A Novel Adaptive Undersampling Framework for Class-Imbalance Fault Detection

Min Qian, *Student Member, IEEE*, and Yan-Fu Li, *Senior Member, IEEE*

***Abstract*—Class-imbalance is a prevalent and challenging problem in the field of fault detection. The undersampling ensemble framework is an effective method to deal with imbalance problems. However, designing a suitable sampling strategy to generate effective and divergent subsets is a major difficulty of this type of method. Hence, we propose a novel adaptive undersampling framework. It models the entire training process as a Markov decision process (MDP), thus, enabling dynamic decision-making for subsequent sampling strategies based on the current training performance of the ensemble framework. The sampler is optimized by the soft actor-critic reinforcement learning method. Considering the imbalance dataset's nature and the need for state definition, the clustering method is applied to the original training dataset. The state (training performance) and the action (sampling strategy) are determined according to the clustering results. The unique state definition and sampling decision mechanism are designed to ensure the convergence speed of MDP and improve the divergence of the subsets. We validate the effectiveness of the proposed framework on the real-world wind turbine blade cracking datasets and the high-speed train braking system dataset. The experimental results show that the classification performance and robustness of the proposed framework are significantly better than the 16 benchmark methods.**

***Index Terms*—Class-imbalance, cluster analysis, fault detection, reinforcement learning, undersampling ensemble.**

I. Introduction

DATA-DRIVEN fault detection methods enable practitioners to fully utilize the history and current data of the industrial systems, which helps to reduce the reliance on domain experience and expert knowledge [1]. Advancements in sensor technology and other data collection techniques have further enhanced the practical value of data-driven methods [2]. However, there are still many obstacles and challenges in the practical implementation of data-driven fault detection methods [3]. A major challenge is the rare occurrence of critical failure in modern industrial systems. Consequently, it has become more difficult for researchers to collect sufficient faulty samples among the booming volume of normal operation data. Thus, the class-imbalance turns out to be a common phenomenon in the datasets for fault detection [4], [5], [6], [7]. In machine learning, this type of problem is referred to as imbalanced classification problems, that is, the number of samples in some classes is significantly larger than that in other classes.

Since most classification methods are modeled on the premise of class-balance, they usually perform poorly on imbalanced datasets [8]. Especially when the imbalance ratio is high, the traditional classification method tend to divide all samples into the majority class. Researchers have developed several methods for imbalanced datasets at different stages of fault detection [9]. In the data preprocessing stage, the imbalanced dataset can be balanced through undersampling or oversampling. In the feature extraction stage, neural network or feature adaptation can be designed to directly learn fault features from limited fault samples. In the training stage, developing a imbalanced classifier and applying ensemble learning theory can improve classification performance. A more detailed review of the class-imbalance methods are summarized in Section II.

According to [10], oversampling can easily lead to overfitting problems, especially for highly imbalanced datasets. It is challenging to use limited minority samples to generate a large number of synthetic samples that conform to the true distribution, which often results in limited diversity of synthetic samples and leads to overfitting problems. In contrast, although the undersampling faces the potential information loss of the majority class, it can be effectively alleviated by combining integrated learning and mechanism design. Existing studies have proved the superiority of the undersampling ensemble framework in dealing with imbalanced datasets [11], thus, this article focuses on developing an undersampling ensemble framework to solve the class-imbalance problem.

A good ensemble framework requires each base classifier to vary as much as possible while ensuring classification accuracy [12]. The diversity mainly comes from the sampled training subsets, thus, designing an intelligent sampler is the key to this framework. Among the existing undersampling ensemble frameworks, the diversity in subsets of bagging-based methods such as UnderBagging entirely relies on random sampling. Although RUSBoost, Cascade, etc. [13], give larger sampling weight to the previously misclassified samples, they ignore the overfitting problem caused by this. All these existing frameworks cannot dynamically adjust the sampling strategy according to the current training state, making it difficult to take full advantage of the ensemble. Hence, it is desired to

Manuscript received 3 March 2022; revised 27 May 2022 and 4 August 2022; accepted 11 October 2022. Date of publication 25 October 2022; date of current version 1 September 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 71731008, and in part by the Natural Science Foundation of Beijing Municipality under Grant L191022. Associate Editor: Ed Pohl. *(Corresponding author: Yan-Fu Li.)*

The authors are with the Department of Industrial Engineering, Tsinghua University, Beijing 100084, China (e-mail: qm19@mails.tsinghua.edu.cn; liyanfu@tsinghua.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TR.2022.3214519.

Digital Object Identifier 10.1109/TR.2022.3214519

design an undersampling ensemble framework can dynamically adjust the sampling strategies according to the current ensemble classifiers' performance. The sampler should have learning and decision-making capabilities to handle complex industrial imbalanced datasets in the real world.

To achieve the above goals, the core problem is the parameterized definition of the ensemble classifiers state and the optimization of the sampler. We propose a novel adaptive undersampling framework. First, we cluster the original dataset. Samples in the same cluster have similar information. The clustering result can provide more comprehensive information to the sampler. At the same time, the state of the current ensemble classifier can also be defined from the perspective of clusters. Compared with the definition from the individual perspective, it can reduce the dimensionality of the state while avoiding information loss as much as possible, which is crucial for optimizing the sampler. Each cluster adopts a different sampling strategy based on its imbalanced ratio. Diverse sampling strategies are beneficial to increasing the diversity of training subsets.

We introduce reinforcement learning to solve the nondifferentiable optimization problem of the sampler. According to the classification error of the current ensemble classifier, the current state can be defined as the input, and the decision is made to determine the sampling strategy for the next stage. Based on the decision and the clustering result, the sampler selects a subset to train a new base classifier and adds it to the ensemble. This constitutes a complete iteration, thereby establishing a Markov decision process (MDP) model. The decision of reinforcement learning is not defined as the selected subset but as the value of a crucial parameter in the sampling strategy. The advantage is that the dimensionality of the decision space is greatly reduced, the convergence speed of reinforcement learning is improved, and the optimization hardness is largely resolved. The cluster-based state definition also reduces the dimensionality of the state space, which further enhances this advantage.

To verify the ability of the proposed framework to handle actual industrial tasks, we conducted experiments on two real-world datasets—a wind turbine blade cracking fault detection task and a high-speed train brake system fault detection task. The main contributions of this article can be summarized as the following four points.

1) This article introduces reinforcement learning theory to model the entire undersampling ensemble framework as an MDP, making the sampler based on rules and dynamically adjusted according to the nature of the dataset and the current training performance.
2) The proposed framework designs a cluster-based state definition, which reduces the state space dimension and the optimization difficulty of the sampler. Each cluster can adopt different sampling strategies according to its imbalanced ratio to increase the diversity of sampling subsets.
3) The sampler is optimized by a soft actor-critic (SAC) reinforcement learning method. Sampling fully considers the sampling information of individuals and clusters and proposes a weighted sampling strategy based on the Gaussian function. This strategy simplifies the output of MDP and improves the convergence speed.
4) The experimental results on two real-world industrial datasets prove that the classification performance and robustness of the proposed framework are significantly better than the 16 existing methods.

The rest of this article is organized as follows. Section II reviews the existing imbalanced classification methods. Section III describes the proposed adaptive undersampling ensemble framework in detail. In Section IV, the proposed method is validated on two real-world datasets. Finally, Section V concludes this article.

II. Related Work

Several methods are proposed to address the imbalanced classification, which can be divided into four categories, including resampling methods, feature learning approaches, classifier design approaches, and ensemble learning. We have summarized them in Fig. 1 for easy understanding.

Resampling methods include oversampling which generates new minority samples and undersampling which removes certain majority samples. The most widely used oversampling methods are the synthetic minority over-sampling technique (SMOTE) and its modified variants [14]. These methods generate new samples through the convex combination of the minority samples, which might change the training set's marginal distribution and make the classification performance worse. Recently, researchers have tried to use deep learning methods such as generative adversarial networks (GAN) [15] and variational autoencoder (VAE) [16] to generate new samples. However, when the minority samples are absolutely scarce, these methods often face problems such as difficulty in model training and poor quality of generated samples. Undersampling methods can be divided into distance-based, clustering-based, and evolutionary-based [17]. The difference between these methods is the selection strategy for the majority samples, and there is no way to solve the problem of limited information of minority under the highly imbalanced. If the minority information of training set is very limited (minority samples are absolutely rare), the undersampling methods often perform poorly.

Feature learning approaches attempt to learn effective fault features for classification. Deep neural networks (DNN) such as autoencoder and convolutional neural networks (CNN) show strong feature extraction capabilities, which can significantly improve the classification performance [18]. Nevertheless, the DNN presents difficulties in training and parameter optimization. Many transfer learning methods have been applied to solve the problem of sparse fault samples, including transfer component analysis (TCA) [19], joint distribution adaptation (JDA) [20], and deep neural networks (DNN) [21]. However, constructing an auxiliary transferable dataset is also a big challenge in engineering scenarios.

Classifier design approaches focus on the calibration of the main classifiers, e.g., support vector machine (SVM) [22], for the imbalanced classification problem. A common practice is to give the classifier different penalties for different types of

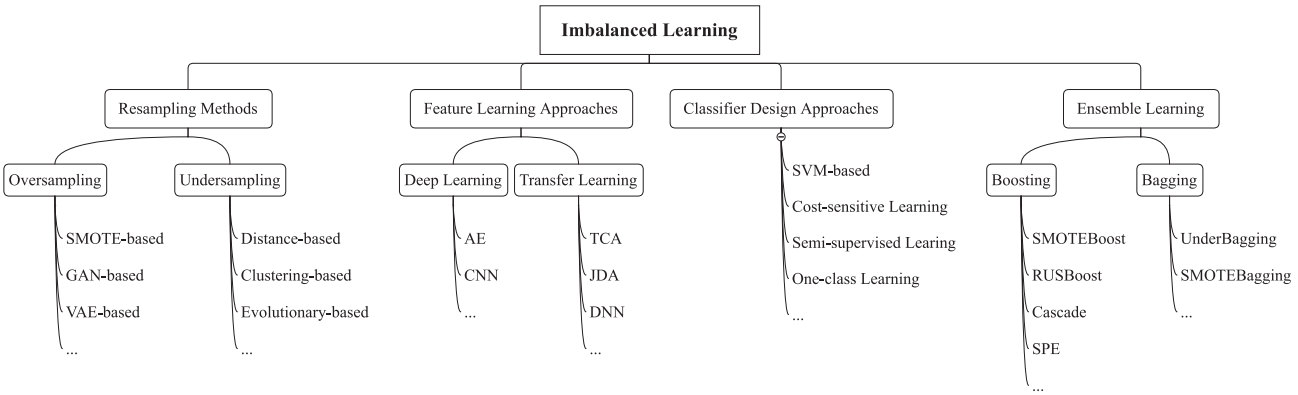

Fig. 1. Overview of the existing imbalanced learning methods.

misclassification, also known as cost-sensitivity [23]. But the determination of the penalty coefficient is a big problem. A new attempt is adding some unlabeled samples into the training set to assist training and use semisupervised learning to balance the training set [24]. However, the addition of unlabeled samples does not always improve the classification performance and sometimes even damages the classification boundary to make the result worse. Different from the conventional classifiers, which are trained on two or more classes, one-class learning is trained upon only one class sample [25]. It can circumvent the problem of class-imbalance. However, learning the separable boundary by one-class learning is generally difficult when overlapping patterns are present in certain feature subspaces.

Ensemble learning seeks to obtain more reliable classification results by combining several base classifiers. There are mainly two types of ensemble methods, bagging and boosting. Bagging obtains multiple subtraining sets through bootstrap sampling, including: UnderBagging [26] and SMOTEBagging, etc [27]. Boosting performs sequential sampling by increasing the sampling weight of the current misclassified samples, including: SMOTEBoost [28], RUSBoost [29], Cascade [13], SPE [30], MESA [31], etc. It is generally combined with resampling methods or imbalance-designed classifiers because ensemble learning cannot solve class-imbalance problems directly.

III. Novel Adaptive Undersampling Framework

Assuming a binary classification problem, there is a imbalanced dataset $\mathcal{S}$: $\{\boldsymbol{X}, \boldsymbol{y}\} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}$, where $\boldsymbol{x} \in \mathbb{R}^d$, $y \in \{0, 1\}$ and d is the input feature dimensions. The dataset can be divided into the majority set $\mathcal{N}$: $\{(\boldsymbol{x}, y)|y = 0, (\boldsymbol{x}, y) \in \mathcal{S}\}$ and the minority set $\mathcal{P}$: $\{(\boldsymbol{x}, y)|y = 1, (\boldsymbol{x}, y) \in \mathcal{S}\}$, $|\mathcal{N}| \gg |\mathcal{P}|$,[1] $\mathcal{S} = \mathcal{P} \cup \mathcal{N}$. Our purpose is to learn a final classifier $F : \mathbb{R}^d \to \{0, 1\}$ based on the $\boldsymbol{X} \to \boldsymbol{y}$ mapping.

A. Undersampling Ensemble Framework

Ensemble learning has been widely used in imbalanced classification problems. Studies such as [27] and [32] have proved that a multiclassifier ensemble is easier to achieve excellent generalization ability in imbalanced datasets than a single classifier. In particular, the combination of undersampling methods and ensemble learning, such as the classic RUSBoost, Cascade to the latest SPE, MESA, etc., have achieved excellent performance on imbalance problems. The basic process of this type of framework is shown in Fig. 2. Different frameworks will use different undersampling strategies to divide the original imbalanced dataset into several balanced subsets. Then use each subset to train the corresponding base classifiers. In the final test stage, the classification results of all base classifiers are summarized by the ensemble strategy to obtain the final classification result. The existing theory proves that the prerequisite for ensemble learning to achieve good performance is that the base classifier has the characteristics of "good but different" [12], [33]. Intuitively, if the performance of the base classifier is too poor, the performance of the final ensemble will be degraded. On the other hand, if the classification performance of the base classifier is the same, the result of the ensemble will be the same, and such an ensemble has no meaning. This means that in addition to ensuring the classification accuracy (bias), each base classifier should also have diversity (variance). There is a conflict between the "accuracy" and "diversity" of the base classifier. Assuming the samples' label is accurate, the generalization error on the test set $\{\boldsymbol{X}, \boldsymbol{y}\}$ of the classifier f trained on $\mathcal{D}$ can be decomposed

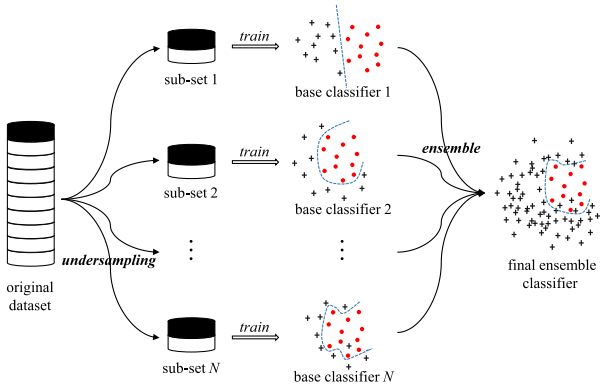

Fig. 2. Basic process of undersampling ensemble framework.

[1]To avoid confusion, we use $|\cdot|$ and $abs(\cdot)$ to represent cardinality and absolute value, respectively.

into

$$E(f;\mathcal{D}) = \mathbb{E}_{\mathcal{D}}\left[\left(f(\boldsymbol{X};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[f(\boldsymbol{X};\mathcal{D})]\right)^2\right] + \left(\mathbb{E}_{\mathcal{D}}[f(\boldsymbol{X};\mathcal{D})] - \boldsymbol{y}\right)^2 = Variance + Bias^2. \quad (1)$$

Generally, after the accuracy reaches a relatively high level, the accuracy needs to be sacrificed to increase the diversity. In the undersampling ensemble framework, the base classifiers are homogeneous, so the source of diversity depends on the undersampling strategy. Therefore, how to design a suitable undersampling strategy to train multiple sets of "good but different" base classifiers is the core of the framework.

B. Clustering and the Sampling Information

Cluster analysis is an unsupervised learning method which divides the samples with high similarity into the same cluster. The dataset can be analyzed from the perspective of clusters instead of analyzing from the perspective of a single sample. Commonly used clustering algorithms include k-means, Gaussian mixture model, etc.

In the undersampling ensemble framework, we need to design an undersampling strategy to select a certain number of samples from a large number of majority samples. The selected samples should describe the overall characteristics of the majority class. If only the information of the sample individuals is considered, it will cause the loss of the overall information of the majority class. Therefore, we apply cluster analysis in the undersampling stage to provide information from the two different granularities of the individuals and clusters to make the final sampling decision more reasonable. At the same time, the clustering result can help identify within-class imbalance in the dataset and avoid overlooking this problem.

The function f denotes a single base classifier, if it is a hard classifier, $f : \mathbb{R}^d \to \{0, 1\}$, if it is a soft classifier, $f : \mathbb{R}^d \to [0, 1]$. The function $F_t : \mathbb{R}^d \to [0, 1]$ denotes the ensemble classifier that is formed by t base classifiers. The dataset $\mathcal{S}$ can be randomly divided into a training set $\mathcal{T}$ and a validation set $\mathcal{V}$ according to a certain proportion.

As mentioned before, we use the clustering algorithm C to divide the dataset $\mathcal{S}$ into k clusters. After clustering, the samples i can be denoted as $(\boldsymbol{x}_i, y_i, c_i)$, where the $c_i \in \{1, 2, \ldots, k\}$ is the clustering index. Motivated by the boost-based framework, the main information of the ensemble training process provided to the sampler is the individual samples' classification errors of the trained base classifiers F_t. The symbol $\mathcal{H}$ denotes the classification error function, it can be any "decomposable" error functions, such as absolute error, cross entropy, and squared error. Then the classification error of sample i with the trained base classifiers F_t can be given by the function

$$e_{t,i} = \mathcal{H}(F_t(\boldsymbol{x}_i), y_i). \quad (2)$$

For example, if $\mathcal{H}$ is absolute error, the classification error is $e_{t,i} = abs(F_t(\boldsymbol{x}_i) - y_i)$.

Unlike the sampling information provided by the existing boost-based frameworks, our proposed framework not only improves the classification error of individual samples but also provides the average classification error of the cluster to which the sample belongs. The advantage is it provides information from different fields of view. The sampler comprehensively applies local information and overall information of the dataset, making the sampling results more diverse. The cluster classification error of the sample i with the trained base classifiers F_t can be expressed as

$$ce_{t,i} = \frac{\sum_{(\boldsymbol{x}_j, y_j) \in \mathcal{C}_{c_i}} \mathcal{H}(F_t(\boldsymbol{x}_j), y_j)}{|\mathcal{C}_{c_i}|} \quad (3)$$

where

$$\mathcal{C}_{c_i} = \{(\boldsymbol{x}, y, c) | c = c_i\}, c_i \in \{1, 2, \ldots, k\}. \quad (4)$$

Hence, the sampling information of samples i at time t is $(e_{t,i}, ce_{t,i})$.

C. Sampling Strategy and Ensemble Strategy

After obtaining the sampling information, we need to design the corresponding sampler and sampling strategy. Existing undersampling methods based on deep learning have designed the sampler as an end-to-end decision-making network. For example, input the sampling information of all samples at once, and train a neural network with a large output layer to output which samples are selected to form the subset [34]. Or based on the memorability of the recurrent neural network, all samples are input sequentially, and each sample is successively decided whether to add to the subset [35]. However, such an end-to-end design often has complex architecture and numerous parameters, bringing extra memory costs, training costs, and hardship in optimization.

To make the sampler more concise and efficient, we apply a weighted sampling strategy based on the Gaussian function instead of an end-to-end sampling decision. This strategy determines the samples' sampling weight according to the samples' sampling information and the decision of the sampler. Finally, it builds the subset based on the sampling weight of all samples. The sampling weight of $(\boldsymbol{x}_i, y_i, c_i)$ is defined as

$$w_{\mu_t, \sigma_{c_i}}(\boldsymbol{x}_i, y_i) = e^{-\frac{1}{2}\left(\frac{\mathcal{H}(F_t(\boldsymbol{x}_i), y_i) - \mu_t}{\sigma_{c_i}}\right)^2} \quad (5)$$

where e is the Euler number, $\mu \in [0, 1]$ is a hyper-parameter decided by sampler and $\sigma \in [\sigma_{min}, \sigma_{max}]$ is a hyper-parameter determined by the clustering information.

In the definition of sampling weight, the role of μ_t is to decide which types of majority samples will be given a higher sampling weight. For example, if $\mu_t = 1.0$, the higher the probability of misclassified in the previous stage, the higher the sample sampling weight in the next stage. If $\mu_t = 0.5$, the harder it is to determine its category in the previous stage, the higher the sample sampling weight in the next stage. If $\mu_t = 0$, the higher the probability of correctly classifying in the previous stage, the higher the sampling weight in the next stage. Therefore, when different values of μ_t are selected, different types of the majority samples will be selected in the next stage, increasing the diversity

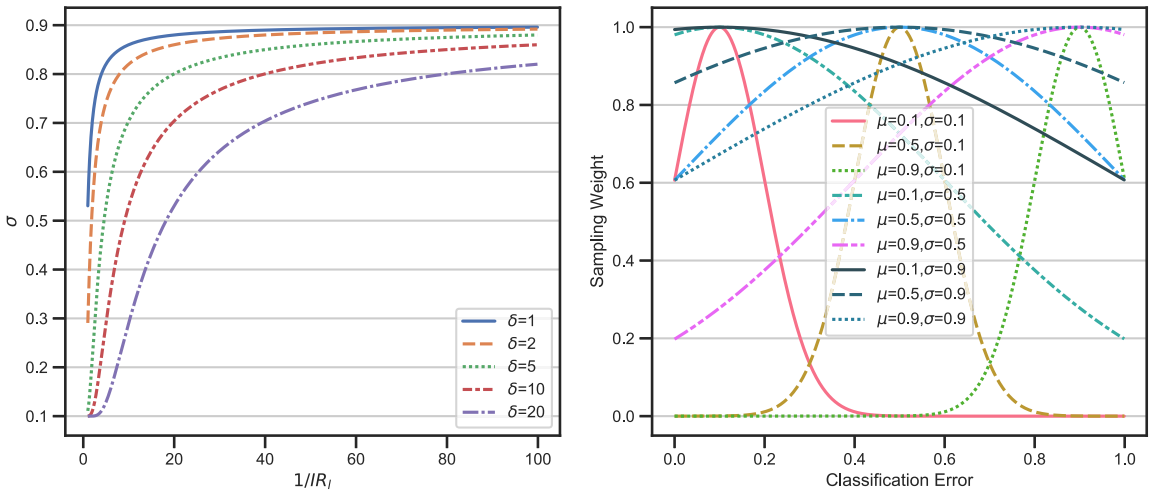

Fig. 3. Visual display of relevant formulas in sampling strategy and the influence of some key parameters. (Best viewed in color).

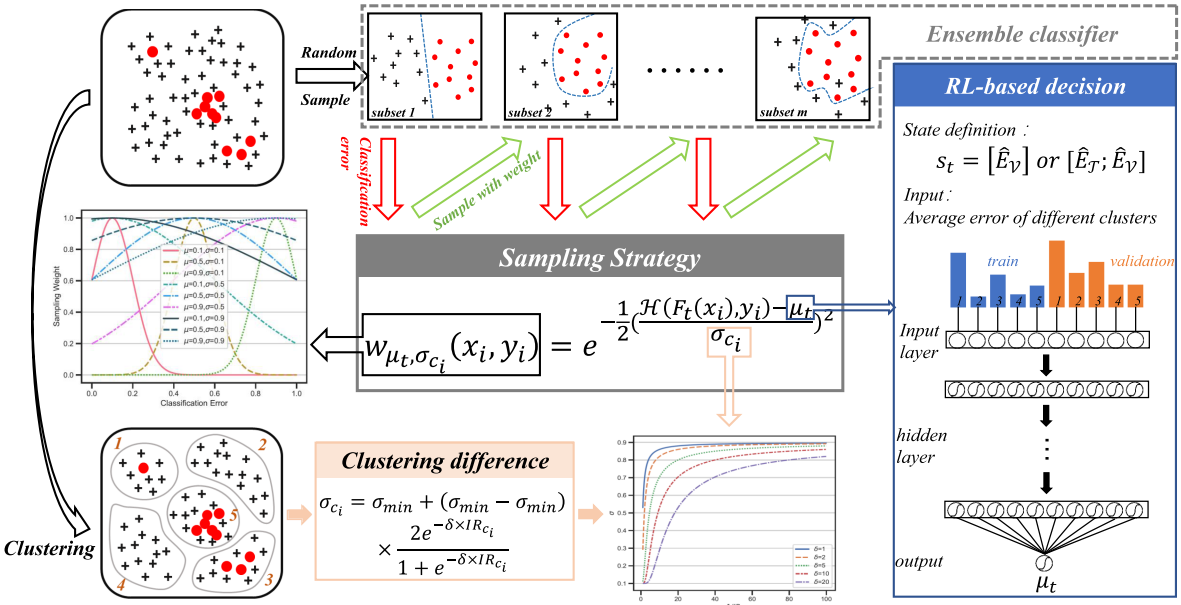

Fig. 4. Overview of the proposed adaptive undersampling framework.

of the subtraining sets and improving the final classification performance. Equation (5) is different from the sampling weight of traditional undersampling ensemble methods such as RUSBoost and Cascade. With the change of the decision parameter μ, the sample with larger classification error in the previous stage will not always obtain a larger sampling weight, as shown in Fig. 3.

For clusters with different properties, we hope to implement different sampling strategies with the help of hyper-parameter σ. As shown in Fig. 4, after we divide the original dataset $\mathcal{S}$ into k clusters, the distribution of minority class in different clusters will vary greatly due to the imbalanced class distribution and the within-class imbalance. For example, the clusters 2 and 4 do not contain minority samples and the cluster 1 contains only a few minority samples. The sampling algorithm hopes to preserve the overall distribution framework of its majority class as much as possible, so the sampling weight variance of samples in these clusters should be relatively small. For clusters such as cluster 3 and cluster 5 that contain most of the minority samples, the sampling algorithm needs to pay more attention to the samples at the classification boundary of the majority class and the minority class. The samples near the boundary are easier to be misclassified. When the value of μ_t is large, a larger sampling weight will be assigned to the samples near the boundary, and a smaller sampling weight will be assigned to the samples far away from the boundary. This enables a greater focus on samples at the boundary of majority/minority classes for clusters with minority samples. Hence, the sampling strategy of different clusters should be different. To achieve the above purpose, the value of the hyper-parameter σ will be determined by the imbalanced ratio of the cluster to which the sample

Algorithm 1: The Training Process of the Proposed Framework.

Input: Dataset $\mathcal{S}$, base classifier f, number of base classifier m, clustering algorithm $\mathcal{C}$, number of clusters k, neural network sampler $Sampler(\cdot)$, classification error function $\mathcal{H}(\cdot)$, evaluation index $score(\cdot)$;

Output: Ensemble classifier F_m;

Train clustering algorithm $\mathcal{C}$ on $\mathcal{S}$, adding the clustering information into dataset, $\mathcal{S} = \{(\boldsymbol{x}, y, c)\}$;
Dividing $\mathcal{S}$ into a training set $\mathcal{T}$ and a validation set $\mathcal{V}$;
Apply Eqs.(6) and Eqs.(7) to calculate σ of each cluster in the training set $\mathcal{T}$;
Train base classifier f_1 with random balanced subset $\mathcal{T}_1$ of $\mathcal{T}$;
Divide the training set $\mathcal{T}$ into the majority set $\mathcal{N}_\mathcal{T}$ and the minority set $\mathcal{P}_\mathcal{T}$;
for $t = 1, \ldots, m-1$ **do**
  $F_t(\boldsymbol{x}) = \frac{1}{t}\sum_{i=1}^{t} f_i(\boldsymbol{x})$ or $F_t(\boldsymbol{x}) = \frac{\sum_{i=1}^{t} score(f_i,\mathcal{V}) f_i(\boldsymbol{x})}{\sum_{i=1}^{t} score(f_i,\mathcal{V})}$;
  Compute the sample classification error $\mathcal{H}(F_t(\boldsymbol{x}), y)$ and the cluster classification error $ce_{t,i}$ by Eqs.(3);
  Get the current state $s_t = [\hat{E}_\mathcal{V}]$ or $s_t = [\hat{E}_\mathcal{T}; \hat{E}_\mathcal{V}]$;
  for *each environment update step* **do**
    Take action by the current sampler $\mu_t \sim Sampler(\mu_t|s_t)$;
    Get the reward $r_t = score(F_{t+1}, \mathcal{V}) - score(F_t, \mathcal{V})$ and the next state s_{t+1};
    Store transition to the memory $\mathcal{M}$, $\mathcal{M} = \mathcal{M} \cup (s_t, \mu_t, r_t, s_{t+1})$;
  for *each gradient update step* **do**
    Update the sampler's parameters $Sampler(\cdot)$ according to SAC [36];
  Assign each sample $(\boldsymbol{x}_i, y_i, c_i)$ in $\mathcal{N}_\mathcal{T}$ with sampling weight $w_{\mu_{t_i}, \sigma_{c_i}}(\boldsymbol{x}_i, y_i)$ by Eqs.(5);
  Sample the majority subset $\mathcal{N}'_\mathcal{T}$ from $\mathcal{N}_\mathcal{T}$ according to the sampling weight $\boldsymbol{w}$, where $|\mathcal{N}'_\mathcal{T}| = |\mathcal{P}_\mathcal{T}|$;
  Get the current balanced training subset $\mathcal{T}_t = \mathcal{N}'_\mathcal{T} \cup \mathcal{P}_\mathcal{T}$;
  Train the new base classifier f_{t+1} with subset $\mathcal{T}_{t+1}$;
Get the final ensemble classifier $F_m(\boldsymbol{x}) = \frac{1}{m}\sum_{i=1}^{m} f_i(\boldsymbol{x})$ or $F_m(\boldsymbol{x}) = \frac{\sum_{i=1}^{m} score(f_i,\mathcal{V}) f_i(\boldsymbol{x})}{\sum_{i=1}^{m} score(f_i,\mathcal{V})}$.

belongs, as follows:

$$\sigma_{c_i} = \sigma_{min} + (\sigma_{max} - \sigma_{min}) \times \frac{2e^{-\delta \times IR_{c_i}}}{1 + e^{-\delta \times IR_{c_i}}} \tag{6}$$

where

$$IR_{c_i} = \frac{|\{(\boldsymbol{x}, y, c)|c = c_i, y = 0\}|}{|\{(\boldsymbol{x}, y, c)|c = c_i, y = 1\}|}, c_i \in \{1, 2, \ldots, k\} \tag{7}$$

σ_{min} and σ_{max} are the lower bound and the upper bound of σ and δ is the scale parameter. It can be seen from Fig. 3 that the larger the imbalance ratio and the larger σ, the smaller the fluctuation of the sampling weight. Fig. 4 shows the overall structure of the proposed framework.

There are two main ensemble strategies for base classifiers: average ensemble and weighted ensemble.

1) The average ensemble is simple. The ensemble result is the average result of all base classifiers, which can be expressed as

$$F_t(\boldsymbol{x}) = \frac{1}{t}\sum_{i=1}^{t} f_i(\boldsymbol{x}). \tag{8}$$

2) In weighted ensemble, the weight of the base classifier is often determined by the performance of the base classifier on the validation set $\mathcal{V}$. Assuming that the evaluation index is $score(\cdot)$, it is defined as follows:

$$F_t(\boldsymbol{x}) = \frac{\sum_{i=1}^{t} score(f_i, \mathcal{V}) f_i(\boldsymbol{x})}{\sum_{i=1}^{t} score(f_i, \mathcal{V})}. \tag{9}$$

In the proposed framework, the error information of the validation set needs to be used to represent the training state of the ensemble framework at the current moment and determine subsequent sampling decisions. Therefore, the validation set is no longer suitable for determining the weight of the base classifier. From the perspective of the limited training data and computational convenience, the subsequent experiments adopt an average ensemble.

D. Sampling Decision Based on Reinforcement Learning

As mentioned before, the proposed sampler $Sampler(\cdot)$ is trained to decide the next stage's sampling strategy based on the classification performance of the current ensemble classifiers. The input state s needs to represent the current performance comprehensively. Considering the input dimensions, we apply the classification error of each cluster instead of each sample. Suppose the classification error distribution on dataset $\mathcal{S}$ is $E_\mathcal{S}$, we use the cluster classification error of each clusters to approximate the $E_\mathcal{S}$, i.e., $\hat{E}_\mathcal{S} = [ce_1, ce_2, \ldots, ce_k] \in \mathbb{R}^k$. From the perspective of the training set and the validation set, it can be described as the following two strategies.

1) Strategy A: Only considering the classification error of the validation set, we have the state definition: $s = [\hat{E}_\mathcal{V}] \in [0, 1]^k$.
2) Strategy B: Considering the classification error of the training set and the validation set, we have the state definition: $s = [\hat{E}_\mathcal{T}; \hat{E}_\mathcal{V}] \in [0, 1]^{2^\sim k}$.

The decision is not an end-to-end sampled subset but a critical parameter μ of the Gaussian function to decide each majority sample's sampling weight. The sampler is expected to optimize its decision from a state(s_t)-action(μ_t)-state(s_{t+1}) interactions. In response to such problems, we naturally thought of modeling the sampler as an MDP and optimizing it through reinforcement learning (RL) methods.

We model the sampler's training as a MDP defined by $< S, A, T, R, p_0 >$, where S is the state space, $S = [0,1]^k$ or $[0,1]^{2^-k}$; A is the action space, $A = [0,1]$; $T(s_{t+1}; s_t, a_t)$ is the unknown state transition probability of changing to next state s_{t+1} from current state s_t when action a_t is taken; $r(s_t, a_t)$ is the reward of executing action a_t in state s_t; $p_0(s)$ denotes the probability of starting in state s. The sampler (policy) is the mapping π: $S \to A$. The optimization goal is to find the best policy π^* to maximize the expected cumulative reward, that is

$$\pi^* = \arg\max_{\pi} \sum_{s_0 \in S} p_0(s_0) \mathbb{E}\left[\sum_{t=0}^{m} r(s_t, a_t) \mid p_0(s_0), s_0, \pi\right]. \tag{10}$$

More specifically, in the implementation process, first random undersampling is performed from the original dataset to obtain the initial balanced training subset $\mathcal{T}_1$, the first trained base classifier f_1 and initial state s_0. Then the ensemble classifier F_t is trained iteratively. In each environment update step, we can compute the current state $s_t = [\hat{E}_{\mathcal{V}}]$ or $s_t = [\hat{E}_{\mathcal{T}}; \hat{E}_{\mathcal{V}}]$ through the current trained ensemble classifier F_t. Then the action $a_t = \mu_t \sim \pi(\mu_t \mid s_t)$ is selected. After getting the parameter μ_t and the parameter σ, sampling weight of each sampling can be calculated. Perform weighted sampling on majority samples in the training set to obtain a new balanced training subset $\mathcal{T}_{t+1}$, and use it to train a new base classifier f_{t+1}. A new ensemble classifier F_{t+1} can be obtained by adding f_{t+1} into the ensemble. So far we can get the reward of an update, which is represented by the classification performance difference of the ensemble classifier before and after the update on the validation set, $r_t = score(F_{t+1}, \mathcal{V}) - score(F_t, \mathcal{V})$. The next state s_{t+1} is sampled according to transition probability, $s_{t+1} \sim T(s_{t+1}; s_t, a_t)$.

Due to the dimensionality of the state space and continuous domain, the optimization of the above problem is intractable for the traditional reinforcement learning method. We take advantage of a reinforcement learning method named SAC [36] to optimize the sampler. SAC is an off-policy actor-critic algorithm based on the maximum entropy RL framework. Compared with other RL methods, it can increase the randomness of the policy, accelerate the subsequent learning speed, and prevent the policy from prematurely converging to the local optimum. The SAC implementation of this problem is divided into four parts: a parameterized state value function $V_\psi(s_t)$, a target value function $V_{\bar{\psi}}(s_t)$ which is an exponentially moving average of the state value function weight, a soft Q-function $Q_\theta(s_t, a_t)$, and a tractable policy (the sampler) $\pi_\phi(a_t \mid s_t)$. These functions are in the form of fully connected neural networks, and the parameters of these networks are $\psi, \bar{\psi}, \theta$ and ϕ. The specific parameter optimization rules are shown in [36, Algorithm 1], which will not be described in detail here. See Appendix A for more details of the SAC framework.

Algorithm 1 shows the overall training procedures of the proposed adaptive undersampling framework. For complexity analysis, the proposed framework can be roughly regarded as a clustering model, an undersampling ensemble learning framework and a sampler optimization process. The complexity of the clustering model depends on the clustering method used, which is not analyzed here. Assuming that the cost of training a base classifier $f(\cdot)$ with sample size N is $T_f(N)$, then the training cost of the undersampling ensemble part is: $kT_f(2|\mathcal{P}|)$. Suppose the cost of the sampler performing a single gradient update is T_{update}, we can easily get the training cost of the sampler is $(n_{random} + n_{update})T_f(2|\mathcal{P}|) + n_{update}T_{update}$, where n_{random} is the steps for collecting transitions with random actions before start updating and n_{random} is the steps for collecting online transitions and perform gradient updates.

IV. Experiments and Evaluations

In this section, the proposed undersampling ensemble method and other imbalanced classification methods are conducted on two real-world industrial datasets: the wind turbine blade cracking fault detection datasets and a high-speed train (HST) brake system fault dataset. A series of experiments and comparisons are carried out to evaluate the performance and explore some properties of the proposed framework.

A. Experimental Settings

For imbalanced datasets, the accuracy cannot reflect the classification performance well. Therefore, in this study, we apply $F1\text{-}score$, $G\text{-}mean$, MCC, and $AUCPRC$ four evaluation indexes to evaluate the results. Their formulas are defined as follows:

$$F1\text{-}score = \frac{2TP}{2TP + FP + FN} \tag{11}$$

$$G\text{-}mean = \sqrt{\frac{TP^2}{(TP + FN)(TP + FP)}} \tag{12}$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{13}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$AUCPRC = Area\ Under\ Precision\text{-}Recall\ Curve \tag{14}$$

where TP, TF, FP, and FN, respectively, represent the number of true positive, true negative, false positive, and false negative samples in the prediction result.

To compare the effectiveness of the proposed framework in solving imbalanced classification problems, we select 16 existing imbalanced learning methods as benchmarks. Including eight resampling methods in the imbalanced-learn Python package [37], specifically three undersampling methods Random UnderSampling (RUS), Cluster Centroids (CC) [38], Edited Nearest Neighbor (ENN), and TomekLinks (Tomek); three oversampling methods Synthetic Minority Oversampling TEchnique (SMOTE), Borderline SMOTE (BSMOTE), and ADAptive SYNthetic oversampling (ADASYN); and two hybrid sampling methods SMOTE with ENN (SMOTEENN) and SMOTE with Tomek (SMOTETomek). It also includes seven

TABLE I
INFORMATION OF 4 WIND TURBINE BLADE CRACKING FAILURE DATASETS

Wind turbine no.	#Samples	Imbalance ratio	Raw sample	Preprocessed sample
37	1474	17.20:1	75×480	41×1
49	2343	20.11:1	75×480	41×1
63	3887	25.09:1	75×480	41×1
80	778	15.21:1	75×480	41×1

resampling ensemble frameworks [39], namely, RUSBoost [29], SMOTEBoost [28], UnderBagging [26], SMOTEBagging [27], Cascade [13], SPE [30], and MESA [31]. In the subsequent experiments, for the validity of the comparison, decision tree (DT) is used as classifiers for all methods without special instructions, and the number of base classifiers of the ensemble framework are set to ten. The proposed framework uses k-means as the clustering method with the number of clusters $k = 10$. Each experiment will be repeated ten times to reduce the randomness of the results.

B. Case 1: Wind Turbine Blade Cracking Fault Detection

Blade cracking is a vital cause of wind turbine operation failure. This dataset comes from a data mining competition held by China State Power Investment Co. Ltd. on DataFountain. The data contain time series of 75 features of different types such as current, voltage, temperature, pressure, etc., collected by the supervisory control and data acquisition (SCADA) system. The goal is to use the abovementioned sensor monitoring data to predict whether the wind turbine will experience blade cracking within a week. The dataset provides time-discontinuous samples. Each sample contains 10 min of monitoring data. The sampling frequency of the system is about 480 times/10 min, so the raw sample is a 75×480 matrix. We perform a simple preprocessing on the datasets of the four wind turbines before the experiment. First, the time-series signal within 10 min of each feature is converted into 6 statistical features, which are: maximum, minimum, average, median, variance, and range. Then the feature selection method is applied, and 41 features are selected as the input features. The information of the final experimental datasets is shown in Table I.

μ_t and σ are the two most important parameters determining the proposed algorithm's sampling strategy. To explore the influence of its design on the final classification performance, we take μ_t and σ as a fixed value for Algorithm 1 A, respectively. The experimental results are shown in Fig. 5. It can be seen that different values of μ_t and σ can greatly affect the results, especially μ_t. When μ_t takes a larger value, that is, a higher sampling weight is given to the samples with a high probability of misclassification in the previous stage, and the final classification result is better. For σ, it is better to take between 0.1 and 0.3. To further verify the influence of each part in Algorithm 1, we designed an ablation experiment. The results are shown in Table II, where random divided means that the clustering algorithm is not used, but the training set is randomly divided into k clusters. The bold values in tables show the best results among all methods. According to the experimental results, it can be proved that the clustering,

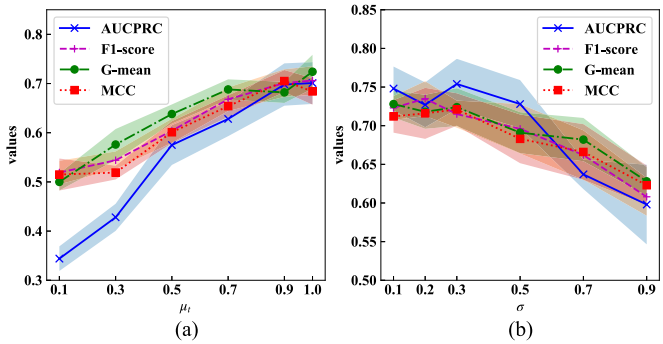

Fig. 5. Different μ_t and σ of Algorithm 1 A for #49 datasets.

TABLE II
ABLATION EXPERIMENT OF ALGORITHM 1

Method	Metric	#37	#49	#63	#80
Algorithm 1 A	$AUCROC$	**0.687-0.041**	**0.790-0.027**	**0.733-0.065**	**0.909-0.094**
	$F1$-$score$	**0.719-0.033**	**0.761-0.061**	**0.731-0.035**	**0.903-0.054**
	G-$mean$	**0.745-0.019**	**0.775-0.042**	**0.721-0.037**	0.894-0.074
	MCC	**0.725-0.060**	**0.753-0.049**	**0.727-0.042**	0.888-0.072
Algorithm 1 B	$AUCROC$	0.666-0.057	0.758-0.058	0.698-0.053	0.905-0.040
	$F1$-$score$	0.709-0.027	0.748-0.057	0.709-0.034	0.868-0.075
	G-$mean$	0.723-0.060	0.763-0.041	0.718-0.045	**0.901-0.060**
	MCC	0.716-0.049	0.735-0.054	0.709-0.052	**0.896-0.062**
Random divided	$AUCROC$	0.630-0.051	0.727-0.044	0.695-0.046	0.866-0.060
	$F1$-$score$	0.665-0.049	0.703-0.045	0.644-0.053	0.859-0.083
	G-$mean$	0.693-0.050	0.691-0.051	0.655-0.027	0.856-0.060
	MCC	0.673-0.058	0.706-0.056	0.657-0.041	0.837-0.098
$\mu_t = 1.0$	$AUCROC$	0.512-0.110	0.718-0.085	0.657-0.046	0.854-0.095
	$F1$-$score$	0.581-0.050	0.706-0.058	0.694-0.046	0.838-0.054
	G-$mean$	0.636-0.065	0.724-0.069	0.681-0.049	0.788-0.101
	MCC	0.600-0.036	0.684-0.054	0.626-0.043	0.789-0.052
$\sigma = 0.2$	$AUCROC$	0.659-0.068	0.727-0.058	0.714-0.047	0.876-0.024
	$F1$-$score$	0.677-0.061	0.735-0.045	0.701-0.067	0.890-0.064
	G-$mean$	0.674-0.069	0.718-0.044	0.687-0.047	0.863-0.088
	MCC	0.670-0.040	0.716-0.066	0.704-0.037	0.860-0.072

the decision of μ_t, and the definition of σ in Algorithm 1 are all meaningful for the final classification result, and can effectively improve the final classification performance.

Fig. 6 visualizes the training processes of three undersampling frameworks based on reinforcement learning. To provide enough accurate decision-making information for the environment, random action selection is performed in the first 50 epochs. The results of each random action are stored in the memory. It can be seen that the three methods can effectively improve the classification performance, the results on # 49 dataset tend to be stable around the 100th epoch, and the $AUCPRC$ of the training set can reach approximately 1.00. Strategy A and Strategy B of the proposed framework performed significantly better than MESA on the validation set and test set. Among them, strategy A had the best stability and accuracy on the test set. This fully proves the excellent effect of the state definition based on the clustering method. Strategy A is better than strategy B because accurate classification can be achieved on the training set, and its information does not bring value to the decision but the accuracy of the value function estimation is affected by the increase in the state dimension.

We compare the proposed framework with the 16 benchmark methods previously mentioned on 4 wind turbine blade cracking

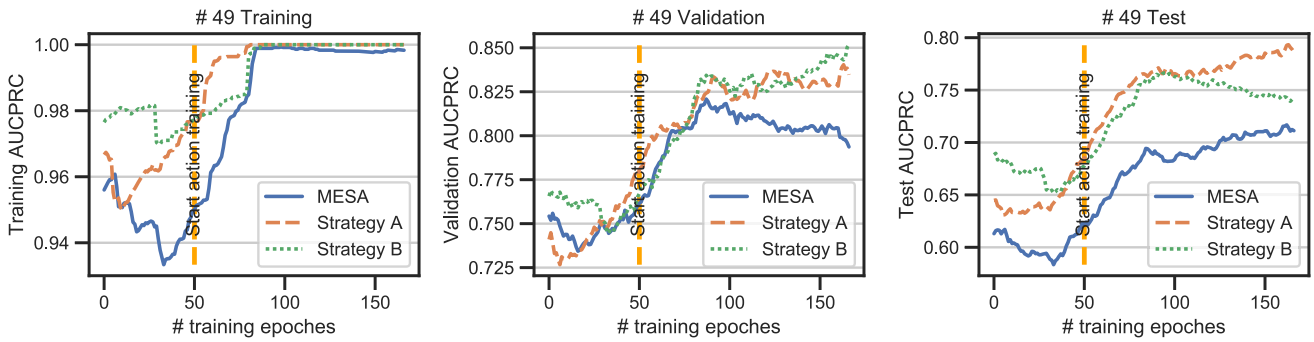

Fig. 6. Visualization of the proposed framework reinforcement learning sampler's training process. (Slide mean window = 50 epochs, best viewed in color).

TABLE III
COMPARISON OF CLASSIFICATION PERFORMANCE OF THE PROPOSED FRAMEWORK AND RESAMPLING METHODS ON 4 WIND TURBINE BLADE CRACKING FAULT DETECTION DATASETS

Dataset	Metric	RUS	CC	ENN	Tomek	SMOTE	BSMOTE	ADASYN	SMOTEENN	SMOTETomek	Algorithm 1 A	Algorithm 1 B
#37	*AUCPRC*	0.158±0.04	0.152±0.016	0.399±0.028	0.468±0.055	0.313±0.061	0.227±0.07	0.249±0.042	0.243±0.063	0.236±0.038	**0.687±0.041**	0.666±0.057
	F1-score	0.309±0.07	0.307±0.025	0.612±0.022	0.666±0.04	0.529±0.063	0.434±0.087	0.464±0.049	0.454±0.081	0.451±0.045	**0.719±0.033**	0.709±0.027
	G-mean	0.373±0.055	0.362±0.025	0.612±0.023	0.668±0.041	0.531±0.062	0.436±0.088	0.466±0.051	0.457±0.08	0.452±0.046	**0.745±0.019**	0.723±0.060
	MCC	0.3±0.083	0.295±0.029	0.589±0.025	0.649±0.045	0.503±0.066	0.399±0.093	0.432±0.052	0.42±0.086	0.417±0.049	**0.725±0.060**	0.716±0.049
#49	*AUCPRC*	0.148±0.027	0.096±0.021	0.362±0.019	0.415±0.034	0.304±0.079	0.295±0.041	0.271±0.053	0.259±0.041	0.274±0.05	**0.790±0.027**	0.758±0.058
	F1-score	0.274±0.039	0.178±0.036	0.58±0.017	0.627±0.03	0.52±0.084	0.52±0.041	0.493±0.057	0.475±0.043	0.494±0.054	**0.761±0.061**	0.748±0.057
	G-mean	0.374±0.038	0.302±0.038	0.588±0.017	0.633±0.027	0.527±0.088	0.521±0.042	0.495±0.057	0.488±0.046	0.501±0.057	**0.775±0.042**	0.763±0.041
	MCC	0.311±0.047	0.207±0.064	0.564±0.018	0.611±0.029	0.5±0.091	0.495±0.044	0.467±0.06	0.456±0.048	0.471±0.06	**0.753±0.049**	0.735±0.054
#63	*AUCPRC*	0.141±0.036	0.050±0.006	0.414±0.06	0.442±0.03	0.407±0.073	0.431±0.074	0.332±0.087	0.418±0.049	0.434±0.056	**0.733±0.065**	0.698±0.053
	F1-score	0.265±0.059	0.097±0.012	0.631±0.049	0.655±0.024	0.62±0.062	0.64±0.064	0.556±0.076	0.63±0.038	0.643±0.043	**0.731±0.035**	0.709±0.034
	G-mean	0.364±0.052	0.211±0.014	0.632±0.048	0.655±0.023	0.627±0.062	0.646±0.063	0.558±0.076	0.637±0.04	0.65±0.046	**0.721±0.037**	0.718±0.045
	MCC	0.313±0.063	0.085±0.036	0.616±0.05	0.641±0.024	0.61±0.064	0.63±0.066	0.539±0.08	0.621±0.042	0.634±0.048	**0.727±0.042**	0.709±0.052
#80	*AUCPRC*	0.204±0.026	0.207±0.034	0.624±0.031	0.355±0.043	0.26±0.068	0.253±0.052	0.26±0.152	0.246±0.07	0.265±0.067	**0.909±0.094**	0.905±0.040
	F1-score	0.35±0.031	0.345±0.048	0.781±0.02	0.569±0.041	0.469±0.086	0.464±0.061	0.457±0.135	0.452±0.088	0.475±0.08	**0.903±0.054**	0.868±0.075
	G-mean	0.443±0.035	0.452±0.040	0.781±0.02	0.57±0.041	0.471±0.086	0.466±0.062	0.458±0.135	0.457±0.086	0.478±0.081	0.894±0.074	**0.901±0.060**
	MCC	0.378±0.042	0.386±0.051	0.766±0.021	0.542±0.043	0.432±0.092	0.427±0.066	0.42±0.145	0.414±0.095	0.439±0.087	0.888±0.072	**0.896±0.062**

TABLE IV
COMPARISON OF CLASSIFICATION PERFORMANCE OF THE PROPOSED FRAMEWORK AND RESAMPLING ENSEMBLE FRAMEWORKS ON FOUR WIND TURBINE BLADE CRACKING FAULT DETECTION DATASETS

Dataset	Metric	RUSBoost	SMOTEBoost	UnderBagging	SMOTEBagging	Cascade	SPE	MESA	Algorithm 1 A	Algorithm 1 B
#37	*AUCPRC*	0.61±0.064	0.508±0.076	0.323±0.081	0.449±0.037	0.514±0.102	0.629±0.074	0.634±0.081	**0.687±0.041**	0.666±0.057
	F1-score	0.681±0.066	0.607±0.083	0.479±0.085	0.53±0.037	0.602±0.078	0.71±0.058	0.684±0.046	**0.719±0.033**	0.709±0.027
	G-mean	0.696±0.072	0.615±0.078	0.498±0.076	0.54±0.034	0.609±0.076	0.716±0.06	0.704±0.044	**0.745±0.019**	0.723±0.060
	MCC	0.379±0.058	0.545±0.099	0.463±0.082	0.511±0.039	0.589±0.079	0.701±0.063	0.701±0.035	**0.725±0.060**	0.716±0.049
#49	*AUCPRC*	0.472±0.093	0.622±0.062	0.536±0.097	0.65±0.06	0.67±0.087	0.735±0.065	0.731±0.091	**0.790±0.027**	0.758±0.058
	F1-score	0.546±0.074	0.675±0.052	0.603±0.075	0.673±0.05	0.67±0.069	0.711±0.054	0.705±0.052	**0.761±0.061**	0.748±0.057
	G-mean	0.566±0.057	0.688±0.055	0.619±0.066	0.68±0.047	0.677±0.066	0.718±0.053	0.704±0.056	**0.775±0.042**	0.763±0.041
	MCC	0.393±0.041	0.629±0.034	0.596±0.072	0.664±0.049	0.658±0.071	0.705±0.055	0.717±0.041	**0.753±0.049**	0.735±0.054
#63	*AUCPRC*	0.452±0.067	0.701±0.041	0.53±0.079	0.56±0.091	0.665±0.033	0.661±0.075	0.691±0.050	**0.733±0.065**	0.698±0.053
	F1-score	0.504±0.044	0.713±0.03	0.601±0.053	0.602±0.063	0.666±0.021	0.672±0.066	0.692±0.070	**0.731±0.035**	0.709±0.034
	G-mean	0.512±0.042	0.717±0.028	0.609±0.05	0.611±0.066	0.677±0.024	0.68±0.067	0.692±0.053	**0.721±0.037**	0.718±0.045
	MCC	0.358±0.038	0.688±0.052	0.593±0.053	0.598±0.07	0.666±0.025	0.669±0.07	0.691±0.076	**0.727±0.042**	0.709±0.052
#80	*AUCPRC*	0.697±0.156	0.447±0.105	0.528±0.128	0.643±0.085	0.814±0.102	0.889±0.074	0.898±0.058	**0.909±0.094**	0.905±0.040
	F1-score	0.737±0.111	0.562±0.074	0.636±0.107	0.67±0.077	0.786±0.094	0.852±0.077	0.840±0.080	**0.903±0.054**	0.868±0.075
	G-mean	0.744±0.109	0.591±0.075	0.668±0.089	0.685±0.079	0.798±0.087	0.857±0.073	0.884±0.057	0.894±0.074	**0.901±0.060**
	MCC	0.47±0.07	0.458±0.066	0.639±0.099	0.662±0.084	0.783±0.094	0.848±0.078	0.865±0.062	0.888±0.072	**0.896±0.062**

datasets. All methods use DT as classifiers, and the number of base classifiers for all resampling ensemble frameworks is set to ten. Due to space constraints, the experimental results are displayed in two tables. Table III shows the experimental results of all resampling benchmark methods, and Table IV shows the experimental results of all resampling ensemble frameworks. In general, the performance of the resampling ensemble frameworks is significantly better than the performance of the resampling methods.

The proposed framework achieves the best performance on the four evaluation indexes of the four datasets, among which Strategy A achieves the best performance on 14 indexes, and Strategy B achieves the best performance on two indexes. Compared with other benchmark methods, the Algorithm 1 Strategy A has a significant improvement in each evaluation index of each dataset, which fully proves the effectiveness and robustness of the proposed framework to deal with imbalanced classification problems. Compared with other methods, Cascade, SPE, and

TABLE V
p-VALUES OF THE PAIRED T-TEST BETWEEN BENCHMARKS AND ALGORITHM 1

Method	RUS	CC	ENN	Tomek	SMOTE	BSMOTE
p-value to Algorithm 1 A	3.969×10^{-12}	1.484×10^{-12}	2.106×10^{-6}	5.367×10^{-5}	1.274×10^{-6}	7.877×10^{-7}
p-value to Algorithm 1 B	4.085×10^{-12}	1.100×10^{-12}	3.040×10^{-6}	1.243×10^{-4}	2.624×10^{-6}	1.584×10^{-6}
Method	ADASYN	SMOTEENN	SMOTETomek	RUSBoost	SMOTEBoost	UnderBagging
p-value to Algorithm 1 A	4.491×10^{-8}	5.333×10^{-7}	7.059×10^{-7}	1.030×10^{-6}	2.830×10^{-4}	1.062×10^{-8}
p-value to Algorithm 1 B	8.697×10^{-8}	1.048×10^{-6}	1.399×10^{-6}	2.824×10^{-6}	8.932×10^{-4}	3.321×10^{-8}
Method	SMOTEBagging	Cascade	SPE	MESA		
p-value to Algorithm 1 A	7.530×10^{-9}	2.783×10^{-9}	2.739×10^{-8}	2.485×10^{-7}		
p-value to Algorithm 1 B	5.181×10^{-8}	3.689×10^{-8}	9.140×10^{-7}	2.172×10^{-6}		

MESA also perform better, which also verifies the effectiveness of the undersampling ensemble framework mentioned in Section III-A for imbalance problems. The best comprehensive performance among all methods is Algorithm 1 Strategy A, Algorithm 1 Strategy B and MESA, which are all the undersampling ensemble frameworks based on reinforcement learning. This shows that reinforcement learning can effectively improve the rationality of sampling strategies and enhance the effect of ensemble framework. The improvement of the proposed framework compared with MESA also verifies that the state definition and sampling strategy determination based on the clustering results are suitable for this framework, and can generate more reasonable and divergent subtraining sets.

In addition, Paired t-test is utilized to test whether the benchmark and the proposed method perform significantly differently. The performances of each benchmark in Tables III and IV on the four datasets is compared with the Algorithm 1 A and B, respectively, and the p-values of the paired t-test are summarized in Table V. It can be seen that all p-values are far less than 0.01, indicating that the proposed method has a significant improvement.

The proposed framework is very flexible. It can use different kinds of classification methods as base classifiers, and it can also choose different numbers of base classifiers for the ensemble. To better explore the nature of the proposed framework, we conduct the following two sets of experiments.

We choose six commonly used classification methods as base classifiers to experiment on the # 49 dataset, including:

1) decision tree (DT);
2) K-nearest neighbors (KNN);
3) support vector machine (SVM);
4) multilayer perceptron neural network (MLP-NN);
5) random forest (RF);
6) XGBoost (XGB).

The number of all base classifiers is set to ten. The experimental results are summarized in Table VI . The proposed framework achieves the best results when DT, KNN, RF, and XGB are applied as base classifiers, while the performance of SVM and MLP-NN as base classifiers is not as good as SMOTEBagging. Algorithm 1 is more suitable for tree-based classifiers such as DT and RF, while methods with strong classification performance, especially for the SVM do not achieve good results. It can also be seen that the Bagging-based ensemble strategy performs significantly better than the boosting-based ensemble strategy when strong classifiers such as SVM and MLP-NN are applied. Hence, the appropriate strategy needs to be chosen according to the nature of the classifier in practical applications. The overall performance is optimal when RF is chosen as the base classifier,

TABLE VI
CLASSIFICATION PERFORMANCE OF THE RESAMPLING ENSEMBLE FRAMEWORKS ON #49 WITH DIFFERENT TYPES OF BASE CLASSIFIER

Classifier	Method	$AUCPRC$	$F1$-$score$	G-$mean$	MCC
DT	RUSBoost	0.472±0.093	0.546±0.074	0.566±0.057	0.393±0.041
	SMOTEBoost	0.622±0.062	0.675±0.052	0.688±0.055	0.629±0.034
	UnderBagging	0.536±0.097	0.603±0.075	0.619±0.066	0.596±0.072
	SMOTEBagging	0.65±0.06	0.673±0.05	0.68±0.047	0.664±0.049
	Cascade	0.67±0.087	0.67±0.069	0.677±0.066	0.658±0.071
	SPE	0.735±0.065	0.711±0.054	0.718±0.053	0.705±0.055
	MESA	0.731±0.091	0.705±0.052	0.704±0.056	0.717±0.041
	Algorithm 1 A	**0.790±0.027**	**0.761±0.061**	**0.775±0.042**	**0.753±0.049**
	Algorithm 1 B	0.758±0.058	0.748±0.057	0.763±0.041	0.735±0.054
KNN	UnderBagging	0.537±0.074	0.599±0.075	0.611±0.072	0.592±0.075
	SMOTEBagging	0.642±0.074	0.652±0.06	0.657±0.058	0.638±0.063
	Cascade	0.425±0.046	0.592±0.044	0.594±0.043	0.571±0.046
	SPE	0.36±0.025	0.528±0.023	0.542±0.019	0.513±0.021
	MESA	0.642±0.034	0.642±0.034	0.639±0.035	0.621±0.023
	Algorithm 1 A	**0.665±0.020**	**0.685±0.025**	**0.683±0.029**	**0.666±0.029**
	Algorithm 1 B	0.659±0.019	0.670±0.023	0.677±0.020	0.666±0.017
SVM	RUSBoost	0.319±0.12	0.47±0.131	0.52±0.1	0.455±0.11
	SMOTEBoost	0.249±0.066	0.376±0.102	0.418±0.085	0.356±0.094
	UnderBagging	0.522±0.053	0.597±0.045	0.61±0.034	0.587±0.038
	SMOTEBagging	**0.656±0.069**	**0.664±0.038**	**0.675±0.038**	**0.66±0.04**
	Cascade	0.435±0.095	0.535±0.049	0.559±0.048	0.533±0.056
	SPE	0.28±0.078	0.446±0.1	0.484±0.097	0.444±0.111
	MESA	0.396±0.033	0.562±0.011	0.595±0.011	0.553±0.016
	Algorithm 1 A	0.361-0.018	0.557±0.028	0.588±0.018	0.548±0.020
	Algorithm 1 B	0.399-0.019	0.574±0.011	0.598-0.018	0.558±0.020
MLP-NN	UnderBagging	0.525±0.086	0.595±0.057	0.614±0.058	0.591±0.064
	SMOTEBagging	**0.661±0.071**	**0.673±0.063**	**0.681±0.066**	**0.667±0.067**
	Cascade	0.461±0.093	0.546±0.064	0.562±0.067	0.539±0.072
	SPE	0.454±0.073	0.568±0.061	0.591±0.043	0.565±0.049
	MESA	0.514±0.027	0.601±0.012	0.620±0.015	0.598±0.027
	Algorithm 1 A	0.569±0.053	0.629±0.038	0.627±0.017	0.601±0.024
	Algorithm 1 B	0.574±0.041	0.608±0.038	0.642±0.034	0.592±0.020
RF	RUSBoost	0.705±0.056	0.689±0.059	0.698±0.053	0.654±0.051
	SMOTEBoost	0.844±0.022	0.807±0.022	0.81±0.021	0.803±0.023
	UnderBagging	0.512±0.039	0.584±0.039	0.603±0.031	0.578±0.034
	SMOTEBagging	0.663±0.066	0.667±0.059	0.672±0.059	0.656±0.061
	Cascade	0.798±0.061	0.779±0.054	0.784±0.055	0.774±0.058
	SPE	0.864±0.041	0.823±0.054	0.827±0.051	0.819±0.053
	MESA	0.859±0.024	0.815±0.023	0.807±0.031	0.805±0.021
	Algorithm 1 A	**0.891±0.025**	**0.833±0.038**	**0.846±0.021**	**0.832±0.022**
	Algorithm 1 B	0.877±0.023	0.828±0.024	0.823±0.024	0.817±0.027
XGB	UnderBagging	0.512±0.045	0.591±0.03	0.604±0.033	0.58±0.035
	SMOTEBagging	0.659±0.084	0.658±0.068	0.667±0.063	0.65±0.066
	Cascade	0.701±0.06	0.71±0.053	0.717±0.051	0.704±0.053
	SPE	0.732±0.073	0.725±0.068	0.732±0.063	0.717±0.068
	MESA	0.694±0.089	0.683±0.025	0.672±0.041	0.669±0.047
	Algorithm 1 A	**0.791±0.048**	**0.76±0.057**	**0.743±0.033**	**0.739±0.026**
	Algorithm 1 B	0.756±0.039	0.727±0.024	0.723±0.046	0.725±0.057

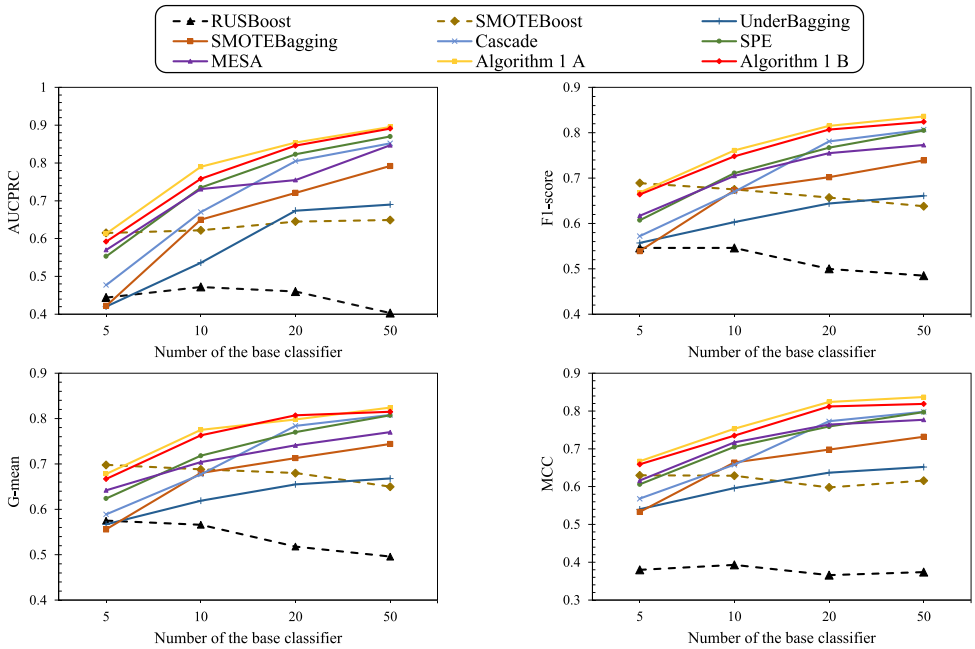

Fig. 7. Comparison of classification performance of the proposed framework and resampling ensemble frameworks with different number of base classifier on #49 datasets. (Best viewed in color).

so it is recommended to use this framework in combination with tree classification methods.

To investigate the effect of the number of base classifiers on the classification results, we choose DT as the base classifier and set the number of base classifiers as 5, 10, 20, and 50. The final experimental results are shown in Fig. 7. It can be seen that as the number of base classifiers increases, the classification performance also improves accordingly. The marginal effect of this improvement is obvious. For example, increasing the number of base classifiers from 20 to 50 is much less effective than increasing the number of base classifiers from 5 to 10, which will lead to a significant reduction in the computational efficiency of the algorithm. Hence, it is necessary to make trade-offs in practical applications. Moreover, the overall performance of the proposed framework is optimal for different numbers of base classifiers compared with benchmark methods, which also verifies the robustness of the proposed framework.

In addition, for the #49 dataset, Fig. 8 provides the performance of Algorithm 1 as the number of clusters is varied. When k is small, it has a greater impact on the results, and when k reaches 5, the experimental results are less sensitive to the increase of k. This is because the clustering results only indirectly affect the sampling weights. The overall performance of Algorithm 1 is relatively stable as k changes. In practical applications, the cluster number k can be adjusted according to the dispersion of minority samples D_{min}

$$D_{min}(k) = -\frac{\sum_{m=1}^{k} \frac{h_m}{|\mathcal{P}|} \log \frac{h_m}{|\mathcal{P}|}}{\log k} \tag{15}$$

where $\boldsymbol{H} = [h_1, h_2, \ldots, h_k]$ is the number of minority samples in each cluster. When D_{min} is smaller, the minority samples are more concentrated, and the performance is better.

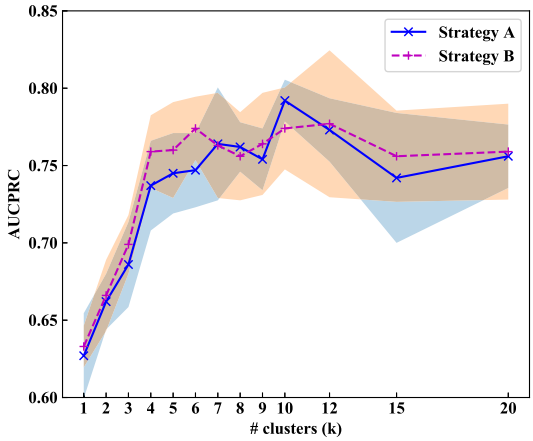

Fig. 8. Robustness to cluster number k on #49 datasets.

In actual industrial application scenarios, problems with different imbalance ratios may be encountered. To better analyze the relationship between imbalance ratios and classification performance, the experimental results of the proposed framework and other resampling ensemble frameworks using training sets with different imbalance ratios are recorded in Table VII. All ensemble frameworks integrate 10 DTs, and training sets with different imbalanced ratios are obtained by randomly undersampling the #49 dataset. Due to space limitations, only the $AUCPRC$ of the experimental results are shown. In general, the performance of all methods decreases as the imbalance ratio of the training set increases. Algorithm 1 Strategy A has the best overall performance, consistently performing best when IR is below 50 and only slightly behind SMOTEBoost when IR=50. The robustness of the proposed framework under different IR is also verified. In addition, with

TABLE VII
AUCPRC OF THE RESAMPLING ENSEMBLE METHODS ON #49 DATASET WITH DIFFERENT IMBALANCED RATIOS

Method	IR=20	IR=30	IR=40	IR=50
RUSBoost	0.472±0.093	0.342±0.07	0.266±0.066	0.198±0.075
SMOTEBoost	0.622±0.062	0.639±0.031	0.521±0.07	**0.519±0.057**
UnderBagging	0.536±0.097	0.547±0.101	0.353±0.097	0.326±0.05
SMOTEBagging	0.65±0.06	0.609±0.063	0.434±0.133	0.449±0.093
Cascade	0.67±0.087	0.648±0.092	0.453±0.086	0.351±0.105
SPE	0.735±0.065	0.625±0.096	0.455±0.065	0.425±0.088
MESA	0.731±0.091	0.639±0.057	0.480±0.090	0.427±0.040
Algorithm 1 A	**0.790±0.027**	**0.706±0.062**	**0.524±0.079**	0.496±0.090
Algorithm 1 B	0.758±0.058	0.696±0.094	0.510±0.086	0.481±0.067

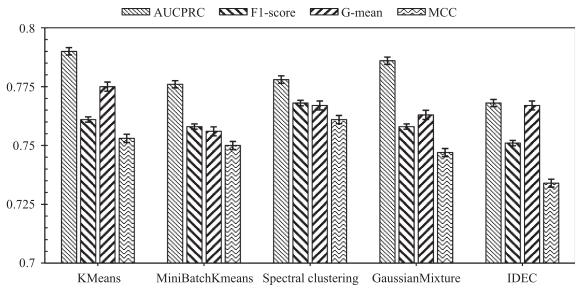

Fig. 9. Comparison of classification performance of the proposed framework with different clustering method on #49 datasets.

the increase of the IR, the method based on undersampling cannot obtain sufficient information due to the decreasing number of minority samples, and its performance is gradually inferior to the method based on oversampling. Therefore, if you want to use an undersampling ensemble framework to achieve good results, it is better to ensure that the training set contains a sufficient number of minority samples, usually more than 50 are recommended.

As shown in Algorithm 1, the proposed framework needs to define the state of the sampling decision process and determine the sampling strategy of each cluster based on the clustering results. So how to choose a suitable clustering method and whether different clustering methods will have a great impact on the final classification results, we need to conduct experimental verification. Five clustering methods are selected for this experiment, including Kmeans, MiniBatchKmeans, spectral clustering, Gaussian mixture, and improved deep embedded clustering (IDEC) [40]. The results of Algorithm 1 Strategy A in Fig. 9 show that the performance of different clustering methods is very close to each other, and combined with the contents of Tables III and IV, we can find that the classification performance of the proposed framework with different clustering methods is always better than the benchmark methods, among which KMeans has the best overall performance, and IDEC has slightly worse performance. This proves that the proposed framework is robust to different clustering methods, and in practice, different clustering methods can be used according to the nature of the dataset. Usually, KMeans is enough.

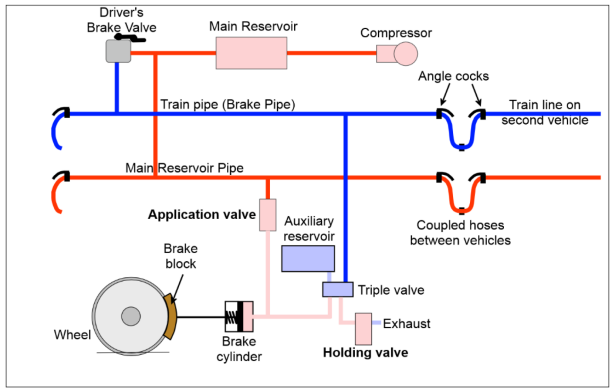

Fig. 10. Schematic of the pneumatic layout of a simple electro-pneumatic brake system. (http://www.railway-technical.com/trains/rolling-stock-index-l/train-equipment/brakes/electro-pneumatic-brakes-d.html).

C. Case 2: High-Speed Train Brake System Fault Detection

The braking system is a crucial mission subsystem for the safe operation of HST. Fig. 10 shows a schematic of the pneumatic layout of a simple electro-pneumatic brake system. This dataset comes from our project cooperation with Alstom. There are a total 44 monitored variables related to the braking system faults of HST. They include train-level conditions, e.g., speed, operation mode, external power supply, line voltage, line current, and braking system-level conditions, e.g., internal temperature, battery voltage, detected slip or slide, ED brake state, TCL brake state, achieved brake effort, etc. All the nominal variables are transformed into the numeric feature. After data transformation, the dataset contains 46 numeric features, 31 features are binary ones, and 15 features are continuous. In the braking system of HST, different faults may occur, e.g., stop cock emergency application valve closed, pneumatically brake stop cock closed on the bogie, stop cock locked, etc. In this experiment, the different faults are not distinguished, only the occurrence of faults is predicted. There are 22 368 samples in the processed HST dataset, including 389 faulty samples, and the IR of the dataset is 56.50:1. The classification performance of the proposed framework and all benchmark methods on the HST dataset are summarized in the Table VIII. Similar to the results of the previous wind turbine datasets, the overall performance of the resampling ensemble frameworks is better than the resampling methods. The proposed framework achieves the best performance on all evaluation indexes, which further verifies the superior effectiveness of the proposed framework in solving practical industrial imbalance problems. Unlike before, the overall performance of Strategy B is slightly better than Strategy A on the HST dataset, which may be due to that the accuracy of its training set does not reach a very high level during the training process, so Strategy B can provide more valid information for the sampler compared to Strategy A.

V. CONCLUSION

This article proposed a novel adaptive undersampling framework for the class-imbalance fault detection problems. The

TABLE VIII
CLASSIFICATION PERFORMANCE OF DIFFERENT METHODS ON HST DATASET

Method	*AUCPRC*	*F1-score*	*G-mean*	*MCC*
RUS	0.094±0.007	0.187±0.012	0.303±0.012	0.276±0.013
CC	0.031±0.002	0.061±0.004	0.175±0.005	0.116±0.008
ENN	0.294±0.012	0.532±0.011	0.535±0.012	0.526±0.012
Tomek	0.317±0.02	0.555±0.019	0.556±0.018	0.548±0.019
SMOTE	0.281±0.046	0.513±0.043	0.523±0.043	0.513±0.044
BSMOTE	0.241±0.016	0.478±0.017	0.482±0.017	0.472±0.018
ADASYN	0.278±0.017	0.511±0.016	0.521±0.016	0.511±0.017
SMOTEENN	0.213±0.017	0.427±0.018	0.455±0.019	0.442±0.02
SMOTETomek	0.266±0.022	0.498±0.023	0.509±0.023	0.498±0.023
RUSBoost	0.324±0.03	0.44±0.031	0.455±0.027	0.338±0.012
SMOTEBoost	0.565±0.042	0.619±0.026	0.624±0.025	0.605±0.038
UnderBagging	0.324±0.046	0.436±0.045	0.463±0.029	0.45±0.031
SMOTEBagging	0.519±0.034	0.581±0.025	0.583±0.026	0.576±0.026
Cascade	0.514±0.037	0.578±0.024	0.584±0.022	0.577±0.023
SPE	0.571±0.035	0.607±0.044	0.614±0.045	0.607±0.047
MESA	0.566±0.025	0.622±0.043	0.627±0.032	0.607±0.018
Algorithm 1 A	0.607±0.049	0.642±0.023	**0.652±0.029**	0.641±0.029
Algorithm 1 B	**0.610±0.047**	**0.653±0.024**	0.647±0.037	**0.643±0.026**

framework first used a clustering algorithm to perform cluster analysis on the original imbalanced dataset. The state of subsequent reinforcement learning decisions was based on the clustering results. The sampling process of the subtraining set of ensemble learning was modeled as an MDP. The sampler was optimized by the reinforcement learning method, thereby ensuring the rationality and divergence of the subtraining sets, and improving the performance of the ensemble framework. The reasonable definition of sampling states and the design of sampling strategy ensured that the optimization of the sampler could quickly converge and significantly improved the classification performance. The effectiveness of the proposed framework was verified by a series of experiments carried out on two cases: wind turbine blade cracking dataset and high-speed train brake system dataset. Experimental results showed that the proposed framework could effectively improve the ensemble's performance, and the classification results and robustness were significantly better than the benchmark method for imbalanced datasets.

The imbalance problem was very challenging in the field of fault detection, especially for the systems with high reliability in actual industrial applications. The current classification performance could be further improved, and the oversampling strategy could be added to the decision-making process in future research to improve the ability of processing complex real-world problems.

APPENDIX A
SAC FRAMEWORK

Standard RL maximizes the expected sum of rewards. The SAC will consider a more general maximum entropy objective, which favors stochastic policies by augmenting the objective with the expected entropy of the policy

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,\mu_t)\sim\rho_\pi}[r(s_t,\mu_t) + \alpha\mathcal{H}(\pi(\cdot|s_t))]. \quad (16)$$

The temperature parameter α determines the relative importance of the entropy term against the reward, thus controlling the optimal policy's stochasticity. This parameter can be adaptively adjusted by the method in [41].

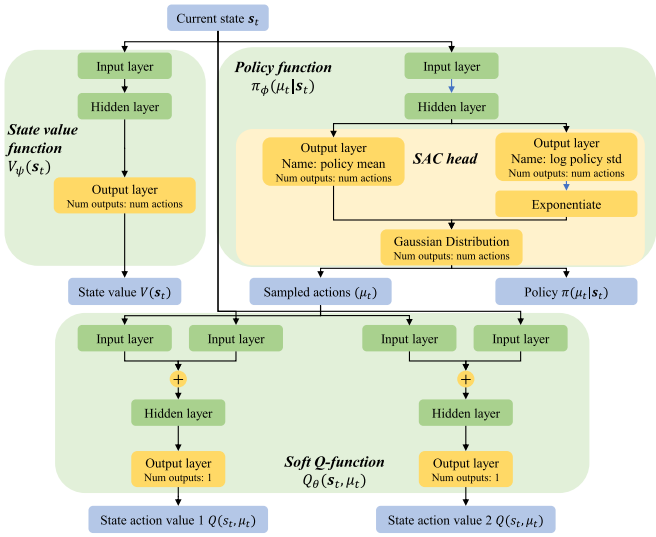

Fig. 11. SAC network structure. (Refer to the https://intellabs.github.io/coach/components/agents/policy_optimization/sac.html).

For each environment step, we get the action by the state s_t of current stage and the policy π_ϕ, $\mu_t \sim \pi_\phi(\mu_t|s_t)$. After μ_t is determined, the weight of each majority sample in the training set can be determined according to μ_t, and a new subtraining set $\mathcal{T}_{t+1}$ can be sampled. Use the new subtraining set to train a base classifier f_{t+1} to obtain the classification result and combine the classification results of the previous stage to compute the reward r_t and the new state s_{t+1}. So far we can get the tuple $(s_t, \mu_t, r_t, s_{t+1})$ of one step.

SAC algorithm can be derived starting from a maximum entropy variant of the policy iteration method [36]. It uses function approximators for both the Q-function and the policy, and instead of running evaluation and improvement to convergence, it alternates between optimizing both networks with stochastic gradient descent. We consider a parameterized state value function $V_\psi(s_t)$, soft Q-function $Q_\theta(s_t, \mu_t)$, and a tractable policy $\pi_\phi(\mu_t|s_t)$ as Fig. 11. The parameters of these networks are ψ, θ, and ϕ. Suppose the number of clusters is k, and the value function is modeled as a three-layer fully connected neural network ($\{k$-50-1$\}$). The soft Q-function is a two head three-layer fully connected neural network ($2\times\{k$+1-50-1$\}$). The policy is a Gaussian with mean and covariance given by a three-layer fully connected neural network ($\{k$-50-2$\}$). The objective functions and optimization strategies of each network can be found in Section IV-B of [36].

APPENDIX B
EXPERIMENT WITH SYNTHETIC DATASETS

To more intuitively demonstrate the effect of the proposed method, we conduct comparative experiments on the synthetic dataset. We create three 3×3 checkerboard datasets. Every dataset contains nine Gaussian components, shown as Fig. 12. By adjusting the variance and the number of samples in each Gaussian component, different datasets have different degrees of class overlap and intraclass imbalance.

TABLE IX
PERFORMANCE OF DIFFERENT ENSEMBLE METHODS ON SYNTHETIC CHECKERBOARD DATASETS

Method	Dataset (a)				Dataset (b)				Dataset (c)			
	AUCPRC	F1-score	G-mean	MCC	AUCPRC	F1-score	G-mean	MCC	AUCPRC	F1-score	G-mean	MCC
RUSBoost	0.723±0.015	0.75±0.013	0.751±0.012	0.669±0.013	0.52±0.027	0.587±0.02	0.601±0.019	0.493±0.026	0.551±0.029	0.621±0.02	0.626±0.02	0.508±0.01
SMOTEBoost	0.648±0.019	0.754±0.016	0.755±0.016	0.717±0.021	0.449±0.012	0.566±0.016	0.57±0.016	0.505±0.023	0.514±0.025	0.602±0.019	0.606±0.016	0.546±0.025
UnderBagging	0.702±0.02	0.742±0.017	0.743±0.016	0.717±0.018	0.508±0.021	0.582±0.02	0.595±0.021	0.547±0.019	0.535±0.034	0.623±0.022	0.628±0.022	0.589±0.025
SMOTEBagging	**0.781±0.01**	**0.778±0.011**	**0.779±0.011**	**0.757±0.012**	0.535±0.018	0.562±0.012	0.567±0.012	0.526±0.011	0.59±0.016	0.627±0.024	0.635±0.024	0.608±0.027
Cascade	0.719±0.014	0.756±0.009	0.757±0.009	0.734±0.01	0.512±0.019	0.57±0.021	0.579±0.02	0.532±0.022	0.561±0.033	0.611±0.024	0.615±0.02	0.575±0.026
SPE	0.724±0.008	0.757±0.008	0.758±0.007	0.735±0.007	0.533±0.027	0.586±0.016	0.592±0.015	0.546±0.018	0.598±0.012	0.637±0.022	0.64±0.022	0.605±0.023
MESA	0.751±0.013	0.762±0.011	0.760±0.014	0.737±0.012	0.574±0.016	0.603±0.014	0.609±0.017	0.549±0.015	0.604±0.023	0.642±0.014	0.636±0.010	0.609±0.018
Algorithm 1 A	0.759±0.013	0.767±0.009	0.771±0.01	0.749±0.016	0.618±0.014	**0.639±0.019**	**0.643±0.021**	0.598±0.009	0.659±0.013	0.683±0.021	**0.679±0.014**	0.654±0.019
Algorithm 1 B	0.761±0.014	0.765±0.009	0.768±0.011	0.744±0.012	**0.623±0.017**	0.637±0.022	0.639±0.015	**0.604±0.013**	**0.665±0.024**	**0.687±0.018**	0.674±0.021	**0.659±0.017**

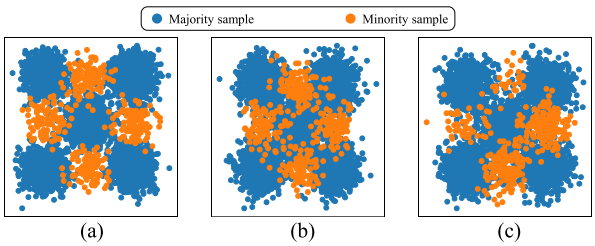

Fig. 12. Visualization of the synthetic checkerboard datasets. We set the number of majority samples and minority samples are 4000 and 400, respectively. Among them, (b) has more severe class overlap than (a), and (c) has intraclass imbalance of minority class compared to (b).

The experimental results are summarized in Table IX. For dataset (a), with low class overlap and no intraclass imbalance, the proposed Algorithm 1 has no clear advantage. On the contrary, SMOTEBagging achieves the best performance due to the properties of Gaussian distribution and low overlap of dataset classes. With the increasing degree of class overlap and the emergence of intraclass imbalance, the proposed Algorithm 1 exhibits clear advantages. The classification performance is much better than other benchmark methods on datasets (b) and (c). This proves that the proposed method is suitable for imbalanced classification problems with severe class overlap and intraclass imbalance, which are often the challenges faced in practical industrial fault detection.

REFERENCES

[1] H. Chen, Z. Chai, O. Dogru, B. Jiang, and B. Huang, "Data-driven designs of fault detection systems via neural network-aided learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 10, pp. 5694–5705, Oct. 2021.

[2] M. N. Nguyen, C. Bao, K. L. Tew, S. D. Teddy, and X.-L. Li, "Ensemble based real-time adaptive classification system for intelligent sensing machine diagnostics," *IEEE Trans. Rel.*, vol. 61, no. 2, pp. 303–313, Jun. 2012.

[3] E. Zio, "Some challenges and opportunities in reliability engineering," *IEEE Trans. Rel.*, vol. 65, no. 4, pp. 1769–1782, Dec. 2016.

[4] P. R. Bal and S. Kumar, "WR-ELM: Weighted regularization extreme learning machine for imbalance learning in software fault prediction," *IEEE Trans. Rel.*, vol. 69, no. 4, pp. 1355–1375, Dec. 2020.

[5] J. Liu, Y.-F. Li, and E. Zio, "A SVM framework for fault detection of the braking system in a high speed train," *Mech. Syst. Signal Process.*, vol. 87, pp. 401–409, 2017.

[6] Z. Wu, W. Lin, B. Fu, J. Guo, Y. Ji, and M. Pecht, "A local adaptive minority selection and oversampling method for class-imbalanced fault diagnostics in industrial systems," *IEEE Trans. Rel.*, vol. 69, no. 4, pp. 1195–1206, Dec. 2020.

[7] Z. Ding and L. Xing, "Improved software defect prediction using pruned histogram-based isolation forest," *Rel. Eng. Syst. Saf.*, vol. 204, 2020, Art. no. 107170.

[8] A. Ali, S. M. Shamsuddin, and A. L. Ralescu, "Classification with class imbalance problem: A review," *Int. J. Adv. Soft Compu. Appl*, vol. 7, no. 3, 2015.

[9] T. Zhang et al., "Intelligent fault diagnosis of machines with small & imbalanced data: A state-of-the-art review and possible extensions," *ISA Trans.*, vol. 119, pp. 152–171, 2021.

[10] D. A. Cieslak and N. V. Chawla, "Start globally, optimize locally, predict globally: Improving performance on imbalanced data," in *Proc. 8th IEEE Int. Conf. Data Mining*, 2008, pp. 143–152.

[11] W. Lu, Z. Li, and J. Chu, "Adaptive ensemble undersampling-boost: A novel learning framework for imbalanced data," *J. Syst. Softw.*, vol. 132, pp. 272–282, 2017.

[12] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*. Boca Raton, FL, USA: CRC Press, 2012.

[13] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Trans. Syst., Man, Cybern., Part B.*, vol. 39, no. 2, pp. 539–550, Apr. 2009.

[14] G. Kovács, "An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets," *Appl. Soft Comput.*, vol. 83, 2019, Art. no. 105662.

[15] T. Han, Y.-F. Li, and M. Qian, "A hybrid generalization network for intelligent fault diagnosis of rotating machinery under unseen working conditions," *IEEE Trans. Instrum. Meas.*, vol. 70, 2021, Art. no. 3520011.

[16] K. Yan, J. Su, J. Huang, and Y. Mo, "Chiller fault diagnosis based on VAE-enabled generative adversarial networks," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 1, pp. 387–395, Jan. 2022.

[17] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications," *Expert Syst. Appl.*, vol. 73, pp. 220–239, 2017.

[18] A. E. Prosvirin, F. Piltan, and J.-M. Kim, "Hybrid rubbing fault identification using a deep learning-based observation technique," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 11, pp. 5144–5155, Nov. 2020.

[19] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Trans. Neural Netw.*, vol. 22, no. 2, pp. 199–210, Feb. 2011.

[20] T. Han, C. Liu, W. Yang, and D. Jiang, "Deep transfer network with joint distribution adaptation: A new intelligent fault diagnosis framework for industry application," *ISA Trans.*, vol. 97, pp. 269–281, 2020.

[21] W. Qian, S. Li, and X. Jiang, "Deep transfer network for rotating machine fault analysis," *Pattern Recognit.*, vol. 96, 2019, Art. no. 106993.

[22] Q. Kang, L. Shi, M. Zhou, X. Wang, Q. Wu, and Z. Wei, "A distance-based weighted undersampling scheme for support vector machines and its application to imbalanced classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 9, pp. 4152–4165, Sep. 2018.

[23] L. Gong, S. Jiang, L. Bo, L. Jiang, and J. Qian, "A novel class-imbalance learning approach for both within-project and cross-project defect prediction," *IEEE Trans. Rel.*, vol. 69, no. 1, pp. 40–54, Mar. 2020.

[24] M. Qian and Y.-F. Li, "A weakly supervised learning-based oversampling framework for class-imbalanced fault diagnosis," *IEEE Trans. Rel.*, vol. 71, no. 1, pp. 429–442, Mar. 2022.

[25] P. Wu, J. Liu, and F. Shen, "A deep one-class neural network for anomalous event detection in complex scenes," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 7, pp. 2609–2622, Jul. 2019.

[26] R. Barandela, R. M. Valdovinos, and J. S. Sánchez, "New applications of ensembles of classifiers," *Pattern Anal. Appl.*, vol. 6, no. 3, pp. 245–256, 2003.

[27] W. Feng, W. Huang, and J. Ren, "Class imbalance ensemble learning based on the margin theory," *Appl. Sci.*, vol. 8, no. 5, 2018, Art. no. 815.

[28] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "Smoteboost: Improving prediction of the minority class in boosting," in *Proc. Eur. Conf. Princ. Data Mining Knowl. Discov.*, 2003, pp. 107–119.

[29] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "Rusboost: A hybrid approach to alleviating class imbalance," *IEEE Trans. Syst., Man, Cybern.-Part A: Syst. Humans*, vol. 40, no. 1, pp. 185–197, Jan. 2010.
[30] Z. Liu et al., "Self-paced ensemble for highly imbalanced massive data classification," in *Proc. IEEE 36th Int. Conf. Data Eng.*, 2020, pp. 841–852.
[31] Z. Liu et al., "Mesa: Boost ensemble imbalanced learning with meta-sampler," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 14463–14474.
[32] S. Wang, L. L. Minku, and X. Yao, "Resampling-based ensemble methods for online class imbalance learning," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 5, pp. 1356–1368, May 2015.
[33] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation, and active learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 7, 1994.
[34] M. A. Arefeen, S. T. Nimi, and M. S. Rahman, "Neural network-based undersampling techniques," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 52, no. 2, pp. 1111–1120, Feb. 2022.
[35] M. Peng et al., "Trainable undersampling for class-imbalance learning," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, pp. 4707–4714.
[36] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.
[37] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 559–563, 2017.
[38] W.-C. Lin, C.-F. Tsai, Y.-H. Hu, and J.-S. Jhang, "Clustering-based undersampling in class-imbalanced data," *Inf. Sci.*, vol. 409–410, pp. 17–26, 2017.
[39] X.-Y. Liu and Z.-H. Zhou, "Ensemble methods for class imbalance learning," *Imbalanced Learning: Foundations, Algorithms, and Applications*, Wiley, 2013, ch. 4, pp. 61–82, doi: 10.1002/9781118646106.ch4.
[40] X. Guo, L. Gao, X. Liu, and J. Yin, "Improved deep embedded clustering with local structure preservation." in *Proc. 26th Int. Joint Conf. Artif. Intell.*, 2017, pp. 1753–1759.
[41] T. Haarnoja et al., "Soft actor-critic algorithms and applications," 2018, *arXiv:1812.05905*.

Min Qian (Student Member, IEEE) was born in 1997. He received the B.E. degree in industrial engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2019. He is currently working toward the Ph.D. degree in management science and engineering from Tsinghua University, Beijing, China.

His current research interests include imbalance learning, fault diagnosis, and machine learning.

Yan-Fu Li (Senior Member, IEEE) received the B.S. degree in software engineering from Wuhan University, Wuhan, China, in 2005, the Ph.D. degree in industrial and systems engineering from the National University of Singapore, Singapore, in 2010.

He was a Faculty Member with the Laboratory of Industrial Engineering, CentraleSupélec, University of Paris-Saclay, Gif-sur-Yvette, France, from 2011 to 2016. He is currently a Professor with Industrial Engineering Department, Tsinghua University, Beijing, China. He has led/participated in several projects supported by EU, France, and Chinese governmental funding agencies and various industrial partners. He has coauthored over 100 publications, on international journals, conference proceedings, and books. His current research interests include reliability, availability, maintainability, safety (RAMS) assessment and optimization with the applications onto various industrial systems.

Dr. Li is an Associate Editor for IEEE TRANSACTIONS ON RELIABILITY.