Dynamic Ensemble Framework for Imbalanced Data Classification

Tuanfei Zhu, Xingchen Hu, Xinwang Liu, *Senior Member, IEEE*, En Zhu, Xinzhong Zhu, and Huiying Xu

***Abstract*—Dynamic ensemble has significantly greater potential space to improve the classification of imbalanced data compared to static ensemble. However, dynamic ensemble schemes are far less successful than static ensemble methods in the imbalanced learning field. Through an in-depth analysis on the behavior characteristics of dynamic ensemble, we find that there are some important problems that need to be addressed to release the full potential of dynamic ensemble, including but not limited to, correcting the component classifiers' bias towards the majority classes, increasing the proportions of the positive classifiers (i.e., the component classifiers making correct prediction) for difficult samples, and providing the accurate competence estimations on the hard-to-classify samples w.r.t the classifier pool. Inspired by these, we propose a Dynamic Ensemble Framework for imbalanced data classification (imDEF). imDEF first uses the data generation method OREM_{G} to generate multiple artificial synthetic datasets, which have diverse class distributions by rebalancing the original imbalanced data. Based on each of such synthetic datasets, imDEF then utilizes a Classification Error-aware Self-Paced Sampling Ensemble (SPSE_{CE}) method to gradually focus more on difficult samples, to create a low-biased classifier pool and increase the proportions of the positive classifiers for the difficult samples. Finally, imDEF constructs a referee system to achieve the competence estimations by leveraging an Ensemble Margin-aware Self-Paced Sampling Ensemble (SPSE_{EM}) method. SPSE_{EM} incrementally strengthens the learning of the hard-to-classify samples, so that the competent levels of component classifiers could be estimated accurately. Extensive experiments demonstrate the effectiveness of imDEF. The source codes have been made publicly available on GitHub.**

***Index Terms*—Class imbalance problems, oversampling, multiclass imbalance, ensemble learning, dynamic ensemble.**

Received 20 April 2024; revised 31 October 2024; accepted 7 January 2025. Date of publication 13 January 2025; date of current version 25 March 2025. This work was supported in part by the National Science Fund for Distinguished Young Scholars of China under Grant 62325604, in part by the Special Funds of the National Natural Science Foundation of China under Grant 62441618, in part by the National Natural Science Foundation of China under Grant 62276271 and Grant 62376279, and in part by the Natural Science Foundation of Hunan Province, China, under Grant 2024JJ5051. Recommended for acceptance by R. Akbarinia. *(Corresponding author: Xingchen Hu.)*

Tuanfei Zhu is with the College of Computer Science and Engineering, Changsha University, Changsha 410073, China, and also with the School of Computer, National University of Defense Technology, Changsha 410073, China.

Xingchen Hu, Xinwang Liu, and En Zhu are with the School of Computer, National University of Defense Technology, Changsha 410073, China (e-mail: xhu4@ualberta.ca; xinwangliu@nudt.edu.cn).

Xinzhong Zhu and Huiying Xu are with the College of Computer Science and Technology, Zhejiang Normal University, Jinhua 321017, China.

Source codes are available at https://github.com/imbLearning/imDEF.

This article has supplementary downloadable material available at https://doi.org/10.1109/TKDE.2025.3528719, provided by the authors.

Digital Object Identifier 10.1109/TKDE.2025.3528719

I. Introduction

A. Background

IMBALANCED data classification occurs when some classes (i.e., majority classes) overwhelm the others (i.e., minority classes) in sample size. On the one hand, class imbalance could be entangled with data complexity factors, significantly deteriorating the difficulty of learning [1]. Standard classification algorithms are usually incapable of combating class imbalance problems, and exhibit undesired performance in the minority classes. On the other hand, the learning from imbalanced data is prevalent in real-world applications, which the correct prediction of minority samples is often a critical requirement in reality (e.g., disease diagnosis, defect detection). Under the widespread prevalence of imbalanced learning problems, a great realistic research need is formed between the importance of correctly identifying minority samples and the propensity of misclassifying minority samples.

Over the past two decades, a large number of imbalanced learning techniques have been proposed. Among them, imbalanced ensemble algorithms are considered as one of the most effective types of methods [2], [3]. Existing ensemble solutions could be divided into imbalanced static ensemble methods and dynamic ensemble schemes. Static ensemble approaches always use all component classifiers to classify a test sample [2]. By contrast, dynamic ensemble schemes first estimate the competent levels of component classifiers on the considered test sample, then select those most competent classifiers to predict [4], [5]. Apparently, to produce correct collective decision, static ensemble requires the component classifiers making correct prediction must occupy a majority of the vote weights. When dealing with imbalanced classification problems, such a decision-making way is prone to performance bottleneck. The reason behind is that the component classifiers often have a natural bias towards the majority classes, making it highly likely that the minority samples could only be correctly classified by a small percentage of the component classifiers. Unlike static ensemble, dynamic ensemble schemes could produce the correct collective decision as long as there is one component classifier that correctly predicts the test sample. Therefore, dynamic ensemble solutions have significantly greater potential to improve the classification performance of imbalanced data than imbalanced static ensemble methods.

However, imbalanced static ensemble methods are dominant in existing ensemble solutions [3], [6], [7], [8], [9], [10], and

TABLE I
SUMMARY OF MAJOR MATHEMATICAL NOTATIONS

Notations	Mathematical Meanings	Notations	Mathematical Meanings
$\boldsymbol{Tr}$	training dataset	$\boldsymbol{Te}$	testing dataset
$\boldsymbol{X_{tr}}$	feature components of training dataset	Y_{tr}	label of training dataset
$\boldsymbol{Tr_r}$	r-th synthetic dataset	$\boldsymbol{dp_r}$	class distribution of $\boldsymbol{Tr_r}$
$\{\boldsymbol{Tr_r}\}_{r=1}^{\lfloor\sqrt{n}\rfloor}$	set of $\lfloor\sqrt{n}\rfloor$ synthetic datasets	$\mathcal{P}$	classifier pool
$\mathcal{P}_r$	classifier ensemble created by SPSE$_{\mathrm{CE}}$ on $\boldsymbol{Tr_r}$	$\boldsymbol{CL}$	competence label set of the component classifiers
C_t	t-th component classifier	$\mathcal{RC}_t$	referee committee associated with C_t
$\mathcal{RS}$	referee system	$\mathcal{H}(x_i, y_i, \mathcal{P}_r)$	x_i's classification error w.r.t $\mathcal{P}_r$
$\mathcal{GC}^g$	g-th referee collective	$\hat{\mathcal{H}}(x_i, y_i, \mathcal{P}_r)$	normalized $\mathcal{H}(x_i, y_i, \mathcal{P}_r)$
$\mathcal{M}_0(x_i, \boldsymbol{CL})$	x_i's initial Negative dynamic Ensemble Margin (NEM) w.r.t $\boldsymbol{CL}$	R_t^g	referee classifier constructed for C_t in g-th referee collective
$\mathcal{M}_g(x_i, \mathcal{GC}^g, \boldsymbol{CL})$	x_i's NEM w.r.t $\mathcal{GC}^g$ and $\boldsymbol{CL}$	$\mathcal{CM}(x_i)$	x_i's Cumulative NEM (CNEM)
$\mathcal{C}\hat{\mathcal{M}}(x_i)$	normalized $\mathcal{C}\hat{\mathcal{M}}(x_i)$	$C_t(x, l_k)$	the probability that C_t predicts x into l_k

TABLE II
AN EXAMPLE DEMONSTRATING THE SAMPLE IMPORTANCE FOR THE CONSTRUCTION OF REFEREE SYSTEM

	ID	C_1	C_2	C_3	C_4	C_5
A	Sample 1	1	1	1	1	1
	Sample 2	1	1	1	0	1
	Sample 3	1	1	0	1	1
B	Sample 4	0	0	0	1	1
	Sample 5	0	1	0	0	1

0 (/1) denotes the wrong (/correct) prediction.

there is no credible evidence to show that imbalanced dynamic ensemble schemes are better than static ensemble solutions.

In this study, we first conduct a comprehensive experiment to investigate the classification behavior characteristics of conventional dynamic ensemble in handling class imbalance problems, so as to reap valuable observations (or phenomenon). Then, we dig the rallying points for improving dynamic ensemble schemes according to these observations, and discuss the shortcomings of existing dynamic ensemble solutions. Finally, we propose an effective dynamic ensemble framework for imbalanced data classification.

B. Classification Behavior Analysis of Dynamic Ensemble

To investigate the behavior characteristics of dynamic ensemble, we use a series of representative dynamic ensemble schemes to classify the imbalanced datasets in Table III. Dynamic ensemble generally involves three stages when predicting a test sample: 1) generating a classifier pool; 2) estimating the competent level of each component classifier based on the Region of Competence (RoC[1]) of this sample, and then selecting the most competent classifiers; 3) yielding a collective decision by combining all of the selected classifiers. We form the following dynamic ensemble schemes: the most commonly used Bagging [11] and AdaBoost.M2 [12] are applied to generate classifier pool, respectively (stage 1); the popular Dynamic Selection Techniques (DSTs) MCB [13], KNORAE (KNE) [5], KNORAU(KNU) [5], desRRC [14], desP [15], desKL [15], KnoP [16] are used to select the component classifiers, respectively (stage 2); and the selected classifiers are weighted by their corresponding competent levels (stage 3).

The dynamic ensemble schemes are performed 10 times on each dataset with stratified 5-fold cross validation. In each running, 4 of the 5 folds are used as both the training data and Dynamic SELection data (DSEL[2]), the remaining fold is treated as the test data D_{test}. For convenience, we call the component classifiers providing the correct (/wrong) prediction for a test sample the positive (/negative) classifiers of this sample.

Intuitively, the proportion of the positive classifiers within classifier pool would significantly affect whether the corresponding test sample is classified correctly. To investigate the classification performance of dynamic ensemble in a fine-grained way, we divide the samples of D_{test} into 5 segments, $\{Te_s\}_{s=1}^5$, depending on their proportions of positive classifiers: $Te_1 = \{x_i | 0 \le p(x_i) \le 0.15, x_i \in D_{test}\}$, $Te_2 = \{x_i | 0.15 < p(x_i) \le 0.3, x_i \in D_{test}\}$, $Te_3 = \{x_i | 0.3 < p(x_i) \le 0.5, x_i \in D_{test}\}$, $Te_4 = \{x_i | 0.5 < p(x_i) \le 0.7, x_i \in D_{test}\}$, $Te_5 = \{x_i | 0.7 < p(x_i) \le 1, x_i \in D_{test}\}$, where $p(x_i)$ denotes the proportion of x_i's positive classifiers in the classifier pool.

For each of $\{Te_s\}_{s=1}^5$, we introduce the following statistics to explore the behavior characteristics of dynamic ensemble:

- $Pmaj_s = \frac{|MA_s|}{|MA|}$, where MA_s and MA are the majority sample set in Te_s and D_{test}, respectively. This statistic represents the proportion of the majority samples distributed into Te_s.
- $Pmin_s = \frac{|MI_s|}{|MI|}$, where MI_s and MI are the minority sample set in Te_s and D_{test}, respectively. It is similar to $Pmaj_s$.
- $Nmaj_s = \frac{1}{|MA_s|} \sum_{x_i \in MA_s} \frac{|MI_s \cap NN_k(x_i)|}{|NN_k(x_i)|}$, where $NN_k(x_i)$ is x_i's k-nearest neighbors in DSEL ($k = 7$

[1]RoC is usually a small region surrounding the considered test sample. The competent levels of component classifiers are evaluated by measuring how well they classify the samples in this region.

[2]DSEL is a dataset in which the RoC of the test sample are defined.

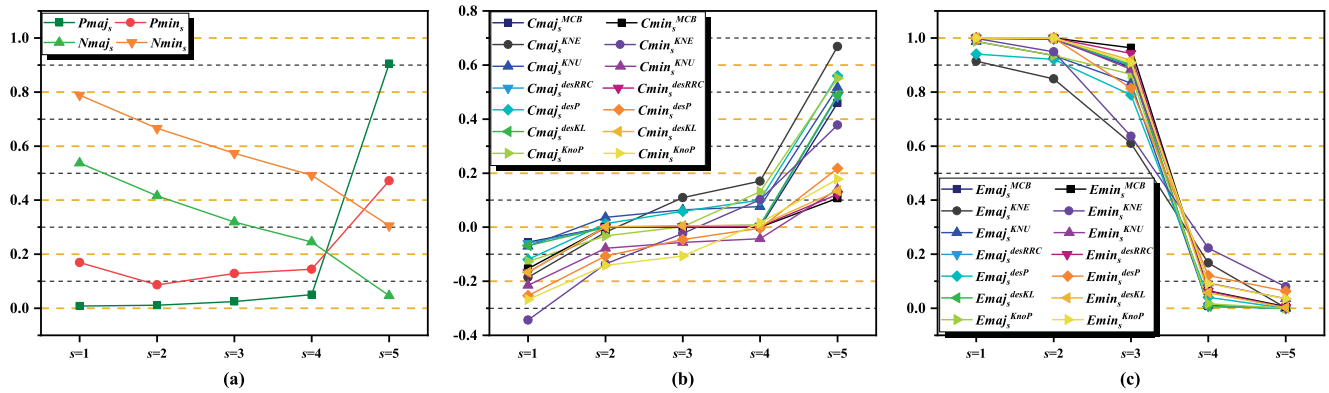

Fig. 1. The behavior characteristics of the dynamic ensemble schemes with Bagging to generate the classifier pool. (a), (b), and (c) show the average values of the statistics $Pmaj_s$, $Pmin_s$, $Nmaj_s$, $Nmin_s$, $Cmaj_s^{***}$, $Cmin_s^{***}$, $Emaj_s^{***}$ and $Emin_s^{***}$ over the datasets of Table III.

here). For the samples of MA_s, this statistic measures the average percentage of the minority samples within their k-nearest neighbors.

- $Nmin_s = \frac{1}{|MI_s|}\sum_{x_i \in MI_s} \frac{|MA_s \cap NN_k(x_i)|}{|NN_s(x_i)|}$. It is $Nmaj_s$'s counterpart. The definitions of $Nmaj_s$ and $Nmin_s$ are similar to the difficulty measure k-Disagreeing Neighbors [17]. They can measure the difficulty levels of the majority and minority samples, respectively.
- $Cmaj_s^{***} = \frac{1}{|MA_s|}\sum_{x_i \in MA_s} \frac{pc(x_i)-nc(x_i)}{pc(x_i)+nc(x_i)}$, where $pc(x_i)$ ($/nc(x_i)$) represents the average competent level of x_i's positive (/negative) classifiers under certain DST. The higher this statistic, the more accurate the competent estimation of component classifiers when predicting the samples in MA_s.
- $Cmin_s^{***} = \frac{1}{|MI_s|}\sum_{x_i \in MI_s} \frac{pc(x_i)-nc(x_i)}{pc(x_i)+nc(x_i)}$. It is similar to $Cmaj_s^{***}$. The higher this statistic, the more accurate the competence estimation when predicting the samples in MI_s.
- $Emaj_s^{***}$ and $Emin_s^{***}$ are the misclassification rates on MA_s and MI_s under certain DST, respectively.

Fig. 1 illustrates the average results of these statistics over the datasets of Table III, where the classifier pool is generated by Bagging. Due to space limitations, the results using AdaBoost.M2 to create classifier pool are shown in Fig. S1 of the supplementary material. Note that the behavior characteristics reflected by Fig. 1 and Fig. S1 are highly similar, and the detailed experimental results are provided in Tables S1 and S2 of the supplementary material. From this experiment, we can obtain the following observations.

Observation i. In dynamic ensemble, a DSEL is required to provide RoC for the test samples. Given the rarity of minority samples, existing dynamic ensemble schemes usually use the training data itself as DSEL. However, this would inevitably cause the overestimation of competent levels, because the samples are used both for training the component classifiers and evaluating their competent levels.

Observation ii. The minority samples are more frequently distributed in the segments with low proportional positive classifiers than the majority samples (see Fig. 1(a), $Pmin_s > Pmaj_s$ when $s \leq 4$). It indicates that the minority samples are easier to be misclassified, and the component classifiers have a bias towards the majority class.

TABLE III
DESCRIPTION OF CHARACTERISTICS OF EXPERIMENTAL TWO-CLASS DATASETS

Dataset	Minority Class	Majority Class	Class Distribution	|F|	IR
Diabetes	'1'	'0'	268/500	8	1.87
Ionosphere	'b'	'g'	126/252	34	2.00
BreastTissue	'car'&"con'	others	35/71	9	2.03
ForestTypes$_1$	'h'&'o'	others	169/354	27	2.09
Yeast$_1$	'NUC'	others	429/1055	8	2.46
Meters-B$_1$	'2'	others	24/68	51	2.83
Vehicle	'opel'	others	212/634	18	2.99
Wifi$_1$	'2'	others	500/1500	7	3.00
Wifi$_2$	'3'	others	500/1500	7	3.00
Voice3	'1'	others	52/158	10	3.04
Parkinsons	'0'	'1'	48/147	22	3.06
Laryngeal-3	'3'	others	82/271	16	3.30
Meters-B$_2$	'1'	others	19/73	51	3.84
Voice9$_1$	'2'&'8'&'9'	'1'&'5'& '6'&'7'	81/332	10	4.10
Vertebral$_T$	'hernia'	others	60/250	6	4.17
Ecoli	'om'&'omL' &'imL'&'imS'	others	64/272	7	4.25
Vowel$_1$	'9'&'1'	others	180/810	10	4.50
ForestTypes$_2$	'h'	others	86/437	27	5.08
Win-red	'4'&'7'	'5'&'8'	252/1347	11	5.35
LEV$_T$	'0'&'4'	others	120/880	4	7.33
ERA$_1$	'7'&'8'	others	119/881	4	7.40
Voice9$_2$	'9'&'2'&'4'	others	50/378	10	7.56
ESL$_T$	'3'&'8'	others	57/431	4	7.56
ERA$_2$	'1'&'9'	others	110/890	4	8.09
Vowel$_2$	'0'	others	90/900	10	10.00
Vowel$_3$	'10'	others	90/900	10	10.00
Car	'good'& 'v-good'	others	134/1594	6	11.90
Laryngeal-2	'0'	'1'	53/637	16	12.02
Cervical cancer	'1'	'0'	55/805	35	14.64
Yeast$_2$	'EXC'&'VAC'	others	65/1419	8	21.83

|F| and IR denote feature dimension and imbalanced ratio, respectively.

Observation iii. The samples with lower proportions of positive classifiers have higher $Nmin_s$ and $Nmaj_s$ (see Fig. 1(a), $Nmin_s$ and $Nmaj_s$ increase as s becomes smaller), which means they have higher difficulty levels. It can be understood as

more difficult samples tend to be correctly predicted by fewer classifiers.

Observation iv. Compared to the positive classifiers, DSTs might assign higher competent levels to the negative classifiers, especially for those difficult samples (see Fig. 1(b), both $Cmaj_s^{xxx}$ and $Cmin_s^{xxx}$ are usually smaller than 0 over $s \leq 3$). We call this phenomenon the competence conflict problem. In existing DSTs, the competent level of a component classifier is estimated based on a local neighborhood around the test sample (i.e., RoC). Given that the RoC of a difficult minority (/majority) sample is often dominated by the majority (/minority) samples, the component classifiers, which classify all samples in the RoC of the difficult minority (/majority) sample into the majority (/minority) class, would naturally obtain highly competent levels, even if they misclassify this difficult sample itself. On the contrary, the classifiers correctly identifying this difficult sample might acquire low competent levels due to the propensity of mispredicting its neighbors.

Observation v. The samples with lower proportions of positive classifiers have a higher likelihood of being misclassified (see Fig. 1(c), both $Emaj_s^{xxx}$ and $Emin_s^{xxx}$ are increased as s decreases). Two reasons could account for this observation. First, these samples require more accurate competence estimations. Small errors on the competent levels of component classifiers could result in wrong collective decisions on them (see Section III-C1 for the details). Second, the lower the proportion of positive classifiers, the higher the difficulty level according to *Observation iii*. DSTs might deteriorate the classification of the difficult samples due to the competence conflict problem.

C. Rallying Points of Improving Dynamic Ensemble Schemes

Based on the observations above, addressing the following problems can serve as the rallying points for improving the performance of dynamic ensemble in handling imbalanced data classification.

Problem i. The overestimation of competent levels should be mitigated according to *Observation i*, as it increases the risk that the incompetent classifiers are selected for prediction.

Problem ii. From *Observation ii*, imbalanced class distribution should be handled, so that the bias toward the majority classes could be corrected in classifier pool.

Problem iii. The learning of difficult samples should be reinforced in the process of training component classifiers to increase their number of positive classifiers. According to *Observations iii* and *v*, the difficult samples have a low proportion of positive classifiers, and the lower the proportion of positive classifiers, the higher the classification errors.

Problem iv. The competent conflict problem should be avoided based on *Observation iv*. It could aggravate the classification of difficult samples during dynamic prediction, while the correct classification of these samples is the key for obtaining a highly effective classification system.

Problem v Inspired by *Observation v*, the reliable competence estimations should be provided when predicting the samples with low proportional positive classifiers (i.e., those hard-to-classify samples w.r.t the classifier pool).

D. Limitations of Existing Dynamic Ensemble Solutions

Conventional dynamic ensemble schemes are generally modified in three aspects to combat class imbalance problems: 1) Imbalanced static ensemble methods are used to generate low-biased classifier pool; 2) resampling techniques are applied to rebalance DSEL, so that the minority samples appear more frequently in the RoC of the test sample; 3) skew-insensitive DSTs are designed to accommodate imbalanced DSEL (note: more details can be found in Section S2.B of the supplementary material). The first modification is to copy with *Problem ii*. The latter two modifications aim to emphasize the importance of minority samples in the competent level estimation of component classifiers. However, the *Problems i*, *iii*, *iv*, *v* mentioned above have not been addressed by existing dynamic ensemble solutions, thereby seriously depreciating the performance of dynamic ensemble way in learning imbalanced data.

E. Motivation and Contributions

Considering that dynamic ensemble has the substantial potential to enhance the performance of imbalanced data classification, while existing imbalanced dynamic ensemble schemes still have significant limitations, we propose an imbalanced Dynamic Ensemble Framework (imDEF) to release the full potential of dynamic ensemble by solving *Problems i*, *ii*, *iii*, *iv*, *and v*.

Our major contributions are highlighted as follows:

a) We deeply analyze the classification behavior characteristics of dynamic ensemble when copying with imbalanced data, and dig out several rallying points for improving dynamic ensemble schemes.
b) We use a data generation method OREM_G to create artificial synthetic datasets from original training data. The generated synthetic datasets and original training data are utilized to train the component classifiers and learn their competence regions, respectively (alleviating *Problem i*). In addition, the synthetic datasets are generated with diverse class distributions by reducing majority samples and increasing minority samples, so as to correct the bias towards the majority classes in the classifier pool (overcoming *Problem ii*).
c) We design a classification error-aware self-paced sampling ensemble SPSE_CE to create the component classifiers based on each synthetic dataset. SPSE_CE uses a self-paced procedure to gradually enhance the learning of the samples with high difficulty levels, thus increasing the number of positive classifiers for the difficult test samples (solving *Problem iii*).
d) Our imDEF builds a referee committee for each component classifier to accomplish its competence estimation, rather than using a neighborhood-based way (avoiding *Problem iv*). Specifically, an ensemble margin-aware self-paced sampling ensemble SPSE_EM is designed to construct the referee system based on the original training data. A self-paced procedure is utilized in SPSE_EM to

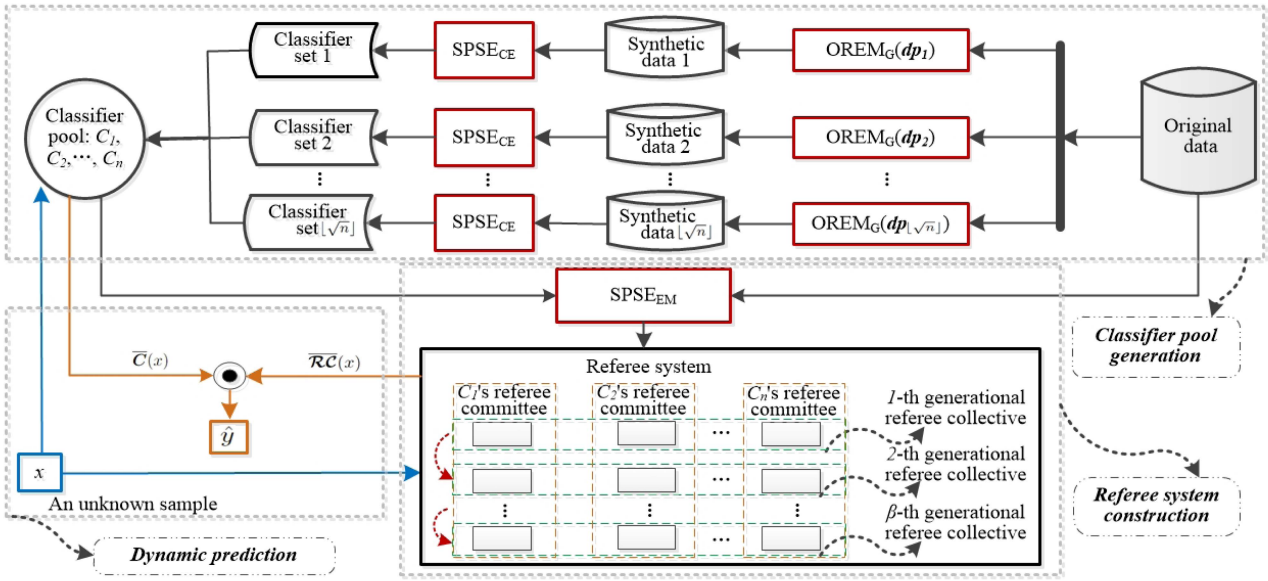

Fig. 2. The overall workflow of imDEF. imDEF contains classifier pool generation, referee system construction, and dynamic prediction. $\overline{C}(x)$ is the prediction vector of all component classifiers on a test sample x. $\overline{\mathcal{RC}}(x)$ is the competence level vector of all component classifiers w.r.t x. The prediction of each component classifier is weighted by its corresponding competent level to obtain the final result of x (i.e., $\hat{y}$).

gradually focus on the samples with large negative ensemble margins, so that the learning of hard-to-classify samples could be emphasized, and more accurate competence estimations could be provided for the hard samples (addressing $Problem\ v$).

e) The effectiveness of imDEF is evaluated on 30 two-class and 18 multiclass imbalanced datasets, respectively. The experimental results demonstrate that imDEF can statistically outperform both state-of-the-art static and dynamic ensemble solutions in terms of F1 (/marco-F1), G-mean (/MG), and AUC (/MAUC).

II. Related Works

Given that ensemble solutions are our interest, we only provide here a brief overview of imbalanced static ensemble methods and dynamic ensemble schemes. A detailed review on existing ensemble solutions are presented in Section S2 of the supplementary material.

Standard ensemble algorithms adhere to an accuracy-oriented design. They need to be combined with resampling techniques or cost-sensitive methods to combat imbalance problems, which leads to two types of static ensemble solutions, i.e., resampling techniques-combined ensemble [2], [18] and cost-sensitive ensemble [10]. In addition, some works veered away from standard ensemble learning frameworks, and develop innovative imbalanced ensemble algorithms by utilizing intelligent optimization techniques [19], deep learning approaches [20], or self-paced learning strategies [3], [21]. In Table S3 of the supplementary material, we summarize five main types of static ensemble solutions, where their ideas and major flaws are listed.

In dynamic ensemble, a key issue is how to dynamically select the component classifiers to predict a test sample, i.e., the design of DSTs. Existing DSTs include two main steps: 1) finding the RoC in DSEL for the considered test sample; 2) evaluating the competent level of each component classifier based on the RoC by utilizing a competence measure, then selecting those most competent classifiers to predict. To facilitate the presentation and comparison of existing DSTs, Table S4 of the supplementary material summarizes the most representative ones, including their process descriptions, and RoC definitions. A DST can be combined with a static ensemble method to form a dynamic ensemble scheme, where the static ensemble method is used to generate classifier pool.

To deal with imbalanced data classification, traditional dynamic ensemble schemes are improved by constructing the classifier pool with low bias, modifying imbalanced DSEL, or designing skew-insensitive DSTs. In [22], [23], [24], the classifier pools are generated by using the imbalanced static ensemble methods BRF [25], EasyBoost [7], and SMOTEBoost [6] to alleviate the bias of component classifiers. In [22], [24], [26], DSEL is balanced via resampling techniques such as RUS, RAMO [8], and RB [9]. Note that the balanced class distribution in DSEL is important for handling class imbalance problems. The reason is that imbalanced DSEL could result in the majority samples have significantly high probabilities of occurring in the RoC of the test samples, then DSTs would prefer to select those component classifiers which perform well on the majority classes. Hence, the ensemble performance in the minority classes would be depreciated. In addition, a few DSTs have been proposed to accommodate the learning from imbalanced data. For example, DES-MI [27] counts the frequency of occurrence for the samples of each class within RoC, and assigns higher weights to the lower-frequency classes. Then, the competent level of a component classifier is the weighted accuracy over the samples in RoC. RMkNN [22] uses a distance reduction function to pull the minority samples closer to the test sample, so as to increase the number of minority samples in RoC.

III. Method

imDEF has three stages: generation of classifier pool, construction of referee system, and dynamic prediction. We first provide an overview of imDEF, then introduce each stage of imDEF in detail.

For ease of reference, Table I lists major notations used throughout this paper along with their mathematical meanings.

A. Framework Overview

Fig. 2 shows the workflow of imDEF, and the pseudocode of imDEF is presented in Algorithm 1.

In the stage of classifier pool generation (lines 1–7 of Algorithm 1), the original training data is fed to OREM_G to yield $\lfloor\sqrt{n}\rfloor$[3] synthetic datasets with different class distributions (lines 2–3). Based on each of the generated synthetic datasets, SPSE_CE is used to create a classifier set of size about $n/\lfloor\sqrt{n}\rfloor$ (lines 4–5). These classifier sets are eventually united into a classifier pool of size n (line 6). Note that the exact size of each classifier set is calculated from line 4. In line 4, $condition?\ expression\ 1:expression\ 2$ is a C-style ternary operator, which returns *expression 1* if *condition* is true, *expression 2* otherwise. In the stage of referee system construction (lines 8–9), the competence label of each component classifier on the original training data $\boldsymbol{X_{tr}}$ is first obtained by predicting $\boldsymbol{X_{tr}}$ and contrasting its true label Y_{tr} (line 8), then SPSE_EM is utilized to build a referee system based on $\boldsymbol{X_{tr}}$ and the competence labels of all component classifiers (line 9). From Fig. 2, one can see that the constructed referee system consists of several generational referee collectives from a vertical view, and is comprised of n referee committees from a horizontal view. Each referee committee is specialized to provide the competence estimation for a component classifier. In the last stage of dynamic prediction (lines 10–15), each component classifier C_t and its corresponding referee committee $\mathcal{RC}_t$ are applied to predict x, respectively (lines 11–13). To obtain the final result of x, the predictions of the component classifiers are weighted by their competent levels which are estimated by the corresponding referee committees (line 14).

B. Generation of Classifier Pool

In contrast to the traditional dynamic ensemble schemes, which build n component classifiers based on the original training data, we use OREM_G to generate $\lfloor\sqrt{n}\rfloor$ artificial synthetic datasets with different class distributions, then create a SPSE_CE ensemble on each artificial dataset.

The class distribution of rth synthetic dataset is acquired from (1) ($r = 1, 2, \ldots, \lfloor\sqrt{n}\rfloor$).

$$\boldsymbol{dp_r} = normalization\left(\boldsymbol{dp_0}^{cf_r}\right) = \left(\frac{p_1^{cf_r}}{\sum_{j=1}^{|L|} p_j^{cf_r}}, \frac{p_2^{cf_r}}{\sum_{j=1}^{|L|} p_j^{cf_r}}, \ldots, \frac{p_{|L|}^{cf_r}}{\sum_{j=1}^{|L|} p_j^{cf_r}}\right), \quad (1)$$

[3]An explanation is provided in the Section S3 of supplementary material, for why $\lfloor\sqrt{n}\rfloor$ synthetic datasets are generated.

Algorithm 1: imDEF ($\boldsymbol{Tr}$, $\boldsymbol{Te}$, n, q, b, α_1, α_2, β).

Input: Training data $\boldsymbol{Tr} = [\boldsymbol{X_{tr}}, Y_{tr}] = \{(\boldsymbol{x_i}, y_i)\}_{i=1}^{|\boldsymbol{Tr}|}$, $y_i \in L$; testing data $\boldsymbol{Te}$; number of component classifiers n; OREM_G's parameter q; self-paced sampling ensemble's parameters α_1, α_2, b, and β.
Output: The prediction result of $\boldsymbol{Te}$

1: **for** $r \leftarrow 1 : \lfloor\sqrt{n}\rfloor$ **do**
2: Acquire a class distribution $\boldsymbol{dp_r}$ using (1)
3: Generate a dataset, $\boldsymbol{Tr_r} \leftarrow \text{OREM}_\text{G}(\boldsymbol{Tr}, \boldsymbol{dp_r}, q)$
4: $n_r \leftarrow (r \le n - \lfloor\sqrt{n}\rfloor \times \lfloor\frac{n}{\lfloor\sqrt{n}\rfloor}\rfloor)?\lfloor\frac{n}{\lfloor\sqrt{n}\rfloor}\rfloor + 1 : \lfloor\frac{n}{\lfloor\sqrt{n}\rfloor}\rfloor$
5: Create a classifier set, $\mathcal{P}_r \leftarrow \text{SPSE}_\text{CE}(\boldsymbol{Tr_r}, n_r, \alpha_1, b)$
6: $\mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{P}_r$
7: **end for**
8: Obtain the competence labels of all component classifiers, $\boldsymbol{CL} \leftarrow isequal(predict(\mathcal{P}, \boldsymbol{X_{tr}}), Y_{tr})$
9: Build referee system $\mathcal{RS} \leftarrow \text{SPSE}_\text{EM}(\boldsymbol{X_{tr}}, Y_{tr}, \boldsymbol{CL}, \alpha_2, b, \beta)$
10: **for all** $\boldsymbol{x} \in \boldsymbol{Te}$ **do**
11: **for all** $C_t \in \mathcal{P}$ **do**
12: Use C_t and its corrsponding referee committee $\mathcal{RC}_t$ in $\mathcal{RS}$ to predict $\boldsymbol{x}$
13: **end for**
14: Use (13) to obtain the final prediction of x
15: **end for**

where p_j is the proportion of the samples of class l_j in the whole original training data; cf_r is an adjustment factor of class distribution, its value actually determines the class distribution. We use (2) to update the value of cf_r:

$$cf_r = 1 - \frac{2 \times (r-1)}{\lfloor\sqrt{n}\rfloor - 1}. \quad (2)$$

From (2), we can see that cf_r decreases from 1 to -1 as r increases. It means that the majority (/minority) classes in the original training data would be generated fewer and fewer (/more and more) synthetic samples, and the majority (/minority) classes would be gradually transformed into the actually $weak$ ($/strong$) classes in the synthetic datasets.

There are several advantages to adopting such a way to create component classifiers. First, the component classifiers are trained using synthetic data rather than original training data. It can mitigate the overestimation problem of competent level (alleviating $Problem\ i$). If the original training data is used to construct both classifier pool and referee system, the true competence regions of the component classifiers would be difficult to be identified. Second, the average proportion of the samples in each class is close to be equal over the synthetic datasets $\{\boldsymbol{Tr_r}\}_{r=1}^{\lfloor\sqrt{n}\rfloor}$. It could be expected that the generated classifier pool is low biased towards the majority classes (rectifying $Problem\ ii$). Third, the strong classes and their degrees of dominance are different over $\{\boldsymbol{Tr_r}\}_{r=1}^{\lfloor\sqrt{n}\rfloor}$. It can encourage the diversity of SPSE_CE ensembles, and facilitate the creation of a certain number of positive classifiers for the test samples from any classes. Finally, we would see that SPSE_CE makes

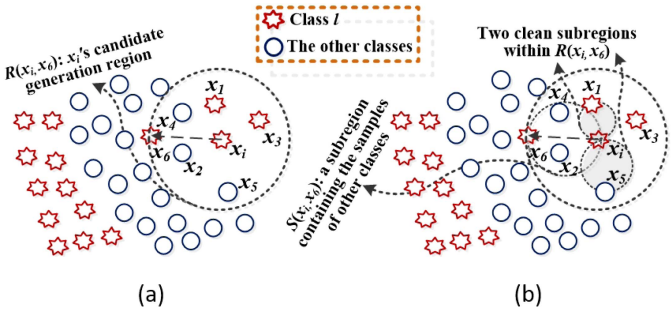

Fig. 3. (a) Figure illustrating x_i's candidate generation region. (b) Figure illustrating the subregions within x_i's candidate generation region; Two clean subregions are shaded with gray background.

the component classifiers gradually focus more on the difficult samples, which could increase the ratios of positive classifiers on the difficult-to-learn samples (handling $Problem\ iii$).

Below, we introduce the two key algorithms OREM_G and SPSE_CE.

1) Interpolated Data Generation With Reliable Generalization (OREM_G): The synthetic data for training component classifiers needs to obey the underlying distribution of the original training data, so that the constructed component classifiers could be generalized to the original classification task. To this end, a qualified data generation method should be capable of producing diversified synthetic samples to fully cover the representative samples of original data, and simultaneously reduce the generation of noisy samples as much as possible.

OREM_G is an interpolated data generation method, i.e., the synthetic samples are the interpolation points between two original samples. It calculates the number of the synthetic samples that need to be generated for each class according to $\boldsymbol{dp_r}$, then applies the oversampling method OREM [18] to generate the synthetic samples for each class. The final synthetic dataset $\boldsymbol{Tr_r}$ is the union of the synthetic samples of all classes.

OREM includes three steps when generating the synthetic samples for any sample, x_i, of a class l ($l \in L$). In the first step, OREM finds the candidate assistant seed set $\mathcal{C}(\boldsymbol{x_i})$ for x_i. $\mathcal{C}(\boldsymbol{x_i})$ is comprised of x_i's p-nearest neighbors, where p is the minimum value when none of the subsequent q neighbors (i.e., $\boldsymbol{x_{i(p+1)}}, \boldsymbol{x_{i(p+2)}}, \ldots, \boldsymbol{x_{i(p+q)}}$) are from the class l. This step explores the Candidate Generation Region (CGR) near x_i. Such a CGR is either a region of class overlapping or a pure region of class l.

Fig. 3(a) illustrates an example of identifying a CGR of class l around x_i. The circular area $R(x_i, x_6)$ centered on x_i, with the distance between x_i and x_6 as the radius, can be considered as a possible region of class l, because a certain number of the samples of class l are distributed in this region with a form of class overlapping. In addition, an abundance of neighbor samples subsequent to x_6 are all from the other classes, which suggests that the samples of class l would have a low probability of appearing in the adjacent area outside $R(x_1, x_6)$. Therefore, $R(x_1, x_6)$ could be regarded as a maximal consecutive CGR around x_i.

Algorithm 2: $\text{SPSE}_{CE}(\boldsymbol{Tr_t}, n_t, \alpha_1, b)$.

Input: Synthetic dataset $\boldsymbol{Tr_r}$; number of classifiers n_r; Self-paced adjusting parameter α_1; number of bins b.
Output: The classifier set $\mathcal{P}_r$

$\boldsymbol{S_0} \leftarrow RandomSampleReplacement(\boldsymbol{Tr_r}, |\boldsymbol{Tr_r}|)$
$C_0 \leftarrow TrainAClassifier(\boldsymbol{S_0})$
$\mathcal{P}_r \leftarrow \mathcal{P}_r \cup \{C_0\}$
for $t \leftarrow 1 : n_r$ **do**
  Use (3) to compute $\mathcal{H}(x_i, y_i, \mathcal{P}_r)$ for each sample
  Cut $\boldsymbol{Tr_r}$ into b bins through (4): $B_1, B_2, \ldots, B_b$
  $d_j = \sum_{i \in B_j} (\mathcal{H}(x_i, y_i, \mathcal{P}_r)/|B_j|), \forall j = 1, 2, \ldots, b$
  Update self-paced factor: $s_t = 1 - (t-1)/(\alpha_1(n_r - 1))$
  Acquire sampling weight of jth bin: $sw_j = (1/d_j)^{s_t}$
  Obtain a sample set $\boldsymbol{S_t}$ by randomly selecting $|\boldsymbol{Tr_r}| \cdot (sw_j / \sum_l sw_l)$ samples from $B_j, \forall j = 1, \ldots, b$
  $C_t \leftarrow TrainAClassifier(\boldsymbol{S_t})$
  $\mathcal{P}_r \leftarrow \mathcal{P}_r \cup \{C_t\}$
end for
$\mathcal{P}_r \leftarrow \mathcal{P}_r \setminus \{C_0\}$

The second step of OREM is to further identify x_i's those reliable assistant seeds, $\mathcal{A}(\boldsymbol{x_i})$, from $\mathcal{C}(\boldsymbol{x_i})$. The identification rule is that if the hypersphere area between a sample in $\mathcal{C}(\boldsymbol{x_i})$ and x_i does not contain any sample from the other classes, this sample could be considered a reliable assistant seed. This step aims to discover the clean subregions within x_i's CGR, and use them to place the synthetic samples of class l. The reason behind is that the clean subregions would have a low probability of occurring the test samples from non-l classes. Fig. 3(b) shows an example of finding the clean subregions near x_i. Obviously, two circular shaded regions $S(x_i, x_1)$ and $S(x_i, x_5)$ are the clean subregions. It is safe to fill these two subregions with the synthetic samples of class l. x_1 and x_5 belong to the assistant seed set of x_i.

The final step of OREM is to generate the synthetic samples for x_i. Each synthetic sample is a random interpolation point between x_i and its an arbitrary assistant seed.

The assistant seed sets in OREM could contain the samples of different classes, and their sizes are variable. In traditional interpolation generation methods, the assistant seeds of a considered sample are often the k-nearest neighbors from the same class. Hence, OREM can generate more diverse synthetic samples compared to traditional interpolation approaches, and the limitation, that the synthetic samples of a class are only created within the convex hull formed by the original samples of this class, could be broken. In addition, OREM only generates synthetic samples in the clean subregions within CGR. It could effectively prevent the synthetic samples from falling into those controversial subregions, and thus reduce the generation of noisy samples.

The pseudocode description of OREM_G is given in Section S4 of the supplementary material. More details on OREM could be found in [18].

2) Classification-Error-Aware Self-Pace Sampling Ensemble ($SPSE_{CE}$): As analyzed in Section I-C, the ensemble methods building classifier pool should reinforce the learning of difficulty samples, and increase the percentages of the positive classifiers on these samples. Inspired by self-paced ensemble [3], we propose a classification-error-aware self-paced sampling ensemble method SPSE_{CE} to train the component classifiers.

Algorithm 2 presents the details of SPSE_{CE}. In lines 1–3, SPSE_{CE} collects a bootstrap replica $\boldsymbol{S_0}$ from the synthetic dataset $\boldsymbol{Tr_r}$, and trains a component classifier C_0 with $\boldsymbol{S_0}$, then uses C_0 to initialize the classifier set $\mathcal{P}_r$. In lines 4–13, SPSE_{CE} generates n_r component classifiers iteratively. First, the classification difficulty of each sample is measured by its classification error w.r.t the current ensemble $\mathcal{P}_r$ (line 5). It is calculated as (3):

$$\mathcal{H}(x_i, y_i, \mathcal{P}_r) = (1/|\mathcal{P}_r|) \sum_{t=0}^{|\mathcal{P}_r|-1} (1 - C_t(x_i, y_i)), \tag{3}$$

where $C_t(x_i, y_i)$ is the confidence degree that the classifier C_t predicts x_i into the true label y_i. Based on the difficulty levels of samples, SPSE_{CE} can divide $\boldsymbol{Tr_r}$ into b bins, i.e., $\{B_j\}_{j=1}^{b}$. B_j is defined as

$$B_j = \left\{ (x_i, y_i) | (j-1)/b \le \hat{\mathcal{H}}(x_i, y_i, \mathcal{P}_r) < j/b \right\}, \tag{4}$$

where $\hat{\mathcal{H}}$ is the normalization form of $\mathcal{H}$, i.e.,

$$\hat{\mathcal{H}}(x_i, y_i, \mathcal{P}_r) = \frac{\mathcal{H}(x_i, y_i, \mathcal{P}_r) - \min\limits_{x_l \in \boldsymbol{Tr_r}} \mathcal{H}(x_l, y_l, \mathcal{P}_r)}{\max\limits_{x_l \in \boldsymbol{Tr_r}} H(x_l, y_l, \mathcal{P}_r) - \min\limits_{x_l \in \boldsymbol{Tr_r}} \mathcal{H}(x_l, y_l, \mathcal{P}_r)}. \tag{5}$$

Hence, $\hat{\mathcal{H}}(x_i, y_i, \mathcal{P}_r) \in [0, 1]$. Then, the average difficulty level d_j is calculated for the samples within B_j (line 7). Obviously, d_1 and d_b have the lowest and highest average difficulty levels, respectively. Finally, we use a self-paced procedure to collect a sample set $\boldsymbol{S_t}$ from $\{B_j\}_{j=1}^{b}$, where $\boldsymbol{S_t}$ is used to train tth component classifier (lines 8–11). The key is to control the sampling weight of each bin through a self-paced factor s_t. s_t is updated in the way of line 8. We can see that as t increases, s_t decreases gradually from 1 to $1 - 1/\alpha_1$, where α_1 is a self-paced adjusting parameter (note: we would detail the role of α_1 in Section IV-A5). Initially, the sampling weight of each bin, sw_j, is inversely proportional to its average difficulty level (line 9, $s_t = 1$, then $sw_j = 1/d_j$). The low-difficulty bins would have higher sampling weights compared to the ones with high difficulty levels. As the iteration progresses (i.e., decreasing s_t), the sampling weights of high-difficulty bins are increased. In particular, s_t could become negative when $\alpha_1 < 1$. It implies that the bins with high difficulty levels would obtain higher sampling weights compared to those low-difficulty bins. Since a higher sampling weight means that more samples would be selected from the corresponding bin, the number of difficult-to-learn samples would be progressively increased in $\boldsymbol{S_t}$ (line 10). In this way, SPSE_{CE} could gradually strengthen the learning of the difficult samples.

Although AdaBoost family can also focus more on the difficult samples, the noisy samples is inevitable in synthetic dataset. The training mechanism of AdaBoost could make the weights being excessively concentrated on difficult-to-learn noise, eventually degrading the qualities of component classifiers. Several boosting methods have been developed to enhance the robustness of AdaBoost. They usually design and optimize new robust loss functions [28]. However, the boosting methods based on convex loss functions are vulnerable to the effects of random noise [29], and the use of nonconvex loss functions could lead to unreliable and unstable solutions. In addition, the base classifier of AdaBoost family is not suitable for the strong classifiers, while self-paced ensemble is able to work with any kind of classifiers [3], [21].

C. Construction of Referee System

1) Preliminary: As discussed in Section I-C, neighborhood-based DSTs inherently suffer from the problem of competence conflict on the difficult samples. To cope with this problem, we build a referee system to accomplish the competence estimation (avoiding $Problem\ iv$). In fact, a few works have used referee-based idea to estimate the competent levels of component classifiers [30]. In these works, each referee is associated with a component classifier. A referee learns the area of expertise of its associated component classifier, and estimates how much accurate (i.e., competent level) the component classifier is for a test sample. In our imDEF, however, we build a multi-generational referee committee for each component classifier. The multi-generational referee committee consists of multiple referees which are sequentially trained. The motivation behind is to gradually enhance the accuracy of competence estimation on the samples with small dynamic ensemble margins (i.e., those hard-to-classify samples; thereby handling $Problem\ v$).

The dynamic ensemble margin is defined as (6):

$$\mathcal{M}(x_i, \mathcal{RS}, \boldsymbol{CL}) = \sum_{t=1}^{n} \mathcal{RC}_t(x_i) \mathbb{I}(\boldsymbol{CL}[i,t] = 1) - \sum_{t=1}^{n} \mathcal{RC}_t(x_i) \mathbb{I}(\boldsymbol{CL}[i,t] = 0), \tag{6}$$

where $\mathcal{RC}_t$ is the referee committee associated with the component classifier C_t; $\mathcal{RS}$ is the set of all referee committees; $\mathcal{RC}_t(x_i)$ is $\mathcal{RC}_t$'s competence estimation for C_t on x_i; $\mathbb{I}(\cdot)$ is an indicator function, which returns 1 if its argument is true, and 0 otherwise; $\boldsymbol{CL}$ is the competent label set of all component classifiers; $\boldsymbol{CL}[i,t]$ represents the competent label of C_t on x_i (i.e., $\boldsymbol{CL}[i,t]$ is 0 if $C_t(x_i) \neq y_i$, and 1 otherwise).

The dynamic ensemble margin shown in (6) is the difference of the weighted votes between the component classifiers that correctly predict x_i and wrongly classify x_i. A large positive (/negative) margin value could be interpreted as a highly confident correct (/incorrect) classification on x_i w.r.t the classifier pool and the current referee system.

In the existing literature, various types of margins have been used in the construction of generalization bounds and the design of machine learning algorithms. It has been shown that the improvement of margin distribution could enhance the robustness and performance of classification system [31].

In this work, the samples with low dynamic ensemble margins can be linked to those most important samples w.r.t the construction of referee system. For example, Table II shows the prediction results of five component classifiers on five samples. We can see that dynamic ensemble way would be likely to correctly classify the samples in the set A with high dynamic ensemble margins, even if a referee system provides random values for the competent levels of component classifiers. This is because most of the classifiers can deliver the correct results for the samples in A. In contrast, the referee system needs to provide accurate competence estimations on the samples in B, i.e., the positive (/negative) component classifiers should obtain high (/low) competent levels. This is the only way to correct the errors delivered by most of the component classifiers, and produce positive dynamic ensemble margins (i.e., correct collective results). Therefore, the accuracy requirement regarding the competence estimation is different for classifying different samples. The referee system should pay more attention to the learning of the samples with low dynamic ensemble margins, so as to provide the highly accurate competence estimations on the hard-to-classify samples. In this way, the advantage of dynamic ensemble could be fully exploited.

2) Ensemble-Margin-Aware Self-Paced Sampling Ensemble (SPSE$_{\mathrm{EM}}$): To construct an effective referee system, we propose an ensemble-margin-aware self-paced sampling ensemble method, SPSE$_{\mathrm{EM}}$. The main process of SPSE$_{\mathrm{EM}}$ is as follows.

a) SPSE$_{\mathrm{EM}}$ computes the Cumulative Negative dynamic Ensemble Margin (CNEM) for each training sample w.r.t the current referee system (note: CNEM is positively correlated with the importance of samples regarding the construction of current referee system).
b) The training samples are divided into several bins according to their values of CNEM.
c) A self-paced procedure combined with a class weight distribution is used to select the samples from each bin, which gradually increases the number of the samples chosen from the bins having high CNEM values.
d) The selected sample set is applied to construct a generational referee collective that consists of the referee classifier associated with each component classifier.
e) Return to step a), until a desired number of referee collectives have been constructed.

Algorithm 3 presents the details of SPSE$_{\mathrm{EM}}$. SPSE$_{\mathrm{EM}}$ first calculates the initial Negative dynamic Ensemble Margin (NEM) for each training sample based on (7) (line 1):

$$\mathcal{M}_0(x_i, \boldsymbol{CL}) = -\sum_{t=1}^{n}(\mathbb{I}(\boldsymbol{CL}[i,t]=1) - \mathbb{I}(\boldsymbol{CL}[i,t]=0)). \tag{7}$$

As there are no referee classifiers at the beginning, we simply assume that the competent levels of all component classifiers are 1. This step is to obtain the original classification hardness of the sample w.r.t the classifier pool.

SPSE $_{\mathrm{EM}}$ constructs multi-generational referee collectives iteratively. After constructed p th generational referee collective ($p \leq \beta$), a sample x_i's CNEM is the sum of x_i's NEM values over these p generational referee collectives, denoted by $\mathcal{CM}(x_i)$:

$$\mathcal{CM}(x_i) = \mathcal{M}_0(x_i, \boldsymbol{CL}) + \sum_{g=1}^{p}\mathcal{M}_g(x_i, \mathcal{GC}^g, \boldsymbol{CL}), \tag{8}$$

where $\mathcal{GC}^g$ is gth referee collective; $\mathcal{M}_g(x_i, \mathcal{GC}^g, \boldsymbol{CL})$ is x_i's NEM w.r.t $\mathcal{GC}^g$, and is defined as follows.

$$\mathcal{M}_g(x_i, \mathcal{GC}^g, \boldsymbol{CL}) = \sum_{R_t^g \in \mathcal{GC}^g} R_t^g(x_i)\mathbb{I}(\boldsymbol{CL}[\boldsymbol{i},\boldsymbol{t}]=0) - \sum_{R_t^g \in \mathcal{GC}^g} R_t^g(x_i)\mathbb{I}(\boldsymbol{CL}[\boldsymbol{i},\boldsymbol{t}]=1). \tag{9}$$

In (9), R_t^g is the referee classifier in $\mathcal{GC}^g$ built for C_t. Initially, $\mathcal{CM}(x_i)$ is $\mathcal{M}_0(x_i, \boldsymbol{CL})$ (line 2).

Given that the original class distribution is imbalanced, we assign each training sample, x_i, a class-dependent weight $w(x_i)$

Algorithm 3: SPSE$_{\mathrm{EM}}(\boldsymbol{X_{tr}}, Y_{tr}, \boldsymbol{CL}, \alpha_2, b, \beta)$.

Input: Feature part of training set $\boldsymbol{X_{tr}}$; label part of training set Y_{tr}; competent label set of the component classifiers $\boldsymbol{CL}$; self-paced adjusting parameter α_2; number of bins b; number of generations β;
Output: The referee committee set $\mathcal{RS} = \{\mathcal{RC}_t\}_{t=1}^n$
1: For each $x_i \in \boldsymbol{X_{tr}}$, compute $\mathcal{M}_0(x_i, \boldsymbol{CL})$ via (7)
2: For each $x_i \in \boldsymbol{X_{tr}}$, initialize $\mathcal{CM}(x_i) \leftarrow \mathcal{M}_0(x_i, \boldsymbol{CL})$
3: For each $x_i \in \boldsymbol{X_{tr}}$, assign its class weight $w(x_i)$ via (10)
4: For each component classifier, initialize its associated referee committee: $\mathcal{RC}_t \leftarrow \varnothing, t = 1, \ldots, n$.
5: Initialize all referee collectives: $\mathcal{GC}_g \leftarrow \varnothing, g = 1, \ldots, \beta$
6: **for** $g \leftarrow 1 : \beta$ **do**
7: For each $x_i \in \boldsymbol{X_{tr}}$, normalize $\mathcal{CM}(x_i)$ into $\mathcal{C\hat{M}}(x_i)$ using (11).
8: Use (12) to cut $\boldsymbol{X_{tr}}$ into b bins w.r.t $\mathcal{C\hat{M}}$:$\{B_j\}_{j=1}^b$
9: $d_j = \sum_{x_i \in B_j}(\mathcal{C\hat{M}}(x_i)/|B_j|), \forall j = 1, 2, \ldots, b$
10: Update self-paced factor: $s_g = 1 - (g-1)/(\alpha_2(\beta-1))$
11: Acquire sampling weight of jth bin: $sw_j = (1/d_j)^{s_g}$
12: $\delta_j \leftarrow (sw_j / \sum_l sw_l) \times |\boldsymbol{X_{tr}}|, \forall j = 1, \ldots, b$
13: $\hat{w_j}(x_i) \leftarrow w(x_i) / \sum_{x_l \in B_j} w(x_l), \forall j = 1, \ldots, b$
14: **for** $t \leftarrow 1 : n$ **do**
15: Obtain a sample set $\boldsymbol{S_t^g}$ by weighting sampling δ_j samples from B_j according to $\boldsymbol{\hat{w}_j}, \forall j = 1, \ldots, b$
16: Extract $\boldsymbol{S_t^g}$'s competence label L_t^g from $\boldsymbol{CL}[:, \boldsymbol{t}]$
17: $R_t^g \leftarrow TrainAClassifier((\boldsymbol{S_t^g}, L_t^g))$
18: $\mathcal{GC}^g \leftarrow \mathcal{GC}^g \cup \{R_t^g\}$
19: $\mathcal{RC}_t \leftarrow \mathcal{RC}_t \cup \{R_t^g\}$
20: **end for**
21: For each $x_i \in \boldsymbol{X_{tr}}$, compute its $\mathcal{M}_g(x_i, \mathcal{GC}^g, \boldsymbol{CL})$ according to (9)
22: For each $x_i \in \boldsymbol{X_{tr}}$, update its $\mathcal{CM}(x_i)$: $\mathcal{CM}(x_i) \leftarrow \mathcal{CM}(x_i) + \mathcal{M}_g(x_i, \mathcal{GC}^g, \boldsymbol{CL})$
23: **end for**

(line 3):

$$w(x_i) = \frac{\max_k \sqrt{m_k}}{\sum_{l_k \in L} \mathbb{I}(y_i = l_k)\sqrt{m_k}}, \tag{10}$$

where m_k is the number of the samples of class l_k in $\boldsymbol{X_{tr}}$. The class weights are used for weighted sampling within each bin. The higher the class weight, the higher the likelihood that the sample is selected. In the imbalanced learning field, rebalancing strategies such as resampling and reweighting, usually assign the sample weights inversely proportional to the class frequency. However, it could result in poor performance as the importance of minority samples might be overemphasized [32]. Hence, we use a *moderate* version in SPSE_{EM} that the class weight of the sample is inversely proportional to the square root of the class frequency [32].

Our referee system is comprised of n referee committees corresponding to the component classifiers from a horizontal view, and consists of several generational referee collectives from a vertical view. At the beginning, each referee committee and all referee collectives are initialized to be empty (lines 4–5). The key part of SPSE_{EM} is the iterative training of multi-generational referee collectives (lines 6–23). For each training sample x_i, $\mathcal{CM}(x_i)$ is normalized into $\mathcal{C}\hat{\mathcal{M}}(x_i)$ (line 7):

$$\mathcal{C}\hat{\mathcal{M}}(x_i) = \frac{\mathcal{CM}(x_i) - \min_{x_l \in \boldsymbol{Tr}} \mathcal{CM}(x_l)}{\max_{x_l \in \boldsymbol{Tr}} \mathcal{CM}(x_l) - \min_{x_l \in \boldsymbol{Tr}} \mathcal{CM}(x_l)}. \tag{11}$$

$\mathcal{C}\hat{\mathcal{M}}(x_i)$ ranges from 0 to 1. The whole training set could be divided into b bins, i.e., $\{B_j\}_{j=1}^{b}$. B_j is defined as (line 8):

$$B_j = \{x_i | (j-1)/b \leq \mathcal{C}\hat{\mathcal{M}}(x_i) < j/b\}. \tag{12}$$

Once n bins have been obtained, we compute the average $\mathcal{C}\hat{\mathcal{M}}$ on the samples within each bin, i.e., $d_j, j = 1, \ldots, b$ (line 9). d_j is combined with a self-paced factor s_g to calculate the number of the samples selected from each bin. The self-paced factor s_g is updated in the way of line 10. With the increase of g, s_g diminishes from 1 to $1 - 1/\alpha_2$, where α_2 is an adjusting parameter similar to α_1. According to the calculation of sampling weight (line 11), the reduction of s_g would lead to higher sampling weights for the bins with high CNEM values (i.e., high d_j). Hence, as the iteration progresses, an increasing number of samples would be selected from those bins with high CNEM values (line 12). Different from SPSE_{CE}, SPSE_{EM} performs a weighted sampling to select the samples from each bin according to the class weight distribution of the samples within each bin (lines 13 and 15). The reason behind is to increase the frequency of selecting minority samples, and combat the class imbalance problem in training data. Finally, g th generational referee collective $\mathcal{GC}^g$ is constructed, which tth referee R_t^g is trained on the sampled data $\boldsymbol{S_t^g}$ and its corresponding competence label L_t^g (lines 15–17). R_t^g is added into both $\mathcal{GC}^g$ and the referee committee $\mathcal{RC}_t$ associated with the component classifier C_t (lines 18–19). After $\mathcal{GC}^g$ is constructed, we could compute the NEM of each training sample w.r.t $\mathcal{GC}^g$ according to (9) (line 21). Then, the CNEM of each training sample is updated to be used for building next generational referee collective (line 22).

D. Dynamic Prediction

The dynamic prediction in our imDEF could be expressed as:

$$\hat{y} = \arg\max_{l_k \in L} \sum_{t=1}^{n} \mathcal{RC}_t(x) C_t(x, l_k), \tag{13}$$

where x is the considered test sample; $C_t(x, l_k)$ is the probability that x is predicted into class l_k by C_t; $\mathcal{RC}_t(x)$ is the competent level of C_t at x estimated by the referee committee $\mathcal{RC}_t$. It is actually the average of the competence estimations provided by all the referees in $\mathcal{RC}_t$, i.e., $\mathcal{RC}_t(x) = (1/\beta)\sum_{g=1}^{\beta} R_t^g(x)$.

IV. Experimental Study

To evaluate the effectiveness of imDEF, imDEF is compared with state-of-the-art imbalanced static ensemble methods, and existing imbalanced dynamic ensemble schemes over two-class and multiclass imbalanced datasets, respectively. In addition, we compare imDEF with its variants to verify the usefulness of each key ingredient of imDEF.

It should be highlighted that due to space constraints, only the essential ingredients of experimental study are presented in the paper. Lots of important content is placed in a 70-page supplementary material. The content includes, but not limited to, the computational complexity analysis and running time comparison for the compared algorithms (Section S5), the performance comparisons of imDEF and non-ensemble imbalance learning methods (Section S6), an experimental investigation on large-scale imbalanced datasets (Section S7), and the parameter optimization analysis of imDEF (Section S8). We highly recommend readers to refer to these elements that enhance the completeness of our work.

A. Experiments on Two-Class Imbalanced Datasets

1) Experimental Setting: Dataset. 30 two-class datasets are selected in this experiment. A few of them (LEV_{T}, ERA_1, ERA_2, and ESL_{T}) are the widely-used ordinal regression benchmark data [33], and the others are available from UCI [34]. These sets are chosen in such a way that they exhibit diversity in terms of sample size, feature dimension, and imbalance ratio. Table III shows the detailed characteristics of the selected datasets.

Assessment Measures. F1, G-mean, and AUC[4] are often used together to evaluate the performance of imbalanced learning methods [10], [18], [24]. To erase the effects of random factors, all the methods are run 10 times on each dataset with stratified 5-fold cross validation. The final performance on each dataset is the average of 50 testing results.

Base Classifier. Classification regression tree is adopted as base classifier for all the compared ensemble methods, because it is most frequently used in existing ensemble solutions [2]. The number of base classifiers is set to 100.

Statistical Analysis. It is important to verify whether there is a significant performance difference between imDEF and each of the other compared algorithms. We apply the commonly-used

[4]A simple description regarding the calculation of AUC is provided in Section S9 of the supplementary material.

TABLE IV
THE STATIC ENSEMBLE SOLUTIONS USED IN THE EXPERIMENTAL STUDY

Method	Brief Description
SMOTEB	In each iteration of AdaBoost.M2, applying SMOTE to balance training data
EasyE	Balanced Bagging with undersampling the majority class, and learning each bag with AdaBoost.M2
BalanceC	Similar to BalanceC, but removing correctly classified majority samples in each bagging iteration
RBB	In each iteration of AdaBoost.M2, using random balance resampling to preprocess training data
SPE	Splitting the majority samples into different bins according to hardness levels, then building ensemble via a self-paced undersampling procedure over all bins
DAPS	Similar to SPE, but focusing on the noise disturbance in the overlapping regions by dividing the entire data into the vetoed samples and non-vetoed samples
OREMB	In each iteration of AdaBoost.M2, applying OREM to balance training data
AdaCC1	Adjusting the misclassified cost on the minority (/majority) class automatically according to the false negative (/positive) rate of the current ensemble
ASE	Using an anomaly detection algorithm to obtain the abnormal scores of samples, then utilizing the anomaly scores to guide the undersampling of majority samples
MBSB	In each iteration of AdaBoost.M2, using model-based oversampling to preprocess training data
AdaAD	Integrating the adaptive sample weight with AdaBoost ensemble to alleviate multi-class imbalance issue

TABLE V
PARAMETER SETTINGS OF THE IMBALANCED STATIC ENSEMBLE METHODS

Algorithm	Parameters of Algorithms	Year
SMOTEB	$k = 5$	2003
EasyE	$T = 10,\ s_i = 10$	2009
BalanceC	$T = 10,\ s_i = 10$	2009
RBB	$k = 5$	2015
SPE	$bin\_n = 20$	2020
DAPS	$B = 10, k = 5$	2022
OREMB	$q = 5$	2023
AdaCC1	/	2023
ASE	$k = 5$	2024
MBSB	$T = 5$	2020
AdaAD	$k = 5$	2024
imDEF	$q = 5, b = 20,$ $\alpha_1 = 0.7, \alpha_2 = 1, \beta = 5$	/

Wilcoxon signed rank test to accomplish the significance tests for each pair of algorithms [35].

2) Comparison With Typical Static Ensemble Solutions: We compare the proposed imDEF with nine state-of-the-art unbalanced static ensemble methods. They are the resampling techniques-combined ensemble solutions SMOTEBoost (SMOTEB) [6], OREMBoost (OREMB) [18], EasyEnsemble (EasyE) [7], BalanceCascade (BalanceC) [7], RBBoost (RBB) [9], and ASE [36], the self-paced ensemble solutions SPE [3] and DAPS [21], and the cost-sensitive ensemble approach AdaCC1 [10]. A brief description of these methods is summarized in Table IV. All the parameters in them use the default settings (see Table V).

Due to space constraints, we put the F1, G-mean, and AUC results of imDEF and the other compared static ensemble solutions in Table S25 of the supplementary material. From this table, one can see that our imDEF achieves the best performances on 8, 12, and 15 out of 30 datasets in terms of F1, G-mean, and AUC, respectively. We further perform the Wilcoxon signed-rank significance tests on the performance values of Table S25. Table VI shows the results of significance tests. We can find that imDEF can achieve statistically superior performance on all the comparison terms, except for the comparisons with EasyE and SPE in G-mean. It demonstrates that imDEF is highly effective compared to the representative imbalanced static ensemble approaches.

3) Comparison With Existing Imbalanced Dynamic Ensemble Schemes: Four existing imbalanced dynamic ensemble schemes are selected here. They are B+RM100+KNU [24], OLP+ENN+KNNE [37], [38], BR+RM+KNU [26], and EE+MCB [23]. Given that imDEF needs to produce 100 component classifiers ($n = 100$) and 500 referees ($\beta = 5$), we derive two versions for each of these compared schemes except for OLP+ENN+KNNE, in which the classifier pools of size 100 and 600 are generated for them, respectively. Note that OLP+ENN+KNNE creates a local pool of linear hyperplane classifiers when predicting a test sample. Each hyperplane is trained on the test sample's neighboring samples within a certain neighborhood size. It is inappropriate to generate a local pool of size 100 or 600 for OLP+ENN+KNNE. Table VII summarizes the configurations of these schemes, including classifier pool generation, DSEL, and DST. The default parameter settings of them are listed in Table VIII.

In Section S10 of the supplementary material, we present an explanation for why these imbalanced dynamic ensemble schemes are chosen to compare imDEF. The complete experimental results are provided in Table S26 of the supplementary material, where imDEF obtains the best F1, G-mean, and AUC values on 15, 14, and 15 out of 30 datasets, respectively. Table IX summarizes the results of the Wilcoxon significance tests between imDEF and each of the compared schemes. We can observe that the significant differences exist on almost all of the comparison items. It shows that imDEF can outperform these schemes significantly.

4) Comparison With imDEF's Variants: imDEF contains three key ingredients: 1) OREM_{G} generates artificial synthetic datasets; 2) SPSE_{CE} creates the classifier pool based on the generated datasets; and 3) SPSE_{EM} constructs a referee system. In this subsection, we validate the usefulness of these three ingredients for imDEF. To this end, several imDEF variants are derived by modifying these ingredients. These variants are as follows.

- imDEF w SM: the synthetic datasets are generated by SMOTE_{G} (note: SMOTE_{G} utilizes SMOTE to create the synthetic samples of each class). The class distributions of these synthetic datasets are $\{\boldsymbol{dp_r}\}_{r=1}^{\lfloor n \rfloor}$.
- imDEF w/o Gen: SPSE_{CE} is trained based on the sample sets randomly selected from the original training data. Their class distributions are $\{\boldsymbol{dp_r}\}_{r=1}^{\lfloor n \rfloor}$.

TABLE VI
p-VALUES OF THE WILCOXON SIGNED-RANK TESTS FOR THE COMPARISONS BETWEEN IMDEF AND EACH OF THE STATIC ENSEMBLE SOLUTIONS

Measures	imDEF vs								
	SMOTEB	EasyE	BalanceC	RBB	SPE	DAPS	OREMB	AdaCC1	ASE
F1	**0.0001**$_{++}$	**0.0001**$_{++}$	**0.0003**$_{++}$	**0.0327**$_{++}$	**3.8e-06**$_{++}$	**1.6e-07**$_{++}$	**0.0106**$_{++}$	**0.0005**$_{++}$	**1.9e-09**$_{++}$
G-mean	**1.9e-08**$_{++}$	0.7418$_{-}$	**0.0350**$_{++}$	**1.9e-09**$_{++}$	0.1840$_{+}$	**1.9e-09**$_{++}$	**0.0001**$_{++}$	**2.5e-07**$_{++}$	**6.0e-06**$_{++}$
AUC	**0.0065**$_{++}$	**0.0013**$_{++}$	**0.0011**$_{++}$	**4.0e-05**$_{++}$	**0.0017**$_{++}$	**0.0005**$_{++}$	**0.0079**$_{++}$	**0.0002**$_{++}$	**3.7e-09**$_{++}$

"++" signifies that imDEF is statistically better(/worse) than the compared algorithm at a significant level of 0.05.
"+" denotes that imDEF is only quantitatively better, whereas "-" implies the contrary.

TABLE VII
THE IMBALANCED DYNAMIC ENSEMBLE SCHEMES USED IN THE EXPERIMENTAL STUDY

Method	Classifier Pool Generation	Dynamic Selection Data	Dynamic Selection Technique
B_{100}+RM100+KNU B_{600}+RM100+KNU	Bagging combined with RAMO with 100% oversampling ratio	Rebalanced original training set by applying RAMO with 100% oversampling ratio	KNORA-U
OLP+KNNE+ENN	Using KNNE to select the training samples, utilizing SGH to generate online local pool	Edited original training set by using ENN	LCA
BR_{100}+RM+KNU BR_{600}+RM+KNU	Balanced random forest	Rebalanced original training set by applying RAMO	KNORA-U
EE_{100}+MCB EE_{600}+MCB	EasyEnsemble	Original training set	MCB
OLP+KNN	Using KNN to select the training samples, utilizing SGH to generate online local pool	Original training set	LCA
BR_{100}+KNU+RMkNN BR_{600}+KNU+RMkNN	Balanced random forest	Original training set	KNORA-U combined with RMkNN

TABLE VIII
PARAMETER SETTINGS OF THE IMBALANCED DYNAMIC ENSEMBLE SCHEMES

Method	Parameters of Methods	Year
B_{100}+RM100+KNU B_{600}+RM100+KNU	$k1 = 5$ $k2 = 10$ $\alpha = 0.3$ (in RAMO); $k = 7$ (in KNORAU)	2018
OLP+KNNE+ENN	$k_s = 7$ $M = 5$ (in OLP and KNNE); $k = 3$ (in ENN)	2019
BR_{100}+RM+KNU BR_{600}+RM+KNU	$k1 = 5$ $k2 = 10$ $\alpha = 0.3$ (in RAMO); $k = 7$ (in KNORAU)	2019
EE_{100}+MCB EE_{600}+MCB	$T = 10$ $s_i = 10$ (in EasyEnsemble); $k = 7$ $similarity\_threshold = 0.7$ $selection\_threshold = 0.1$ (in MCB)	2023
OLP+KNN	$k_s = 7$ $M = 5$	2019
BR_{100}+KNU+RMkNN BR_{600}+KNU+RMkNN	$k = 7$ (in KNORAU); $k = 7$ (in RMkNN)	2020

TABLE IX
p-VALUES OF THE WILCOXON SIGNED-RANK TESTS FOR THE COMPARISONS BETWEEN IMDEF AND EACH OF THE COMPARED IMBALANCED DYNAMIC ENSEMBLE SCHEMES OVER 30 TWO-CLASS IMBALANCE DATASETS

imDEF vs	Measures		
	F1	G-mean	AUC
B_{100}+RM100+KNU	**0.0006**$_{++}$	**1.9e-08**$_{++}$	**0.0006**$_{++}$
OLP+ENN+KNNE	**0.0002**$_{++}$	**5.9e-07**$_{++}$	**1.9e-09**$_{++}$
BR_{100}+RM+KNU	**0.0002**$_{++}$	**0.0432**$_{++}$	**0.0060**$_{++}$
EE_{100}+MCB	**0.0001**$_{++}$	0.9193$_{+}$	**4.2e-06**$_{++}$
B_{600}+RM100+KNU	**0.0032**$_{++}$	**2.6e-07**$_{++}$	**0.0322**$_{++}$
BR_{600}+RM+KNU	**2.1e-05**$_{++}$	0.1142$_{+}$	**0.0239**$_{++}$
EE_{600}+MCB	**0.0008**$_{++}$	0.4249$_{-}$	**0.0449**$_{++}$

- imDEF w Bag: it replaces $SPSE_{CE}$ with Bagging to create classifier pool.
- imDEF w AdaB: it replaces $SPSE_{CE}$ with AdaBoost.M2 to create classifier pool.
- imDEF w/o DS: $SPSE_{EM}$ is abandoned. All component classifiers are used to predict test samples.
- imDEF w Sig: $SPSE_{EM}$ is abandoned. The competence region of each component classifier is learned by a single classification regression tree.

The F1, G-mean, and AUC values of these variants are summarized in Table S24 of the supplementary material. We conduct the Wilcoxon signed-rank significance tests based on Table S24. The corresponding results are shown in Table X. From this table, we can observe that imDEF is often statistically superior to all the variants. A detailed analysis is provided in Section S11 of the supplementary material w.r.t the performance differences between imDEF and each of the variants, which is highly recommended for understanding imDEF in depth.

In addition to imDEF w Sig, we can also replace $SPSE_{EM}$ with existing DSTs to accomplish dynamic prediction. imDEF w KNE, imDEF w KNU, imDEF w desP, imDEF w KnoP, imDEF w desRRC, imDEF w desKL, and imDEF w METADES are implemented to fully verify the usefulness of $SPSE_{EM}$, which KNE [5], KNU [5], desP [15], KnoP [16], desRRC [14], desKL [15], and META-DES.H [4] are used to replace $SPSE_{EM}$, respectively. Note that the training data would be divided into two parts for imDEF w METADES, i.e., 50% for meta-training

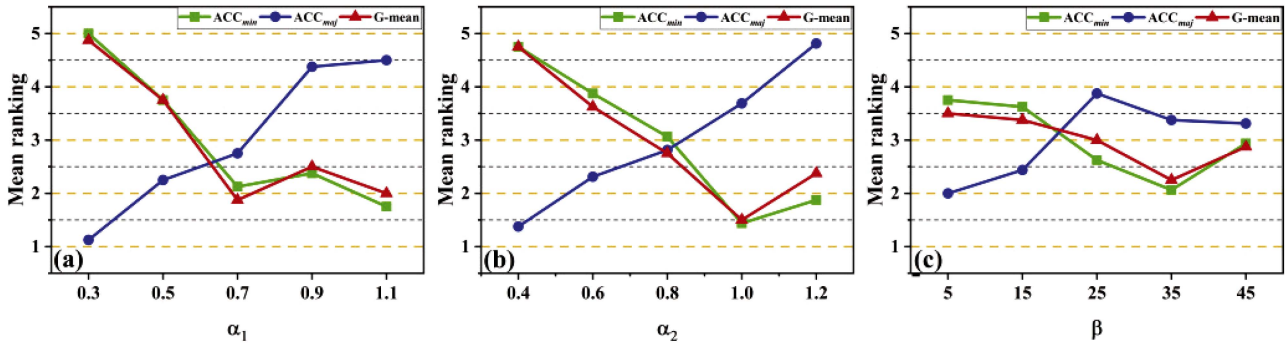

Fig. 4. Figure illustrating how the mean ACC_{min}, ACC_{maj}, and G-mean rankings of imDEF varies with the parameters α_1, α_2, and β.

TABLE X
p-VALUES OF THE WILCOXON SIGNED-RANK TESTS FOR THE COMPARISONS BETWEEN IMDEF AND EACH OF THE COMPARED VARIANTS

imDEF vs	Measures		
	F1	G-mean	AUC
imDEF w SM	0.44909_{+}	$\mathbf{3.8e\text{-}06}_{++}$	$\mathbf{0.0389}_{++}$
imDEF w/o Gen	$\mathbf{0.0259}_{++}$	$\mathbf{9.3e\text{-}09}_{++}$	0.5033_{+}
imDEF w Bag	$\mathbf{1.2e\text{-}05}_{++}$	0.78047_{+}	$\mathbf{0.0063}_{++}$
imDEF w AdaB	$\mathbf{0.0270}_{++}$	0.6738_{-}	$\mathbf{0.0210}_{++}$
imDEF w/o DS	$\mathbf{1.5e\text{-}06}_{++}$	0.8832_{-}	$\mathbf{1.2e\text{-}06}_{++}$
imDEF w Sig	$\mathbf{0.0361}_{++}$	$\mathbf{7.5e\text{-}09}_{++}$	0.5644_{-}
imDEF w KNE	$\mathbf{5.0e\text{-}05}_{++}$	$\mathbf{1.3e\text{-}07}_{++}$	$\mathbf{2.1e\text{-}07}_{++}$
imDEF w KNU	$\mathbf{0.0024}_{++}$	0.2450_{+}	$\mathbf{0.0147}_{++}$
imDEF w desP	$\mathbf{0.0190}_{++}$	$\mathbf{5.3e\text{-}05}_{++}$	$\mathbf{0.0118}_{++}$
imDEF w KnoP	$\mathbf{0.0009}_{++}$	$\mathbf{0.0003}_{++}$	$\mathbf{0.0019}_{++}$
imDEF w desRRC	$\mathbf{1.4e\text{-}06}_{++}$	0.6201_{+}	$\mathbf{6.9e\text{-}06}_{++}$
imDEF w desKL	$\mathbf{4.7e\text{-}07}_{++}$	$\mathbf{0.0405}_{++}$	$\mathbf{2.7e\text{-}05}_{++}$
imDEF w METADES	$\mathbf{8.0e\text{-}06}_{++}$	$\mathbf{3.7e\text{-}09}_{++}$	$\mathbf{2.6e\text{-}08}_{++}$

process and 50% for dynamic selection [4]. The complete experimental results are provided in Table S27 of the supplementary material. Based on Table S27, the corresponding results of the Wilcoxon signed-rank tests are presented in Table X. We could see that imDEF shows the significance advantages on almost all of the comparisons.

Based on the above comparisons, it could be summarized that the three key ingredients OREM_{G}, SPSE_{EM}, and SPSE_{EM} all play a positive role for the performance of imDEF. They together contribute to the effectiveness of imDEF.

5) Parameter Sensitivity Analysis: imDEF involves five parameters, i.e., q, b, α_1, α_2, and β. q and b can refer to the corresponding settings in OREM and SPE, respectively. We only discuss here how the performance of imDEF varies with different α_1, α_2, and β.

Eight datasets in Table III are selected in this experiment. They are *Diabetes*, *Vehicle*, *Laryngeal-3*, *Voice9*$_1$, *Win-red*, *Voice9*$_2$, *ESL*$_T$, *Cervical cancer*. We vary $\alpha_1 = \{0.3, 0.5, 0.7, 0.9, 1.1\}$, $\alpha_2 = \{0.4, 0.6, 0.8, 1, 1.2\}$, and $\beta = \{5, 15, 25, 35, 45\}$. Fig. 4 shows the mean rankings of imDEF over different parameter values in terms of ACC_{min} (the classification accuracy in the minority class), ACC_{maj} (the accuracy in the majority class), and G-mean. The detailed experimental results are given in Table S28 of the supplementary material.

In Fig. 4(a) and (b), the performance of imDEF in ACC_{min} is deteriorated, while the performance in ACC_{maj} is improved with the decrease of α_1 or α_2. We deem the reason for this phenomenon is that the number of synthetic minority (/majority) samples in $\{\boldsymbol{Tr_r}\}_{r=1}^{\lfloor n \rfloor}$ is more (/less) than the number of original minority (/majority) samples, especially for $r > \lfloor n \rfloor/2$. That is, the synthetic datasets are derived from the original imbalanced data by oversampling the minority class and undersampling the majority class. Hence, the borderline majority samples might be easily misclassified, and become the hard samples. From line 8 of Algorithm 2 (/line 10 of Algorithm 3), we can know that the decrease of α_1 (/α_2) implies the self-paced factor s_t (/s_g) can be diminished to a lower value. Hence, more attention would be focused on those difficult samples (/hard-to-classify samples w.r.t the classifier pool). This could explain why their decrease could improve ACC_{maj}. On the contrary, ACC_{min} would be depreciated, as relatively less attention is paid to the minority samples. From Fig. 4(a) and (b), we can find that α_1 and α_2 could achieve the best tradeoff between ACC_{maj} and ACC_{min} (i.e., a good G-mean) around 0.7 and 1.0, respectively.

β is the number of referees in a referee committee. A larger β means that more referees would be created for each component classifier to learn its competence region. Intuitively, increasing β is conducive to obtaining more accurate competence estimation. As shown in Fig. 4(c), the G-mean of imDEF is improved with the increase of β, and 35 is a good setting. In this experimental study, we used a small β (i.e., 5). Nevertheless, imDEF could still outperform the other ensemble solutions significantly.

B. Experiments on Multiclass Imbalanced Datasets

Most imbalanced learning methods are specifically designed for two-class imbalance scenarios. They cannot combat multiclass imbalance problems directly. However, multiclass data is more prone to occur skewed class distributions than binary data, and multiclass imbalance is highly challenging due to the presence of multi-majority classes, multi-minority classes, and severe data difficulty factors (e.g., multiclass overlapping) [18], [39]. Therefore, it is highly meaningful that the proposed imbalance approach has the ability to deal with multiclass imbalance problems. In this subsection, we would evaluate the effectiveness of imDEF over multiclass imbalanced datasets.

TABLE XI
DESCRIPTION OF CHARACTERISTICS OF EXPERIMENTAL MUTLICLASS DATASETS

Dataset	Class distribution	|L|	|F|	IR
Hayes-Roth	51/51/30	3	4	1.40
Vertebral	60/150/100	3	6	2.67
Dermatology	60/71/20	3	35	4.73
New-thyroid	150/35/30	3	5	7.45
Toy	35/87/79/68/31	5	2	8.91
Balance	49/288/288	3	4	9.76
Cleveland	160/54/35/35	4	13	10.19
Voice	158/43/115/59/38	5	10	13.59
Page-blocks	329/28/88/115	4	10	20.91
SWD	399/352/217/32	4	10	28.84
Auto5	91/131/101/59/10	5	7	37.52
LEV	403/27/280/93/197	5	4	40.41
Leaves-plant	128/16/16/16/16/80	6	64	44.60
Stock10	48/110/108/119/168/ 104/104/103/64/22	10	9	53.60
ESL	62/14/351/38/23	5	4	60.00
Heating	20/265/112/51/ 119/85/82/34	8	8	61.98
Abalone	391/568/689/103/67/58	6	10	62.91
ERA	92/771/88/31/18	5	4	93.42

The overall imbalance ratio (IR) is computed as [39].

1) Experimental Setting: Eighteen multiclass imbalanced datasets are used in this experiment. They include the common ordinal regression benchmark datasets [33] and UCI datasets [34]. These datasets are selected by considering the diversities in sample size, feature dimension, number of classes, and application domain. Table XI summarizes the characteristics of them. We still choose classification regression tree as $base\ classifier$, and use the Wilcoxon signed rank test to conduct the $statistical\ analysis$ of significance. For $assessment\ measures$, macro-F1, MG [40], and MAUC [41] are used together. They are the multiclass extension versions of F1, G-mean, and AUC, respectively.

2) Comparison With Existing Multiclass Imbalanced Ensemble Approaches: Given that there are only a few imbalanced ensemble methods for solving multiclass imbalance problems, our imDEF is compared with both static and dynamic ensemble solutions simultaneously. In the static ensemble methods, AdaBoost.M2 (AdaB) [12], SMOTEB, MBSB [42], OREMB, and AdaBoost.AD (AdaAD) [43] are chosen. The descriptions and default parameter settings of them are listed in Tables IV and V, respectively. In each iteration of SMOTEB, MBSB [42], and OREMB, all the classes (except the largest class) are oversampled to achieve complete class balance. In the dynamic ensemble solutions, we select B+RM100+KNU, OLP+KNN [37], [38], BR+RM+KNU, and BR+KNU+RMkNN [22]. The classifier pools of size 100 and 600 are generated for B+RM100+KNU, BR+RM+KNU, and BR+KNU+RMkNN, respectively. Tables VII and VIII present the configurations and parameter settings of them, respectively.

The macro-F1, MG, and MAUC values of all the compared algorithms are summarized in Table S29 of the supplementary material. According to Table S29, imDEF acquires the best macro-F, MG, and MAUC performances on 9, 8, and 7 out of the 18 datasets, respectively. The results of the significance tests are given in Table XII. We can observe that imDEF is almost significantly better than all the compared methods. It suggests that imDEF is also highly competitive in dealing with complex multiclass imbalance problems.

TABLE XII
p-VALUES OF THE WILCOXON SIGNED-RANK TESTS FOR THE COMPARISONS BETWEEN IMDEF AND EACH OF THE COMPARED STATIC AND DYNAMIC ENSEMBLE SOLUTIONS OVER 18 MULTICLASS IMBALANCED DATASETS

imDEF vs	Measures		
	macro-F	MG	MAUC
AdaB	**0.0139**$_{++}$	**3.8e-05**$_{++}$	**9.2e-05**$_{++}$
SMOTEB	**0.0028**$_{++}$	**3.8e-05**$_{++}$	**0.0011**$_{++}$
MBSB	**0.0001**$_{++}$	**2.3e-05**$_{++}$	**1.5e-05**$_{++}$
OREMB	**0.0028**$_{++}$	**0.0001**$_{++}$	**0.0004**$_{++}$
AdaAD	**0.0003**$_{++}$	**0.0056**$_{++}$	**1.5e-05**$_{++}$
B_{100}+RM100+KNU	**0.0101**$_{++}$	**7.6e-06**$_{++}$	**0.0027**$_{++}$
OLP+KNN	**3.1e-05**$_{++}$	**2.3e-05**$_{++}$	**1.5e-05**$_{++}$
BR_{100}+RM+KNU	**0.0077**$_{++}$	0.4951$_{+}$	**0.0064**$_{++}$
BR_{100}+KNU+RMkNN	**0.0077**$_{++}$	**0.0237**$_{++}$	**0.0007**$_{++}$
B_{600}+RM100+KNU	**0.0237**$_{++}$	**1.5e-05**$_{++}$	**0.0483**$_{++}$
BR_{600}+RM+KNU	**0.0311**$_{++}$	0.8650$_{+}$	**0.0332**$_{++}$
BR_{600}+KNU+RMkNN	**0.0090**$_{++}$	**0.0066**$_{++}$	**0.0015**$_{++}$

V. CONCLUSION

imDEF involves three key tasks: generation of artificial synthetic datasets, creation of classifier pool, construction of referee system. In this paper, our imDEF uses OREM_{G}, SPSE_{CE}, and SPSE_{EM} to yield synthetic datasets, build classifier pool, and construct referee system, respectively. To further enhance imDEF's performance, future research works could be devoted to designing more effective algorithms to replace OREM_{G}, SPSE_{CE}, or SPSE_{EM}. For example, OREM_{G} requires the distance calculations between any two samples, and can only deal with numerical data. A more efficient and universal data generation method can be developed to handle imbalanced massive mixed data. In addition, self-paced boosting algorithms could be attempted to create classifier pool, which focuses on the insufficiently learned but reliable synthetic samples, and suppresses the effects of noisy synthetic samples.

REFERENCES

[1] H. Kaur, H. S. Pannu, and A. K. Malhi, "A systematic review on imbalanced data challenges in machine learning: Applications and solutions," *ACM Comput. Surv.*, vol. 52, no. 4, pp. 1–36, 2019.
[2] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches," *IEEE Trans. Syst., Man, Cybern., Part C. (Appl. Rev.)*, vol. 42, no. 4, pp. 463–484, 2011.
[3] 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.
[4] R. M. Cruz, R. Sabourin, and G. D. Cavalcanti, "META-DES.H: A dynamic ensemble selection technique using meta-learning and a dynamic weighting approach," in *Proc. 2015 Int. Joint Conf. Neural Netw.*, 2015, pp. 1–8.

[5] A. H. Ko, R. Sabourin, and A. S. Britto Jr, "From dynamic classifier selection to dynamic ensemble selection," *Pattern Recognit.*, vol. 41, no. 5, pp. 1718–1731, 2008.
[6] 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.*, Springer, 2003, pp. 107–119.
[7] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Trans. Syst., Man, Cybern., Part B. (Cybern.)*, vol. 39, no. 2, pp. 539–550, Apr. 2009.
[8] S. Chen, H. He, and E. A. Garcia, "RAMOBoost: Ranked minority oversampling in boosting," *IEEE Trans. Neural Netw.*, vol. 21, no. 10, pp. 1624–1642, Oct. 2010.
[9] J. F. Díez-Pastor, J. J. Rodríguez, C. Garcia-Osorio, and L. I. Kuncheva, "Random balance: Ensembles of variable priors classifiers for imbalanced data," *Knowl.-Based Syst.*, vol. 85, pp. 96–111, 2015.
[10] V. Iosifidis, S. Papadopoulos, B. Rosenhahn, and E. Ntoutsi, "AdaCC: Cumulative cost-sensitive boosting for imbalanced classification," *Knowl. Inf. Syst.*, vol. 65, no. 2, pp. 789–826, 2023.
[11] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, pp. 123–140, 1996.
[12] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, 1997.
[13] G. Giacinto and F. Roli, "Dynamic classifier selection based on multiple classifier behaviour," *Pattern Recognit.*, vol. 34, no. 9, pp. 1879–1881, 2001.
[14] T. Woloszynski and M. Kurzynski, "A probabilistic model of classifier competence for dynamic ensemble selection," *Pattern Recognit.*, vol. 44, no. 10/11, pp. 2656–2668, 2011.
[15] T. Woloszynski, M. Kurzynski, P. Podsiadlo, and G. W. Stachowiak, "A measure of competence based on random classification for dynamic ensemble selection," *Inf. Fusion*, vol. 13, no. 3, pp. 207–213, 2012.
[16] P. R. Cavalin, R. Sabourin, and C. Y. Suen, "Dynamic selection approaches for multiple classifier systems," *Neural Comput. Appl.*, vol. 22, pp. 673–688, 2013.
[17] M. R. Smith, T. Martinez, and C. Giraud-Carrier, "An instance level analysis of data complexity," *Mach. Learn.*, vol. 95, pp. 225–256, 2014.
[18] T. Zhu, X. Liu, and E. Zhu, "Oversampling with reliably expanding minority class regions for imbalanced data learning," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 6, pp. 6167–6181, Jun. 2023.
[19] P. Z. Sun, T.-Y. Zuo, R. Law, E. Q. Wu, and A. Song, "Neural network ensemble with evolutionary algorithm for highly imbalanced classification," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 7, no. 5, pp. 1394–1404, Oct. 2023.
[20] J. Xiao et al., "A novel deep ensemble model for imbalanced credit scoring in internet finance," *Int. J. Forecasting*, vol. 40, no. 1, pp. 348–372, 2024.
[21] F. Zhou et al., "Dynamic self-paced sampling ensemble for highly imbalanced and class-overlapped data classification," *Data Mining Knowl. Discov.*, vol. 36, no. 5, pp. 1601–1622, 2022.
[22] L. M. Junior, F. M. Nardini, C. Renso, R. Trani, and J. A. Macedo, "A novel approach to define the local region of dynamic selection techniques in imbalanced credit scoring problems," *Expert Syst. Appl.*, vol. 152, 2020, Art. no. 113351.
[23] S. M. Liu, J.-H. Chen, and Z. Liu, "An empirical study of dynamic selection and random under-sampling for the class imbalance problem," *Expert Syst. Appl.*, vol. 221, 2023, Art. no. 119703.
[24] A. Roy, R. M. Cruz, R. Sabourin, and G. D. Cavalcanti, "A study on combining dynamic selection and data preprocessing for imbalance learning," *Neurocomputing*, vol. 286, pp. 179–192, 2018.
[25] C. Chen, A. Liaw, and L. Breiman, "Using random forest to learn imbalanced data," Univ. California, Berkeley, CA, USA, Tech. Rep., vol. 110, no. 112, p. 24, 2004.
[26] L. M. Junior, F. M. Nardini, C. Renso, and J. A. F. de Macêdo, "On combining dynamic selection, sampling, and pool generators for credit scoring," in *Proc. Int. Conf. Mach. Learn. Data Mining Pattern Recognit.*, 2019, pp. 443–457.
[27] S. García, Z.-L. Zhang, A. Altalhi, S. Alshomrani, and F. Herrera, "Dynamic ensemble selection for multi-class imbalanced datasets," *Inf. Sci.*, vol. 445, pp. 22–37, 2018.
[28] K. Wang, Y. Wang, Q. Zhao, D. Meng, X. Liao, and Z. Xu, "SPLBoost: An improved robust boosting algorithm based on self-paced learning," *IEEE Trans. Cybern.*, vol. 51, no. 3, pp. 1556–1570, Mar. 2021.
[29] P. M. Long and R. A. Servedio, "Random classification noise defeats all convex potential boosters," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 608–615.
[30] N. M. Wanas, R. A. Dara, and M. S. Kamel, "Adaptive fusion and co-operative training for classifier ensembles," *Pattern Recognit.*, vol. 39, no. 9, pp. 1781–1794, 2006.
[31] W. Gao and Z.-H. Zhou, "On the doubt about margin explanation of boosting," *Artif. Intell.*, vol. 203, pp. 1–18, 2013.
[32] D. Mahajan et al., "Exploring the limits of weakly supervised pretraining," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 181–196.
[33] W. Chu and Z. Ghahramani, "Gaussian processes for ordinal regression," *J. Mach. Learn. Res.*, vol. 6, pp. 1019–1041, 2005.
[34] D. Dua and E. K. Taniskidou, "UCI machine learning repository," 2024. [Online]. Available: http://archive.ics.uci.edu/ml
[35] A. Benavoli, G. Corani, and F. Mangili, "Should we really use post-hoc tests based on mean-ranks," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 152–161, 2016.
[36] X. Liang, Y. Gao, and S. Xu, "ASE: Anomaly scoring based ensemble learning for highly imbalanced datasets," *Expert Syst. Appl.*, vol. 238, 2024, Art. no. 122049.
[37] M. A. Souza, G. D. Cavalcanti, R. M. Cruz, and R. Sabourin, "On evaluating the online local pool generation method for imbalance learning," in *Proc. 2019 Int. Joint Conf. Neural Netw.*, 2019, pp. 1–8.
[38] M. A. Souza, G. D. Cavalcanti, R. M. Cruz, and R. Sabourin, "Online local pool generation for dynamic classifier selection," *Pattern Recognit.*, vol. 85, pp. 132–148, 2019.
[39] T. Zhu, Y. Lin, and Y. Liu, "Synthetic minority oversampling technique for multiclass imbalance problems," *Pattern Recognit.*, vol. 72, pp. 327–340, 2017.
[40] Y. Sun, M. S. Kamel, and Y. Wang, "Boosting for learning multiple classes with imbalanced class distribution," in *Proc. Int. Conf. Data Mining*, 2006, pp. 592–602.
[41] D. J. Hand and R. J. Till, "A simple generalisation of the area under the ROC curve for multiple class classification problems," *Mach. Learn.*, vol. 45, no. 2, pp. 171–186, 2001.
[42] C.-L. Liu and P.-Y. Hsieh, "Model-based synthetic sampling for imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 8, pp. 1543–1556, Aug. 2020.
[43] S. Li, L. Song, X. Wu, Z. Hu, Y.-M. Cheung, and X. Yao, "Multi-class imbalance classification based on data distribution and adaptive weights," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 10, pp. 5265–5279, Oct. 2024.

Tuanfei Zhu received the PhD degree from Hunan University, China. He is now associate professor with the College of Computer Science and Engineering, Changsha University, and is also a post-doctor with the School of Computer, NUDT. He has published more than 10 peer-reviewed articles. His current research interests include imbalanced data learning, ensemble learning.

Xingchen Hu received the PhD degree from the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada, in 2017. He is currently an associate professor with the College of Systems Engineering, NUDT. His research interests include computational intelligence, granular computing, knowledge discovery and data mining, and evolutionary optimization.

Xinwang Liu (Senior Member, IEEE) received the PhD degree from the National University of Defense Technology (NUDT), China. He is now professor with the School of Computer, NUDT. His current research interests include kernel learning and unsupervised feature learning. He has published more than 100 peer-reviewed papers, including those in highly regarded journals and conferences, such as *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Image Processing*, *IEEE Transactions on Neural Networks and Learning Systems*, *IEEE Transactions on Multimedia*, *IEEE Transactions on Information Forensics and Security*, ICML, NeurIPS, CVPR, ICCV, AAAI, IJCAI, etc. He is an associate editor of *IEEE Transactions on Neural Networks and Learning Systems* and *Information Fusion Journal*. More information can be found at https://xinwangliu.github.io/.

En Zhu received the PhD degree from the National University of Defense Technology (NUDT), China. He is now professor with the School of Computer Science, NUDT, China. His main research interests are pattern recognition, image processing, machine vision and machine learning. He has published more than 60 peer-reviewed papers, including *IEEE Transactions on Circuits and Systems for Video Technology*, *IEEE Transactions on Neural Networks and Learning Systems*, *Pattern Recognition*, AAAI, IJCAI, etc. He was awarded China National Excellence Doctoral Dissertation.

Xinzhong Zhu received the PhD degree from Xidian University, China. He is a professor with the College of Computer Science and Technology, Zhejiang Normal University, Jinhua, China. His research interests include machine learning, computer vision, manufacturing informatization, intelligent manufacturing, kernel learning, feature selection, and multiview clustering. He has published more than 30 peer-reviewed papers, including *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *IEEE Transactions on Image Processing*, *IEEE Transactions on Multimedia*, *IEEE Transactions on Knowledge and Data Engineering*, CVPR, NeurIPS, etc.

Huiying Xu received the MS degree from the National University of Defense Technology (NUDT), China. She is an associate professor with the School of Computer Science and Technology, Zhejiang Normal University. Her research interests include object detection, image processing, pattern recognition, deep clustering, generative adversarial network, diffusion model, clustering ensemble, multiple kernel learning. She has published papers, including *IEEE Transactions on Cybernetics*, *IEEE Transactions on Multimedia*, etc.