density-based algorithm [6]. By comparing a point's local density to its neighbors' local densities, nodes with lower densities than their neighbors will be considered anomalies. OCSVM is based on Support Vector Machine (SVM) [18]. SVM can project data through a non-linear function to a high-dimensional space, and points are separated into different classes. Because kernel function calculation is time-consuming, it usually works slowly for large-scale data.

For each base method, the input is preprocessed data. The processing of input data includes model initialization, fitting data, and output anomaly scores. Model initialization includes the setup of hyper-parameters, such as anomaly fractions, which can be set based on data characteristics. After fitting the data, an anomaly score vector will be output. We use the anomaly score vector of each detection method to identify anomalies and evaluate the performance of each detection method.

Experiments and results

Dataset

In our experiments, we use a Decentralized Application (DApp) monitoring data and two public datasets.

DApps monitoring data. In business scenarios where real-time transactions are required, e.g., energy trading or crowd journalisms [40], the Quality of Service (QoS) metrics of a DApp, such as transaction throughput, latency, and failure rates, are critical to the business value. To deliver such a quality-critical DApp in cloud environments, one needs to select cloud services

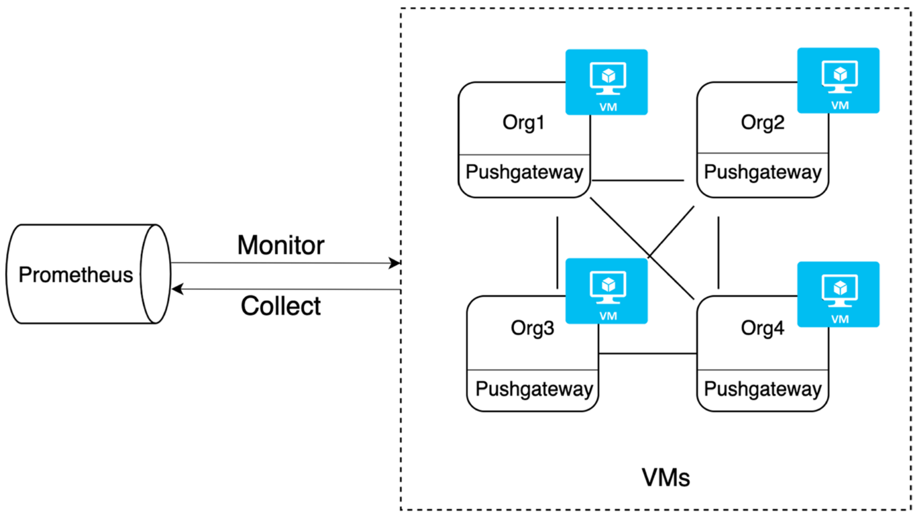

Fig. 1 The monitor component and a DApp in cloud

Table 2 Description of system resource metrics

Resource Metrics	Description
CPU related	Per core and overall load, usage, idle time, I/O wait time, hard and soft interrupt counts, context switch count, etc.
Memory related	Free, cached, active, inactive, dirty memory, etc.
Disk related	Disk space used, IOps, I/O usage, read/write rate, etc.
Network related	Receive/transmit network traffic, etc.

Table 3 General information of three datasets

Dataset	Number of samples	Number of features	Number of extracted features	Anomaly fraction (%)
DApp monitoring data	3237	229	15	28.14
SMD data	28479	38	5	9.46
Vichalana data	45486	13	6	6.45

carefully, customize their capacities, and monitor the run-time status of the application. Figure 1 shows a DApp example developed with Hyperledger Fabric[1]. For the DApp, different organizations, which contain many peer nodes, are deployed on different cloud infrastructure services (VMs) and monitored by a tool Prometheus[2]. We use Prometheus to collect real-time data and use Caliper[3] to simulate workload generation.

For a running DApp, we mainly collect system resource metrics, which can be seen in Table 2. When the DApp receives transaction requests stably, we add system pressures with *stress-ng*[4], such as disk pressure to inject anomalies manually. We increase disk pressure by 20 minutes every hour. We monitor the DApp for twelve hours and collect data at 15-second intervals. Ultimately, the DApp monitoring data contains 3237 samples and 229 resource-related metrics for our experiments. The general information can be seen in the Table 3.

Public dataset. SMD is divided into two subsets of equal size: the first half is the training set and the second half is the testing set. SMD (Server Machine Dataset) is a dataset collected and made publicly available by a large internet company [13]. It contains data collected from many different server machines and includes 38 metrics. In addition, domain experts have labeled anomalies in SMD based on incident reports.

Vichalana is a multivariate time-series dataset that can be used for performance anomaly detection in API Gateways [41]. It has different anomalies, such as high CPU and memory usage. Performance metrics in this dataset are collected when the system operates in normal and anomalous mode. The information of SMD and Vichalana data used in our experiments can be seen in Table 3.

Experimental settings

The DApp monitoring data is collected from a deployed DApp in a cloud environment. We use Microsoft Azure[5] as the cloud environment and deploy the monitor component and DApp separately. The monitor component is

[1] https://www.hyperledger.org/use/fabric

[2] https://prometheus.io/

[3] https://www.hyperledger.org/use/caliper

[4] https://kernel.ubuntu.com/~cking/tarballs/stress-ng/

[5] https://azure.microsoft.com/en-us/

Table 4 Performance of different detection methods on three datasets. For each dataset, the F1 score of the best detection method is shown in bold

Detection methods	DApp monitoring data		SMD data		Vichalana data	
	F1 score	Time(s)	F1 score	Time(s)	F1 score	Time(s)
IForest	0.791	0.318±0.0121	**0.7515**	1.278±0.0195	0.658	1.9814±0.0704
KNN	**0.8033**	0.0246±0.0021	0.5713	0.311±0.0047	0.5519	0.7758±0.0693
LOF	0.5143	0.0439±0.0015	0.5468	0.5379±0.0108	0.5128	1.4684±0.1229
OCSVM	0.737	0.3054±0.0076	0.6047	23.9234±0.8924	**0.6778**	190.118±10.5769

deployed on a VM, with the following properties: Ubuntu 18.04 as the operating system; 2CPU; 4G Memory; and 32GB of Storage. The DApp is deployed on VMs with Ubuntu 18.04 as the operating system, 4CPU, 16G memory, and 32GB of storage.

For feature extraction in data pre-processing, PCA needs to retain as much variance information of the original data as possible, such as 95%. Therefore, we set reduction dimensions as 15 for DApp monitoring data, 5 for SMD data, and 6 for Vichalana data based on a calculated percentage of variance [38].

As for each base detection method, their hyper-parameters are set as below. Anomaly fractions need to be determined first. For the DApp monitoring data, because we inject anomalies 20 minutes every hour, the anomaly fraction is set as 0.3. For SMD and Vichalana data, we use the default anomaly fraction, which is 0.1. Next, the hyper-parameters of each base method need to be determined. We set the tree number for IForest to 100. The neighbor number in KNN is 5. In LOF, we set the neighbor number as 20. In OCSVM, we use the Radial Basis Function (RBF) kernel function.

Evaluation indicators

The performance of these detection methods is evaluated in three aspects: accuracy, robustness, and prediction ability. We use Precision, Recall, and F1 score to indicate accuracy. Precision is about how much of the data detected as anomalies is true anomalies, while recall is about how much of the real anomaly data is detected as anomalies. The F1 score is a function of both Precision and Recall.

$$F1\ score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{1}$$

Therefore, we mainly focus on the F1 score for detection accuracy. Our experiment results also evaluate and present the time spent on each unsupervised detection method and test time for the deep ensemble method. For robustness, we test detection methods on three different datasets and rank detection accuracy to represent performance consistency, which can clearly show the detection performance comparison [42]. We calculate robustness as the average ranking of detection methods on the three datasets. Finally, we normalize the rank and get the robustness score:

$$Robustness\ score = \frac{Rank - Rank_{max}}{Rank_{min} - Rank_{max}} \tag{2}$$

Here, $Rank_{max}$ is the maximum of rank numbers, and $Rank_{min}$ is the minimum of rank numbers. We evaluate prediction ability with accuracy, which is also represented by the F1 score. We set the threshold of 0.8 and calculate the prediction score with

$$Prediction\ score = \sum_{i=1}^{pt} F1\ score_i \tag{3}$$

Here, pt is the furthest predicted time in minutes. The prediction score considers both the furthest prediction time and prediction accuracy because the longer time and more accurate prediction can make it easier to avoid anomalies for applications. Finally, we define the indicator ARP_score for each method considering detection accuracy, robustness, and prediction as:

$$\begin{aligned} ARP\_score = &\frac{1}{d}\sum_{i=1}^{d}(F1\ score_i + Robustness\ score_i \\ &+ Prediction\ score_i) \end{aligned} \tag{4}$$

Here, d is the number of datasets. We take the total of these three scores as the detection performance of a detection method on a dataset. We take the average of each detection method on different datasets with multiple datasets as the final indicator of its detection performance.

Experimental results

We apply the four base methods (IForest, KNN, LOF, and OCSVM) to the DApp monitoring data, SMD, and Vichalana data. The performance of their detection accuracy can be seen in Table 4.

For the DApp monitoring data, we can see that the KNN has the highest F1 score, 0.8033, demonstrating that the data has clustering characteristics because KNN is good at identifying clusters in data. IForest takes into account different features in the data. IForest usually has good detection performance [43], as well as on the DApp monitoring data with an F1 score of 0.791. If the abnormal features are concentrated in a few dimensions, it will be hard to detect anomalies for LOF. Therefore, LOF has the lowest F1 score, 0.5143, for the DApp monitoring data. The F1 score of OCSVM is 0.737, which is not high enough because the projection through a kernel function cannot be divided into normal and abnormal data very well. For time spent, we can see that IForest and OCSVM spend about 0.3s, which is higher than other base methods because the calculation of features takes some time, but the time spent is under 0.5s overall, which is not high actually. As a result, for the DApp monitoring data, the KNN is the best of the four base methods.

For SMD data, we can see that IForest has the highest F1 score, 0.7515, which shows the advantage of IForest for anomaly classification through multiple features. However, F1 scores are not high for other base methods, showing too much noise in this dataset, and the overall distribution of normal and abnormal data is similar. Thus, we can say that anomalies may be mainly in a few features in the SMD data. In addition, the time spent on OCSVM is higher than on others because the kernel function calculation in OCSVM is time-consuming. On the other hand, IForest has the best detection accuracy and takes about 1.3 s, which is the best detection method.

For Vichalana data, we can see that OCSVM has the highest F1 score, 0.6778, showing that the non-linear projection can classify normal and abnormal data but is not very accurate. The F1 score of IForest is 0.658, slightly lower than OCSVM, which means that abnormal data distribution varies in different features, making it hard to detect. The F1 scores of KNN and LOF are pretty low, showing that the overall distribution of normal and abnormal data is also similar. It is worth noting that the time spent on OCSVM is relatively high because the dataset includes more than 40k samples, and it takes too much time for kernel function calculation in OCSVM. Here, IForest only takes about 2s, which is quite faster than OCSVM.

In conclusion, we can see that detection accuracy is not high enough for these base detection methods. In addition, the performance of these methods varies for the three datasets. For example, KNN performs the best on the DApp monitoring data but relatively poorly on the SMD and Vichalana data. Furthermore, these detection methods have no prediction ability. Thus, for the three challenges: high accuracy, good robustness, and multi-step prediction, it is critical to develop suitable performance anomaly detection methods for cloud applications.

Ensemble learning-based detection framework

Base detection methods focus on different features in data and have diverse performances. Therefore, it is reasonable to consider that the integration of base methods can extract more features from data and improve detection performance. Furthermore, ensemble learning is proposed with the assumption that by combining several base models, the errors of a single model will be compensated by others. Therefore, we consider integrating base methods with ensemble learning and propose an Ensemble Learning-Based Detection (ELBD) framework, including three classic linear ensemble methods (maximum, average, and weighted average) and a deep ensemble method.

Basic idea

The ELBD framework can be seen in Fig. 2. First, input data is multivariate time-series monitoring data, including system and service level data, which can be collected and used as input. In this paper, we mainly focus on system resource data. We can represent input data as D_i^t ($i = [1, ..., n]$ is the index of resource metrics. n is number of resource metrics. $t \in N^*$ is the index of timestamps). Next, pre-processing needs to be done for the input data, including feature extraction and train/test split. Feature extraction has been introduced in Feature extraction section. There is no need to do the train/test split for unsupervised learning. However, the train/test split is important to avoid over-fitting for weakly-supervised learning. Therefore, we do the train/test split for the deep ensemble method, as seen in the experimental settings. After pre-processing, data D_j^t ($j = [1, ..., d]$ is the index of data dimensions. d is data dimensions after reduction) will be the input of anomaly detection methods.

The base method selection provides unsupervised detection methods. In this paper, we manually select four typical base methods, which have been introduced in detail in Base detection methods section. The output of base methods can be assembled as an anomaly score matrix. For the matrix, we provide three linear ensemble methods without training and a deep ensemble method, which needs to be trained with a neural network. The output of anomaly detection methods can be represented as C_m^t (m is the index of all detection methods). We mainly focus on accuracy, robustness, and multi-step prediction ability to evaluate the multiple detection methods.

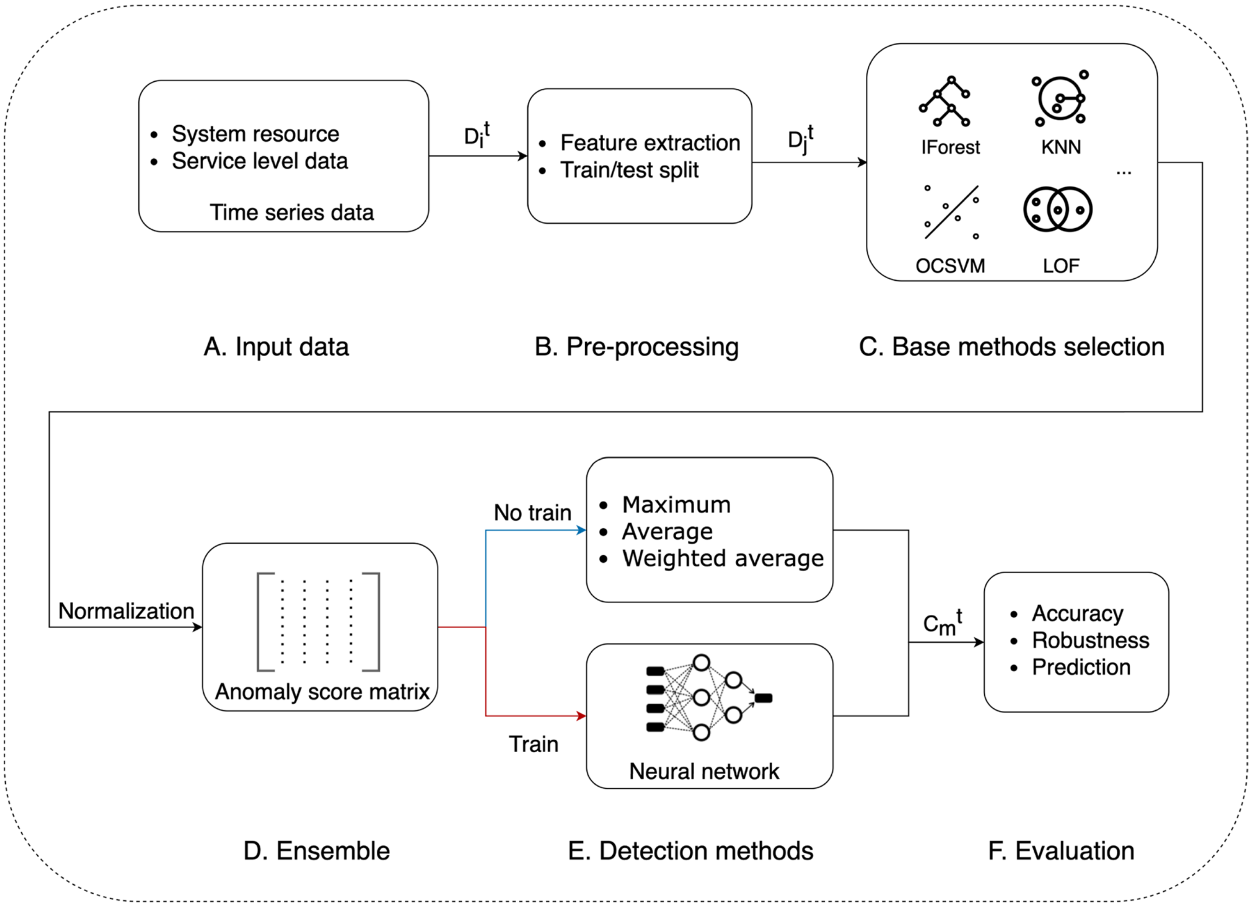

Fig. 2 ELBD framework, including three classic ensemble methods without training (blue line) and a deep ensemble method which need to train a neural network (red line)

Linear ensemble methods

The outputs of base methods have different meanings and scales. For example, the anomaly score of IForest is calculated based on path depth, and KNN is based on distance. Because all the features should be measured in the same units, we apply z-score normalization [44] to ensure that all outputs have the same scale. The z-score method uses the mean and standard deviation of the original data for normalization so that the processed data follows the normal distribution. After normalization, we can represent the anomaly score vector C_k^t (k represents base detection methods) of each base method as O_k^t. Here, k is the index of base detection methods and $k \in [1, r]$, r is the number of base methods. Therefore, by taking each anomaly score vector as a column, we can get the anomaly scores matrix M:

$$M = \begin{bmatrix} O_1^1 & O_2^1 & O_3^1 & O_4^1 \\ O_1^2 & O_2^2 & O_3^2 & O_4^2 \\ \vdots & \vdots & \vdots & \vdots \\ O_1^t & O_2^t & O_3^t & O_4^t \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

The left side of Table 5 can be seen as an example of the matrix. For matrix M, we provide linear ensemble methods first, including maximum ensemble, average ensemble, and weighted average ensemble.

The **maximum ensemble** is to select the max value of each row in matrix M and form a new anomaly score vector.

$$V_{max} = \max_k O_k^t, t \in N^* \tag{5}$$

Table 5 Linear ensemble methods example: on the left side is anomaly scores obtained by each base method; on the right side is anomaly scores obtained through ensemble methods

Index	IForest	KNN	LOF	OCSVM	Max	Avg	Weighted Avg
1	-0.41	-0.23	0.14	-0.88	0.14	-0.35	-0.49
2	-0.18	-0.03	0.63	-0.86	0.63	-0.11	-0.33
3	2.29	5.14	1.07	0.62	5.14	2.28	2.76
4	2.36	4.56	0.86	0.11	4.56	1.97	2.42
5	1.99	1.5	-0.3	-0.19	1.99	0.75	1.14

The **average ensemble** is to calculate the average of each row and form a new anomaly score vector.

$$V_{avg} = \frac{1}{r}\sum_{k=1}^{r} O_k^t, t \in N^* \tag{6}$$

A limitation of the average ensemble is that each base detection method contributes equally to the final anomaly scores. However, some methods perform better or worse than others. Therefore, we can consider assigning different weights for these methods. For example, we assign more weights to better methods and fewer to worse ones. **Weighted average ensemble** is a method developed based on this idea.

Based on the assumption that if a mixed model can maximize the information provided by each model, the mixed model has the best weight distribution strategy. Mutual Information (MI) can measure the difference between models, which can be used to calculate the weight of each base method [45]. To calculate the mutual information of two models, we first need to transfer anomaly scores into anomaly classes (0 or 1). We assume n samples in the two models, a and b. Next, we use N_0^a and N_1^a to represent the number of normal and abnormal data in model a, and N_0^b and N_1^b to represent the number of normal and abnormal data in model b. In addition, N_0^{ab} and N_1^{ab} represent the data that is detected as normal and abnormal by both models. Then we can calculate the MI of models a and b:

$$\begin{aligned} I(A,B) =& N_0^{ab} \log \frac{n * N_0^{ab}}{N_0^a * N_0^b} + (N_0^a - N_0^{ab}) \log \frac{n * (N_0^a - N_0^{ab})}{N_0^a * N_1^b} \\ &+ (N_0^b - N_0^{ab}) \log \frac{n * (N_0^b - N_0^{ab})}{N_1^a * N_0^b} + N_1^{ab} \log \frac{n * N_1^{ab}}{N_1^a * N_1^b} \end{aligned} \tag{7}$$

To normalize it, we can calculate:

$$\phi(A,B) = \frac{I(A,B)}{\sqrt{\left(\sum_{i=0}^{1} N_i^a \log \frac{N_i^a}{n}\right)\left(\sum_{i=0}^{1} N_i^b \log \frac{N_i^b}{n}\right)}} \tag{8}$$

Therefore, the average mutual information of base method is:

$$\sigma_k = \frac{1}{r-1}\sum_{l=1, l\neq k}^{r} \phi\left(\lambda^{(k)}, \lambda^{(l)}\right), k \in [1, r] \tag{9}$$

Here, each base method is $\lambda^{(k)}$. σ_k is the standard value of the difference between models and $\sigma_k \in [0, 1]$. The smaller the value, the greater the difference between the two models. Based on the difference value of each model, we calculate the weights with $w_k = \sigma_k * Z$, Z is the normalization factor. The new anomaly score vector can be calculated as:

$$V_{w\_avg} = \frac{1}{r}\sum_{k=1}^{r} \sigma_k * O_k^t, t \in N^* \tag{10}$$

In Table 5, we provide five samples as an example to show how maximum, average, and weighted average ensemble methods work. In the left part of the table, we show the anomaly scores of four detection methods. In the right part, we can easily get the maximum and average anomaly scores. As for the weighted average ensemble, we assign the weights as (0.39, 0.28, 0.04, 0.29) for base methods based on the calculation. These new anomaly score vectors will be used to identify anomalies and evaluate the performance of these ensemble methods.

The deep ensemble method

The ensemble methods above try to combine different anomaly scores linearly. However, the linear combination may not represent the information extracted by each model well. Therefore, we provide a **deep ensemble** method in Fig. 3,

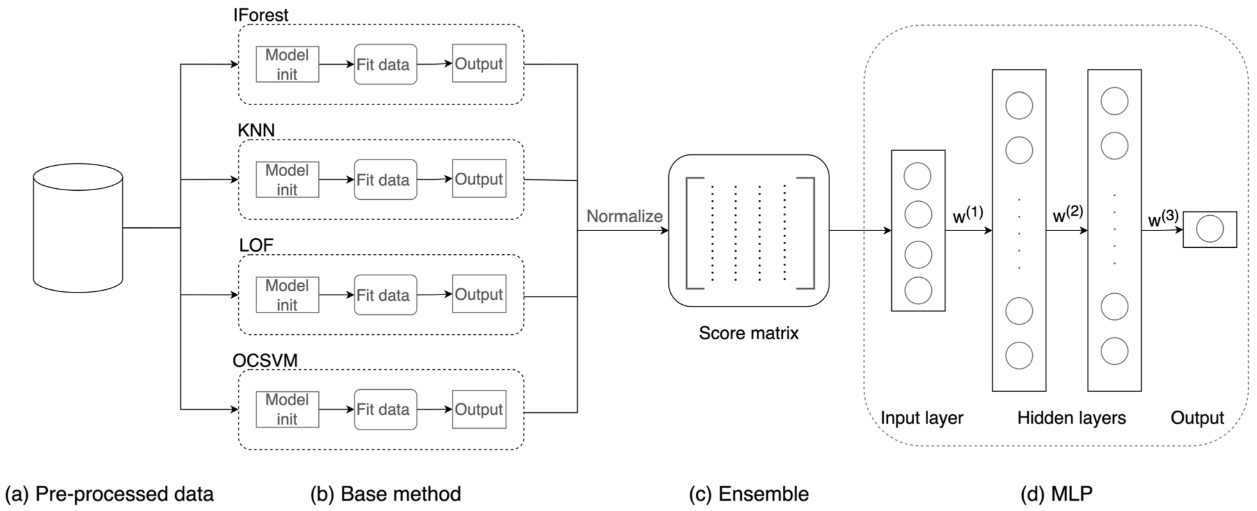

Fig. 3 The architecture of deep ensemble method includes four steps: (a)pre-processing data is sent to four (b)base methods; then after normalization, the (c)ensemble of their outputs forms a score matrix; we finally input the score matrix into an (d)MLP for training

and it combines base methods in a nonlinear way by using an Multi-Layer Perceptron (MLP). An MLP is a supplement to a feed-forward neural network. It consists of three layers: the input layer, the output layer, and the hidden layer. An MLP is suitable for classification or regression problems where inputs are assigned a class or real-value label. Therefore, the deep ensemble method is weakly-supervised and needs to be trained with some labels. Considering that there are fewer labels in reality, we design to train the deep ensemble with fewer labels and then test the trained model.

We provide the MLP architecture in Fig. 3. The input layer receives the anomaly score matrix M at first. We have two hidden layers consisting of an arbitrary number of neurons and use ReLU as an activation function. The output layer has one neuron and outputs the probability using the softmax activation function. We define $x = [O_1^t, O_2^t, O_3^t, O_4^t]$. $W^{(1)}$ and $b^{(1)}$ are weights and biases of the first layer. $W^{(2)}$, $b^{(2)}$ and $W^{(3)}$, $b^{(3)}$ are weights and bias of the two hidden layers. The output can be calculated based on the below functions.

$$\begin{aligned} z^{(1)} &= W^{(1)}x + b^{(1)}, \\ h^{(1)} &= ReLu(z^{(1)}), \\ z^{(2)} &= W^{(2)}h^{(1)} + b^{(2)}, \\ h^{(2)} &= ReLu(z^{(2)}), \\ z^{(3)} &= W^{(3)}h^{(2)} + b^{(3)}, \\ h^{(3)} &= softmax\left(z^{(3)}\right) \end{aligned} \tag{11}$$

For the output $h^{(3)}$, we can calculate the difference between the predicted and actual results y with the cross-entropy error function below. Here, y is the label at time t. The optimization goal is to minimize this equation by constantly adjusting parameters.

$$l = -y^T \log h^{(3)} \tag{12}$$

The deep ensemble method needs to be trained with fewer labels, and then the trained model can be applied to other data to detect anomalies. If we let y be the label of time $t + s$ (s is steps), we can train a model with prediction ability. We provide an ELBD framework for improving detection accuracy, robustness, and predicting anomalies. Experimental results can be seen next.

Experiments and results

Experimental settings

We design two experiments to evaluate the performance of the ELBD framework and compare them with results in Base performance anomaly detection methods section.

- Performance of methods in the ELBD framework. To evaluate the improvement in detection accuracy and algorithm robustness, we compare the performance of methods in the ELBD framework with the best-performing base detection method. Experiment results can be seen in E1.
- Multi-step prediction of the deep ensemble method. As for the deep ensemble method, we evaluate its multi-step prediction ability, which can be seen in E2.

No hyper-parameter exists for maximum, average, and weighted average ensemble methods. We first do the

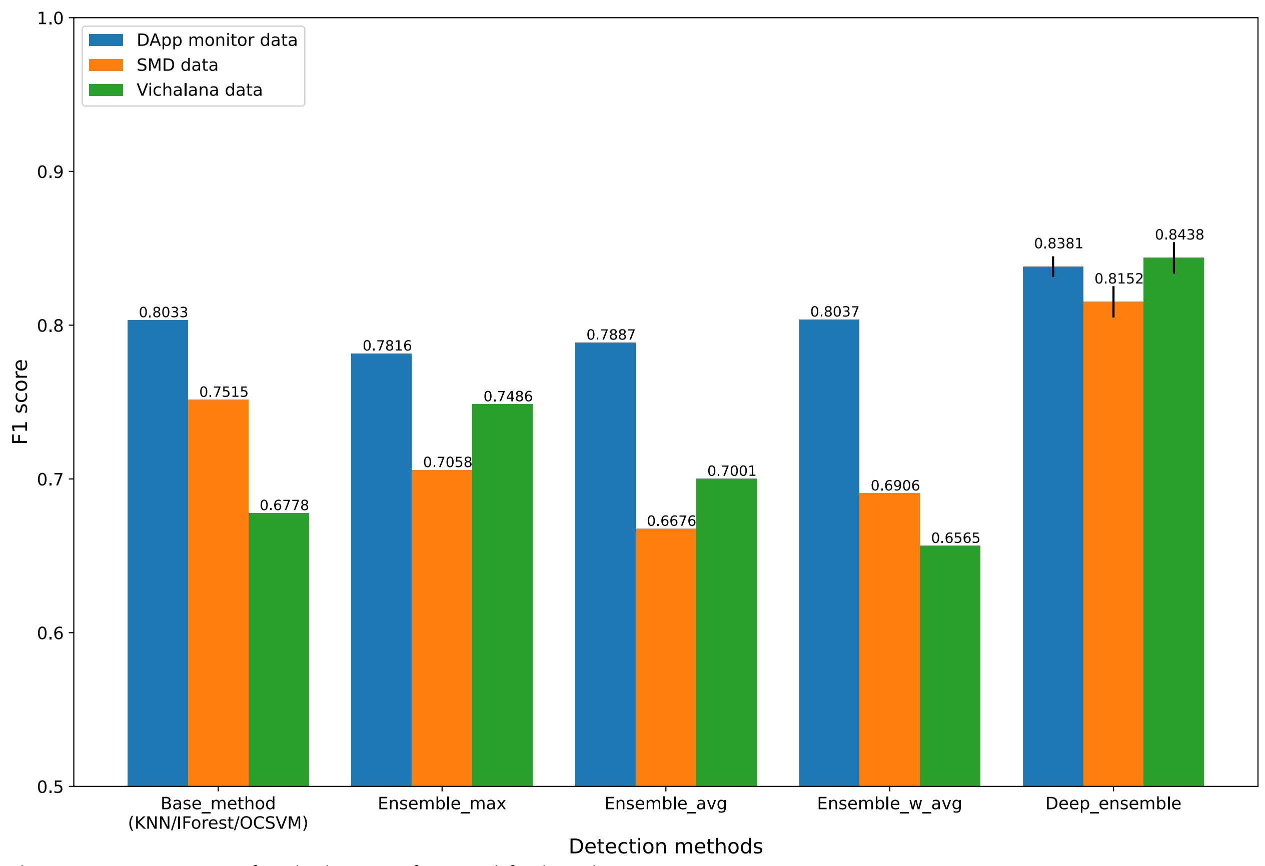

Fig. 4 Detection accuracy of methods in ELBD framework for three datasets

train/test split for the deep ensemble method. Because there are fewer labels in real scenarios, we use only 10% of data with labels to train the model. Next, hyper-parameters in the MLP for the three datasets are the same. The input layer has 4 neurons because we have 4 base methods. In addition, we set 20 neurons in the two hidden layers and the output layer as 1. We train 100 epochs and set the batch size to 20. We use the Adam optimizer for stochastic gradient descent with an initial learning rate of 10^{-3} during model training. We train the deep ensemble method 10 times. We show the error bar in figures and take the average of evaluation metrics in tables, such as F1 score and time, as the final result.

Experimental results

E1: Performance of methods in ELBD framework. We provide different methods in the ELBD framework to improve detection performance. We apply these methods to the DApp monitoring, SMD, and Vichalana datasets to evaluate them. We compare these methods with the best-performing base method and evaluate the detection accuracy and robustness.

For the DApp monitoring data in Fig. 4, we can see that the F1 score of the weighted average ensemble is higher than KNN, maximum, and average ensemble, which shows that ensemble methods can improve the detection accuracy by integrating extracted information of base methods. In addition, the weighted average ensemble assigns weights to base methods to highlight their different contributions. The most noteworthy thing in Fig 4 is that the deep ensemble method has the highest F1 score, 0.8381. We train the deep ensemble method with only 10% labels, but the improvement is significant. The result shows that the nonlinear combination of base methods can extract more information and help improve detection accuracy. As for time spent, in Fig. 5, we can see that the deep ensemble method spends about 0.9s for data testing, and other ensemble methods spend about 0.8s. Time spent on each method for the DApp monitoring data is under 1s, which is not high overall.

For SMD data in Fig. 4, we can see that the F1 score of the IForest is 0.7515, which is higher than the maximum, average, and weighted average ensemble methods. Ensemble methods rely heavily on base methods, and other base methods (KNN, LOF, and OCSVM) perform

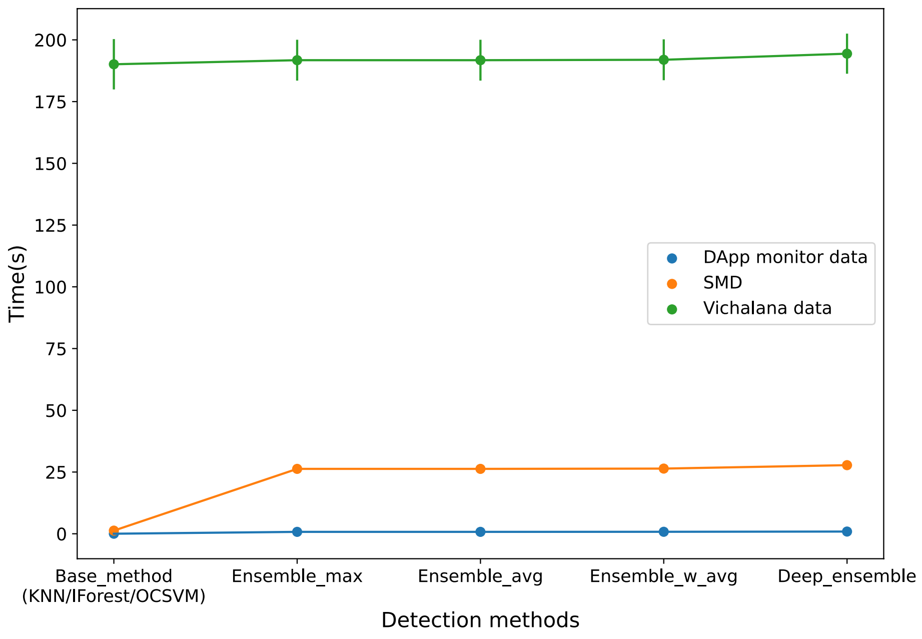

Fig. 5 Time spent of methods in ELBD framework for three datasets

Table 6 Rank results of algorithm robustness

Method	IForest	KNN	LOF	OCSVM	Emsemble_max	Ensemble_avg	Ensemble_w_avg	Deep_ ensemble
DApp monitoring data	4	3	8	7	6	5	2	**1**
SMD	2	7	8	6	3	5	4	**1**
Vichalana data	5	7	8	4	2	3	6	**1**
Average rank	3.7	5.7	8	5.7	3.7	4.3	4	**1**
Robustness score	0.6143	0.3286	0	0.3286	0.6143	0.5286	0.5714	**1**

poorly. The most important thing is that the deep ensemble has the best F1 score, 0.8152, which is much higher than other methods, showing its superior detection ability by integrating information non-linearly. Figure 5 presents the time spent of these methods. We can see that the maximum, average, and weighted average ensemble spend about 26.2s, and the deep ensemble spends about 27.8s. Still, ensemble methods rely on base methods, so their time spent is mainly because of the kernel function calculation in OCSVM and the computational cost of the neural network.

For Vichalana data in Fig. 4, we can see that the F1 scores of the maximum and average ensembles are higher than OCSVM, which shows the detection performance improvement of ensemble-based methods. In contrast, the weighted average ensemble does not assign weights well. In addition, the deep ensemble has the best F1 score, 0.8438, which greatly improves detection accuracy compared with other methods, and it shows the advantages of the non-linear combination of base methods. Figure 5 presents the time spent of these methods. We can see that the maximum, average, and weighted average ensemble spend about 190s, and the deep ensemble spends about 194s. The time spent is still mainly because the large-scale data makes the kernel function calculation in OCSVM time-consuming. In addition, the neural network's computational cost takes a little time.

As for algorithm robustness, we provide rank results in the Table 6. We rank the detection accuracy of all methods, including base methods and methods in the ELBD framework, and calculate their average rank and robustness score, respectively. In the Table 6, we can see that the deep ensemble method has the best detection accuracy on the three different datasets, the DApp monitoring data, SMD, and Vichalana data, which shows that it has not only superior detection accuracy but outstanding robustness for different data distributions. Other ensemble methods have good robustness compared with base

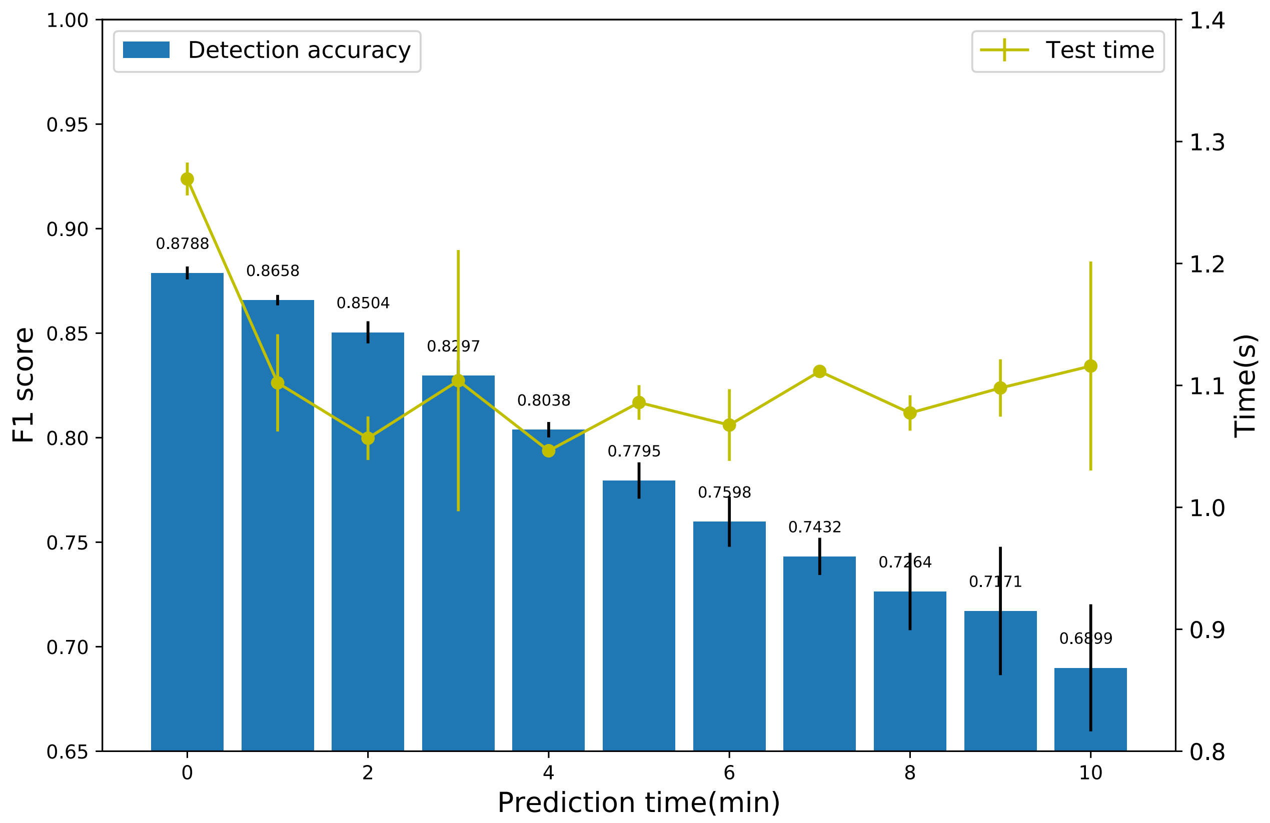

Fig. 6 Prediction accuracy and time spent for different time steps of the deep ensemble method on the DApp monitoring data

Table 7 Comparison results of all detection methods

Challenge	Indicator	IForest	KNN	LOF	OCSVM	Emsemble_max	Ensemble_avg	Ensemble_w_avg	Deep_ensemble
Detection accuracy	F1 score	0.7335	0.6422	0.5246	0.6732	0.7453	0.7188	0.7169	**0.8324**
Algorithm robustness	Robustness score	0.6143	0.3286	0	0.3286	0.6143	0.5286	0.5714	**1**
Multi-step prediction	Prediction score	-	-	-	-	-	-	-	**3.3497**
ARP_score		1.3478	0.9708	0.5246	1.0018	1.3596	1.2474	1.2883	**5.1821**

detection methods. In contrast, base methods show performance inconsistency, except for IForest. IForest has quite good robustness compared with other base detection methods. In conclusion, we can say that methods in the ELBD framework improve detection performance in terms of detection accuracy and robustness, especially the deep ensemble method.

E2: Multi-step prediction of the deep ensemble method. With the deep ensemble method, we can predict multi-step performance anomalies. We mainly test its prediction ability on the DApp monitoring data. The time interval in the DApp monitoring data is 15s. Thus, we can use every 4 steps, which is 1 minute, as the prediction step. Then, we predict whether the anomaly will happen or not after one or two or three minutes. To evaluate the prediction ability, we present the prediction accuracy with the F1 score in Fig. 6.

In Fig. 6, we can see that the longer the prediction time, the lower the detection accuracy, which means that it is difficult to predict long-term anomalies because dependency between data diminishes over time. In addition, we can see that all F1 scores are higher within four minutes than 0.8, which is good detection accuracy. Therefore, we can say that it is available for the deep ensemble to predict anomalies in the next four minutes with high accuracy. We also show the time spent testing the prediction ability in Fig. 6. We can see that the testing time is around 1.1s, meaning that the deep ensemble method can predict anomalies quickly.

For all the detection methods, we provide a Table 7 to compare their performance in terms of detection accuracy, algorithm robustness, and multi-step prediction. In the table, we can see that neither base detection nor linear ensemble methods have prediction ability. In addition, we can notice that IForest and weighted average ensemble methods have good detection accuracy and robustness. The most important thing is that the deep ensemble method perfectly addresses three challenges and has the highest *ARP_score* 5.1821, which is much better than other methods.

In conclusion, we provide the performance evaluation of ensemble methods in the ELBD framework. Our experiments show that these methods improve detection accuracy and robustness by integrating extracted information from base methods. Among those, the deep ensemble method has superior detection performance in terms of accuracy, robustness, and multi-step prediction. In addition, results show that the deep ensemble method can predict anomalies in the next four minutes with high accuracy.

Discussion

This paper provides an ELBD framework for performance anomaly detection and prediction of cloud applications. They are developed based on four base methods to improve detection performance. Our experiments evaluate the performance of methods in the ELBD framework and show an improvement in detection accuracy, robustness, and multi-step prediction ability. However, some aspects of these methods and experiments in this paper can still be improved.

For noise in monitoring data, we first provide feature extraction for pre-processing data. We use PCA to filter features and reduce data dimensions. The PCA is a general feature extraction method that can easily be used on many datasets and improve detection efficiency. However, PCA has some limitations, like assuming features in data are linearly dependent. Therefore, other feature extraction methods like AutoEncoder [46] can be considered in the future.

Our experiments show that the four base detection methods' performances vary on three datasets. The performance inconsistency is because each method extracts different features from the data. Moreover, the outputs of these base methods are assembled as the following methods' inputs, which will severely affect detection performance. In this paper, we manually select the four base detection methods based on their differences. However, a method to automatically select suitable base detection methods while considering data distribution can be researched in the future.

The capacity of the deep ensemble can be tested further. In our experiments, the deep ensemble is trained with fewer labels and has outstanding performance compared with other detection methods. Next, we can test the effects of different numbers of labels. Also, we can consider replacing the MLP with other deep neural networks like LSTM [47] to improve detection accuracy.

Performance anomaly detection methods can be applied to other monitoring data, such as blockchain-level data in DApps. Furthermore, based on performance anomaly detection, root cause analysis can be researched in the future to localize root causes of performance anomalies. For example, when application response time is high, we need to determine the root causes of cloud resource problems or service-level delays.

Conclusions and future work

This paper focuses on performance anomaly detection and prediction of cloud applications, which need to satisfy three challenging requirements: high detection accuracy, robustness, and multi-step prediction. Based on our survey, many machine learning-based methods have been developed for performance anomaly detection. However, these detection methods have inconsistent performance for different datasets and rarely simultaneously solve the three requirements. Therefore, based on existing performance anomaly detection methods, we provide an ELBD framework that integrates existing detection methods to address the three requirements.

We first apply four base detection methods (IForest, KNN, LOF, OCSVM) to study the monitoring data characteristics. The results show that these base methods perform differently on datasets with different data patterns. Then, based on these methods, we develop an ELBD framework (maximum, average, weighted average, and deep ensemble) that integrates existing detection methods for improving detection performance. Our experiments show that methods in the ELBD framework significantly improve detection accuracy and robustness, especially the deep ensemble method. In addition, the deep ensemble method has the multi-step prediction ability, which can predict anomalies in the next four minutes with high accuracy. We also evaluate detection performance with our indicator, and the results show that the deep ensemble method has the highest *ARP_score* 5.1821, which is much better than other methods.

This paper provides an ensemble-based framework for performance anomaly detection of cloud applications, and the results show that the AI-based deep ensemble method has superior performance in terms of detection accuracy, robustness, and prediction ability. However, some aspects of this research can still be improved. For example, we can perform feature selection for multivariate monitoring data, and more experiments and extensions for deep ensemble methods can be researched in the future. In addition, for applying AI methods to help operators and developers better implement performance management of cloud applications, several future research directions can be discussed based on [48].

Data security. For a running cloud application, large-scale monitoring data is collected, which makes it necessary to consider implementing secure data governance for collected data. Collected performance data is mostly stored in centralized or distributed environments, with a high risk of being attacked or stolen [49]. Blockchain-based data storage has been developed recently in IoT [50]. However, blockchain-based storage technologies still have challenges such as durability, availability, and cost, which need to be explored more in the future.

Data labeling. High-quality labeled data can be very helpful in improving detection accuracy. However, there are fewer labels in real scenarios, and labeling data manually is onerous and time-consuming. Nowadays, active learning [51] has been developed to solve label issues by combining both machine and human labor. Therefore, we consider that automated data annotation methods based on active learning can be explored more in the future, for example, by reducing human labor and improving the quality of labels.

Detection efficiency. Except for model robustness and accuracy, efficiency is important for detection methods to meet users' requirements considering a large number of performance data exists. Machine learning methods, especially deep learning methods, usually have high detection accuracy but time-consuming model training [52]. Only a few statistical-based methods for improving detection efficiency have been developed [53]. Therefore, improving the model efficiency and achieving accurate real-time online detection is worth exploring in the future.

Model explainability. For detected anomalies, it is natural to explore why these anomalies happen. Explainable AI [54] has been researched for deep learning models, which are typically viewed as black boxes. Self-explainable methods like IForest have been explored in this paper. However, the explanation of the deep ensemble method can be explored more in the future. In addition, root cause localization to identify metrics that cause anomalies should be investigated more in the future despite complex dependencies between metrics.

Acknowledgements
Not applicable.

Authors' contributions
Ruyue Xin put forward the main ideas, designed experiments, and wrote the manuscript. Hongyun Liu conducted experiments. Peng Chen guided ideas and experiments. Zhiming Zhao guided the research, and provided comments for the writing. All authors read and approved the final manuscript.

Funding
This research is funded by the EU Horizon 2020 research and innovation program under grant agreements 825134 (ARTICONF project), 862409 (BlueCloud project), 824068 (ENVRIFAIR project). The research is partially supported by EU LifeWatch ERIC. The research is also funded by Science and Technology Program of Sichuan Province under Grant 2020JDRC0067 and 2020YFG0326.

Availability of data and materials
The data and materials are available from the corresponding author on reasonable request.

Declarations

Ethics approval and consent to participate
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 26 May 2022 Accepted: 23 December 2022
Published online: 14 January 2023

References

1. Cid-Fuentes JA, Szabo C, Falkner K (2018) Adaptive performance anomaly detection in distributed systems using online svms. IEEE Trans Dependable Secure Comput 17(5):928–941
2. Borghesi A, Bartolini A, Lombardi M, Milano M, Benini L (2019) A semisupervised autoencoder-based approach for anomaly detection in high performance computing systems. Eng Appl Artif Intell 85:634–644
3. Zhu M, Ye K, Xu CZ (2018) Network anomaly detection and identification based on deep learning methods. In: International Conference on Cloud Computing. Springer, Cham, 219–234
4. Siffer A, Fouque PA, Termier A, Largouet C (2017) Anomaly detection in streams with extreme value theory. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17). Association for Computing Machinery, New York, 1067–1075
5. Hu M, Ji Z, Yan K, Guo Y, Feng X, Gong J, Zhao X, Dong L (2018) Detecting anomalies in time series data via a meta-feature based approach. IEEE Access 6:27760–27776
6. Breunig MM, Kriegel HP, Ng RT, Sander J (2000) Lof: identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD international conference on Management of data (SIGMOD '00). Association for Computing Machinery, New York, 93–104
7. Audibert J, Michiardi P, Guyard F, Marti S, Zuluaga MA (2020) Usad: Unsupervised anomaly detection on multivariate time series. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 3395–3404
8. Zhang H, Yu Y, Jiao J, Xing E, El Ghaoui L, Jordan M (2019) Theoretically principled trade-off between robustness and accuracy. In: Proceedings of the 36th International Conference on Machine Learning (ICML), Long Beach, California, USA, Vol. 97 of Proceedings of Machine Learning Research, PMLR. 7472–7482
9. Wu W, He L, Lin W, Su Y, Cui Y, Maple C, Jarvis SA (2020) Developing an unsupervised real-time anomaly detection scheme for time series with multi-seasonality. IEEE Trans Knowl Data Eng 34(9):4147–4160
10. Ibidunmoye O (2017) Performance anomaly detection and resolution for autonomous clouds. PhD thesis, Umeå University

11. Ibidunmoye O, Hernández-Rodriguez F, Elmroth E (2015) Performance anomaly detection and bottleneck identification. ACM Comput Surv (CSUR) 48(1):1–35
12. Qi GJ, Luo J (2020) Small data challenges in big data era: A survey of recent progress on unsupervised and semi-supervised methods. IEEE Trans Pattern Anal Mach Intell 44(4):2168–2187
13. Su Y, Zhao Y, Niu C, Liu R, Sun W, Pei D (2019) Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19). Association for Computing Machinery, New York, 2828–2837
14. Tang J, Chen Z, Fu AWC, Cheung DW (2002) Enhancing effectiveness of outlier detections for low density patterns. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, Berlin, Heidelberg, 535–548
15. Papadimitriou S, Kitagawa H, Gibbons PB, Faloutsos C (2003) Loci: Fast outlier detection using the local correlation integral. In: Proceedings 19th international conference on data engineering (Cat. No. 03CH37405). IEEE, pp 315–326
16. Ramaswamy S, Rastogi R, Shim K (2000) Efficient algorithms for mining outliers from large data sets. In: Proceedings of the 2000 ACM SIGMOD international conference on Management of data (SIGMOD '00). Association for Computing Machinery, New York, pp 427–438
17. Zhang K, Hutter M, Jin H (2009) A new local distance-based outlier detection approach for scattered real-world data. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. Lecture Notes in Computer Science, vol 5476. Springer, Berlin, Heidelberg, 813–822
18. Schölkopf B, Platt JC, Shawe-Taylor J, Smola AJ, Williamson RC (2001) Estimating the support of a high-dimensional distribution. Neural Comput 13(7):1443–1471
19. Song Q, Hu W, Xie W (2002) Robust support vector machine with bullet hole image classification. IEEE Trans Syst Man Cybern C (Appl Rev) 32(4):440–448
20. Liu FT, Ting KM, Zhou ZH (2008) Isolation forest. In: 2008 eighth ieee international conference on data mining. IEEE, pp 413–422
21. Sakurada M, Yairi T (2014) Anomaly detection using autoencoders with nonlinear dimensionality reduction. In: Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis (MLSDA'14). Association for Computing Machinery, New York, 4–11
22. Kingma DP, Welling M (2013) Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114
23. Yuan X, He P, Zhu Q, Li X (2019) Adversarial examples: Attacks and defenses for deep learning. IEEE Trans Neural Netw Learn Syst 30(9):2805–2824
24. Hashemi MJ, Keller E (2020) Enhancing robustness against adversarial examples in network intrusion detection systems. In: 2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN). Leganes. 37–43
25. Galicia A, Talavera-Llames R, Troncoso A, Koprinska I, Martínez-Álvarez F (2019) Multi-step forecasting for big data time series based on ensemble learning. Knowl-Based Syst 163:830–841
26. Zhou ZH (2012) Ensemble methods: foundations and algorithms. Chapman and Hall/CRC, New York
27. Aggarwal CC, Sathe S (2015) Theoretical foundations and algorithms for outlier ensembles. ACM sigkdd Explor Newsl 17(1):24–47
28. Tyralis H, Papacharalampous G, Langousis A (2021) Super ensemble learning for daily streamflow forecasting: Large-scale demonstration and comparison with multiple machine learning algorithms. Neural Comput Applic 33(8):3053–3068
29. Tama BA, Nkenyereye L, Islam SR, Kwak KS (2020) An enhanced anomaly detection in web traffic using a stack of classifier ensemble. IEEE Access 8:24120–24134
30. Adeyemo VE, Abdullah A, JhanJhi N, Supramaniam M, Balogun AO (2019) Ensemble and deep-learning methods for two-class and multi-attack anomaly intrusion detection: an empirical study. Int J Adv Comput Sci Appl 10(9):520–528
31. Wei C, Sohn K, Mellina C, Yuille A, Yang F (2021) Crest: A class-rebalancing self-training framework for imbalanced semi-supervised learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), virtual. Computer Vision Foundation/IEEE. 10857–10866
32. Jian C, Ao Y (2022) Imbalanced fault diagnosis based on semi-supervised ensemble learning. J Intell Manuf 34:1–16
33. Fred AL, Jain AK (2005) Combining multiple clusterings using evidence accumulation. IEEE Trans Pattern Anal Mach Intell 27(6):835–850
34. Huang D, Lai JH, Wang CD (2015) Robust ensemble clustering using probability trajectories. IEEE Trans Knowl Data Eng 28(5):1312–1326
35. Huang D, Lai J, Wang CD (2016) Ensemble clustering using factor graph. Pattern Recog 50:131–142
36. Ünlü R, Xanthopoulos P (2019) A weighted framework for unsupervised ensemble learning based on internal quality measures. Ann Oper Res 276(1):229–247
37. Yang J, Zhang D, Frangi AF, Jy Yang (2004) Two-dimensional pca: a new approach to appearance-based face representation and recognition. IEEE Trans Pattern Anal Mach Intell 26(1):131–137
38. Abdi H, Williams LJ (2010) Principal component analysis. Wiley Interdiscip Rev Comput Stat 2(4):433–459
39. Freund Y, Mason L (1999) The alternating decision tree learning algorithm. In: Proceedings of the Sixteenth International Conference on Machine Learning (ICML '99). Morgan Kaufmann Publishers Inc., San Francisco, 124–133
40. Saurabh N, Rubia C, Palanisamy A, Koulouzis S, Sefidanoski M, Chakravorty A, Zhao Z, Karadimce A, Prodan R (2021) The articonf approach to decentralized car-sharing. Blockchain Res Appl 2(3):100013
41. Geethika D, Jayasinghe M, Gunarathne Y, Gamage TA, Jayathilaka S, Ranathunga S, Perera S (2019) Anomaly detection in high-performance api gateways. In: 2019 International Conference on High Performance Computing & Simulation (HPCS). Dublin. 995–1001
42. Shin K, Fernandes D, Miyazaki S (2011) Consistency measures for feature selection: a formal definition, relative sensitivity comparison and a fast algorithm. In: Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI'11). AAAI Press, Barcelona, 1491–1497
43. Chabchoub Y, Togbe MU, Boly A, Chiky R (2022) An in-depth study and improvement of isolation forest. IEEE Access 10:10219–10237
44. Saranya C, Manikandan G (2013) A study on normalization techniques for privacy preserving data mining. Int J Eng Technol (IJET) 5(3):2701–2704
45. Kuncheva LI, Whitaker CJ (2003) Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Mach Learn 51(2):181–207
46. Wang Y, Yao H, Zhao S (2016) Auto-encoder based dimensionality reduction. Neurocomputing 184:232–242
47. Sundermeyer M, Schlüter R, Ney H (2012) Lstm neural networks for language modeling. In: Thirteenth annual conference of the International Speech Communication Association. Portland, ISCA, 194–197
48. Gill SS, Xu M, Ottaviani C, Patros P, Bahsoon R, Shaghaghi A, Golec M, Stankovski V, Wu H, Abraham A et al (2022) AI for next generation computing: Emerging trends and future directions. Internet Things 19:100514, Piscataway
49. Shah M, Shaikh M, Mishra V, Tuscano G (2020) Decentralized cloud storage using blockchain. In: 2020 4th International conference on trends in electronics and informatics (ICOEI)(48184). IEEE, pp 384–389
50. Li R, Song T, Mei B, Li H, Cheng X, Sun L (2018) Blockchain for large-scale internet of things data storage and protection. IEEE Trans Serv Comput 12(5):762–771
51. Ren P, Xiao Y, Chang X, Huang PY, Li Z, Gupta BB, Chen X, Wang X (2021) A survey of deep active learning. ACM Comput Surv (CSUR) 54(9):1–40
52. Ren H, Xu B, Wang Y, Yi C, Huang C, Kou X, Xing T, Yang M, Tong J, Zhang Q (2019) Time-series anomaly detection service at microsoft. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19). Association for Computing Machinery, New York, 3009–3017
53. Li J, Di S, Shen Y, Chen L (2021) Fluxev: A fast and effective unsupervised framework for time-series anomaly detection. In: Proceedings of the 14th ACM International Conference on Web Search and Data Mining (WSDM '21). Association for Computing Machinery, New York, 824–832
54. Xu F, Uszkoreit H, Du Y, Fan W, Zhao D, Zhu J (2019) Explainable AI: A brief survey on history, research areas, approaches and challenges. In: CCF international conference on natural language processing and Chinese computing. Springer, Cham, 563–574

Publisher's Note