Learning From Imbalanced Data With Deep Density Hybrid Sampling

Chien-Liang Liu, *Member, IEEE*, and Yu-Hua Chang

***Abstract*—Learning from imbalanced data is an important and challenging topic in machine learning. Many works have devised methods to cope with imbalanced data, but most methods only consider minority or majority classes without considering the relationship between the two classes. In addition, many synthetic minority oversampling technique-based methods generate synthetic samples from the original feature space and use the Euclidean distance to search for the nearest neighbors. However, the Euclidean distance is not a precise distance metric in a high-dimensional space. This article proposes a novel method, called deep density hybrid sampling (DDHS), to address imbalanced data problems. The proposed method learns an embedding network to project the data samples into a low-dimensional separable latent space. The goal is to preserve class proximity during data projection, and we use within-class and between-class concepts to devise loss functions. We propose to use density as a criterion to select minority and majority samples. Subsequently, we apply a feature-level approach to the selected minority samples and generate diverse and valid synthetic samples for the minority class. This work conducts extensive experiments to assess our proposed method and compare it with several methods. The experimental results show that the proposed method can yield promising and stable results. The proposed method is a data-level algorithm, and we combine the proposed method with the boosting technique to develop a method called DDHS-boosting. We compare DDHS-boosting with several ensemble methods, and DDHS-boosting shows promising results.**

***Index Terms*—Class imbalance, embedding network, hybrid sampling, imbalanced data, synthetic data.**

I. Introduction

LEARNING from imbalanced data has become one of the most important problems in machine learning, as it poses a major challenge to cope with the datasets that comprise imbalanced class distributions [1], [2]. Most machine learning algorithms assume that there is a balanced class distribution in the underlying data samples, so the classifiers have a bias toward the majority class when dealing with imbalanced datasets. However, the minority class is always the target of interest in this case. The imbalanced data problem arises in many application domains, such as the medical domain [3] and the retail domain [4], [5].

Many methods have been devised to deal with imbalanced data problems over decades, but it is still a challenging task to achieve stable performance for high-dimensional and imbalanced datasets. Random sampling methods, such as oversampling and undersampling, are popular approaches, as these approaches are easy to implement and can get acceptable results. However, oversampling is prone to overfitting, whereas undersampling may discard valuable information. In addition to random sampling methods, synthetic minority oversampling technique (SMOTE) [6] is an oversampling method that generates synthetic examples for the minority class to balance the class distribution by using linear interpolation between a selected minority sample and one of its k-nearest neighbors. SMOTE only considers the minority class to generate synthetic samples, and a high-dimensional space may negatively affect the performance [7]. Hybrid methods of oversampling and undersampling are an alternative approach in practical settings [8], [9], as the combination may ease the problems caused by oversampling or undersampling methods. However, hybrid sampling methods generate or remove arbitrary data samples, which may violate the original data distribution or characteristics.

This article proposes a novel method, called deep density hybrid sampling (DDHS), to address imbalanced data problems. The proposed method learns an embedding network to project the data samples into a low-dimensional separable latent space. The goal is to preserve class proximity during data projection, and we use within-class and between-class concepts to devise loss functions. We propose to use density as a criterion to select the majority and minority samples of high quality. We apply a feature-level method to the selected minority samples and aim to generate synthetic samples that maintain the properties of the minority class while enlarging the diversity of the generated samples.

In the experiments, we compare our proposed method with several classical and state-of-the-art methods on 13 datasets. The experimental results show that the proposed method can yield promising and stable results. The results show that our proposed method works very well in low- and high-dimensional datasets. The proposed method is a data-level algorithm, and we combine the proposed method with the boosting technique to develop a method called DDHS-boosting. We compare DDHS-boosting with several ensemble methods, and DDHS-boosting also shows promising results.

Manuscript received 11 September 2021; accepted 30 January 2022. Date of publication 1 March 2022; date of current version 17 October 2022. This work was supported in part by the Ministry of Science and Technology, Taiwan, under Grant MOST 109-2628-E-009-009-MY3. This article was recommended by Associate Editor L. Wang. *(Corresponding author: Chien-Liang Liu.)*

The authors are with the Department of Industrial Engineering and Management, National Yang Ming Chiao Tung University, Hsinchu 30010, Taiwan (e-mail: clliu@nycu.edu.tw).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSMC.2022.3151394.

Digital Object Identifier 10.1109/TSMC.2022.3151394

The contributions of the proposed method are listed as follows. First, this work proposes a novel method to deal with imbalanced data problems. The key idea is to preserve class proximity during data projection, giving a base to learn a low-dimensional separable space. We propose to use density as the criterion to select high-quality minority and majority samples for the subsequent generation of synthetic examples and model training. Second, we conduct extensive experiments on 13 datasets and compare the proposed method with many methods. The experimental results indicate that the proposed method shows promising results in low- and high-dimensional datasets. Finally, we conduct extensive experiments to analyze our proposed method. We also show that the proposed method can use the boosting technique to become an ensemble learning method.

This article is organized as follows. Section II conducts a literature review of the methods that focus on imbalanced data problems. Section III introduces the proposed method. Next, Section IV lists the experimental settings and experimental results. Section V analyzes the experimental results and discusses the proposed method. Finally, the conclusions are drawn in Section VI.

II. Related Work

We introduce methods that are developed to deal with imbalanced data problems, including data-level, algorithm-level, ensemble learning, and deep learning approaches.

A. Data-Level Methods

The key idea behind data-level methods is to balance the class distributions so that the training of the subsequent classifiers will not have a bias toward the majority class. One advantage of data-level methods is that these methods can be integrated with any classifier, which is why they are popular in practical settings.

Among data-level methods, random sampling is one of the most popular methods for dealing with imbalanced problems, as it is easy to implement and can yield acceptable performance. Among random sampling methods, oversampling duplicates data samples from the minority class, whereas undersampling removes data samples from the majority class. In addition to these two approaches, hybrid methods that combine oversampling and undersampling are also popular in practical settings.

In addition to the random sampling methods, SMOTE generates synthetic data samples for the minority class to balance the class distribution, in which the generated data sample lies in the line segment between two data samples from the minority class. The diversity of the generated data samples is limited, so Liu and Hsieh [10] proposed a method called model-based synthetic sampling (MBS) to address the imbalanced problem by employing linear regression to capture feature relationships and generate synthetic samples based on the feature relationship.

Oversampling may suffer from the overfitting problem by repeatedly generating the same samples, whereas undersampling may discard representative samples, resulting in information loss problems. SMOTE relies on a distance metric to determine the k nearest neighbors, and a typical choice is to use the Euclidean distance. However, the Euclidean distance is imprecise for calculating the distance between two data points in a high-dimensional space.

The Tomek link [11] is an undersampling method by removing overlapping samples, in which a Tomek link defines the overlap. Given a pair (x_i, x_j), where x_i is a minority sample and x_j is a majority sample, and $d(x_i, x_j)$ denotes the distance between x_i and x_j. A pair (x_i, x_j) forms a Tomek link when there is no data sample x_k that satisfies $d(x_i, x_k) < d(x_i, x_j)$ or $d(x_j, x_k) < d(x_i, x_j)$. Here, x_i and x_j are near the border of the two classes, so x_j may be an overlapping sample that should be eliminated from the majority samples.

Borderline-SMOTE [12] extends SMOTE, and the idea is like the Tomek link. All minority samples are equally important in SMOTE, but the samples at the boundary should be different as they are more likely to be classified incorrectly. Therefore, Borderline-SMOTE categorizes the minority data into three levels: 1) safe level; 2) danger level; and 3) noise level. Borderline-SMOTE selects the data belonging to danger level to generate synthetic samples.

Safe-level-SMOTE [13] tackles the problem of class overlap caused by SMOTE. Each minority class is assigned a safe score before generating synthetic samples. Subsequently, the synthetic data samples are generated based on different cases, enabling the synthetic samples to be distributed in the safe area. Noise reduction *a priori* synthetic (NRAS) [14] is an oversampling method that focuses on noise reduction and data generation. Using non-noise minority samples to generate synthetic data samples avoids the problems caused by the noise samples.

Random oversampling methods generate data samples without considering the underlying data distributions, so many previous works have proposed to generate data samples according to the estimated density function. Kernel density estimation (KDE) is a popular estimation approach, as KDE is a nonparametric method. The implementation is to estimate the density function by averaging over a collection of kernel homogeneous functions, each of which is centered at each data point. Once the estimation of the density distribution for the minority class is completed, it can be used to generate new data samples for the minority class [15]–[17].

B. Algorithm-Level Methods

Algorithm-level methods address the class imbalance problem by using different misclassification costs to enable the classifiers to place more emphasis on the minority class [9], [18], [19]. Assume that the minority class is a positive class in the classification outcomes, a false negative prediction should be assigned a higher cost compared to a false positive prediction in the imbalanced data problem. Classification algorithms can take these costs into account during model training, and this can alleviate problems with imbalanced data.

Sun *et al.* [18] investigated meta-techniques to tackle the imbalanced data problems. They proposed three cost-sensitive boosting algorithms by integrating cost-sensitive learning with

AdaBoost to improve the classification performance on the positive class. Zhou and Liu [20] applied three different methods, including sampling, threshold-moving, and soft-ensemble, to devise cost-sensitive neural networks and empirically showed that threshold-moving and soft-ensemble yielded promising results when combined with cost-sensitive neural networks.

Jeatrakul *et al.* [9] proposed an algorithm that combines complementary neural network (CMTNN) with SMOTE to deal with problems of imbalanced data. Their proposed method combines undersampling and oversampling techniques. More specifically, CMTNN uses a pair of complementary feedforward neural networks called Truth NN and Falsity NN to detect and remove misclassified data samples.

C. Ensemble Methods

Ensemble learning is another popular approach for imbalanced data problems, as it can alleviate prediction errors and bias by using multiple classifiers. For instance, Wang and Yao [21] proposed SMOTEBagging that involves Bagging and SMOTE techniques in the construction of multiclassifiers to diversify synthetic samples. The final classification is based on the majority vote. Compared to SMOTEBagging, Chawla *et al.* [22] combined SMOTE with a boosting algorithm to propose a method called SMOTEBoost to generate synthetic examples from the minority class during each boosting iteration. With the combination of weak classifiers, a boosting algorithm can be more accurate than a single classifier. The new weak classifier learns the characteristics of misclassified samples by increasing the weight of samples that are misclassified by the previous weak classifiers. The model can improve the performance during each iteration of the boosting algorithm. Unlike SMOTEBagging and SMOTEBoost, which are based on oversampling, Seiffert [23] proposed RUSBoost based on random undersampling. The benefit of RUSBoost is that it reduces training time and uses AdaBoost to improve performance.

Liu *et al.* [24] introduced the concept of classification hardness to evaluate the difficulty of a classifier based on the number of correct predictions for a classifier. The noises in a dataset should have higher hardness values. Thus, they proposed a learning framework called self-paced ensemble (SPE) based on the hardness value by selecting only informative majority samples to keep representative information. SPE learning is an undersampling framework and does not need any distance metric to estimate the hardness values.

D. Deep Learning

The key idea behind deep learning is to use deep neural networks to learn feature representations and perform classification tasks simultaneously. Although deep learning has had brilliant success in many application domains, it also assumes that there is a balanced class distribution in the dataset. Buda *et al.* [25] have conducted extensive experiments on three benchmark datasets to investigate the effects of imbalance on convolutional neural networks (CNNs). They concluded that the effect of class imbalance on CNN is detrimental. Besides, they examined seven methods that are used in deep learning to deal with CNN training on an imbalanced dataset. The experimental results showed that oversampling is the dominant method. Havaei *et al.* [3] proposed a two-phase training to deal with the brain tumor segmentation problem, which is an extremely imbalanced data problem. The two-phase training is a transfer learning technique, in which the first phase uses a balanced dataset to pretrain the model, while the second phase uses the fine-tuning technique to keep the output layers with a more representative distribution of the labels. He *et al.* [26] proposed cost-effective semisupervised learning with crowd framework (CSLC) that focused on multivariate time series data by integrating two sampling strategies and a cost-sensitive crowd labeling approach. Wang *et al.* [27] combined area under the curve (AUC) maximization and extreme learning machines (ELMs) [28] framework called AUC-ELM and SAUC-ELM to deal with imbalanced binary classification problems.

Khan *et al.* [29] proposed a cost-sensitive deep neural network (CoSen) to learn feature representations for the majority and minority classes. Their proposed method jointly optimizes the network parameters and the class-sensitive costs, so CoSen is an algorithm-level method. Arefeen *et al.* [30] proposed two undersampling methods (NUS-1 and NUS-2) based on neural networks to remove majority samples that overlap with the minority class.

III. Proposed Method

Many data-level methods extend the concept of SMOTE to generate synthetic examples by searching for the k nearest minority examples, leading to two problems when confronted with imbalanced and high-dimensional datasets. First, it only considers minority samples and ignores the influence brought by the majority class. Second, the Euclidean distance is the most popular distance metric to determine the nearest neighbors. However, using the Euclidean distance cannot accurately calculate the distance between data samples in high-dimensional datasets. Many previous works have shown that learning models can benefit from projecting data points to an optimal lower-dimensional representation. The projection can help the spectral clustering identify the optimal neighbors of each data point [31] and unravel the intrinsic cluster structure [32]. Moreover, the calculation of the semantic correlation between the query terms and the concepts can be more accurate [33].

Therefore, we propose DDHS to resolve the previous problems, and Fig. 1 shows the steps involved in our proposed method. The first step is to use a learnable embedding network to project data points into a low-dimensional separable latent space so that all data points in the latent space can preserve the property of class proximity. In the second step, we use density as a criterion to select high-quality data samples. Subsequently, we apply a feature-level oversampling to the selected minority samples to generate synthetic samples. Next, a verification step is performed to ensure that the synthetic samples generated meet the requirements. Finally, we can merge all minority samples with the synthetic samples generated to form the final

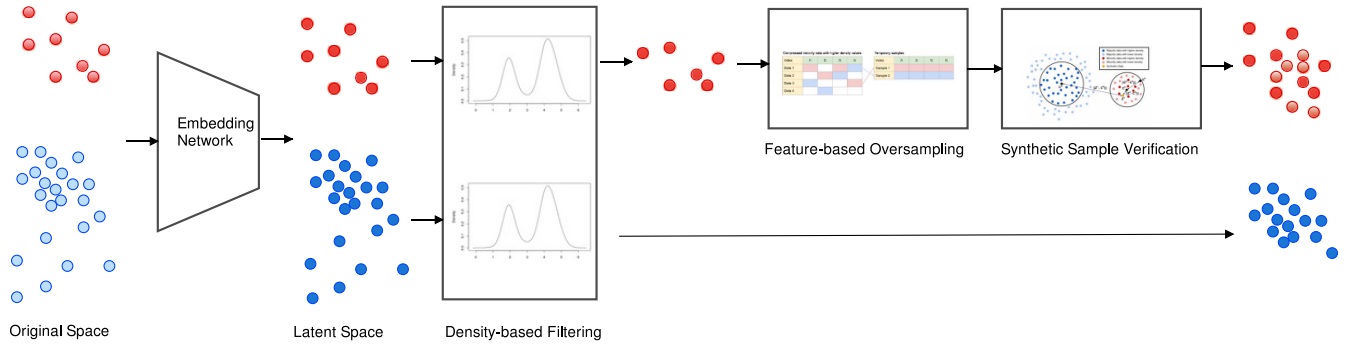


Fig. 1. Proposed method.

minority set. The majority samples that satisfy the density criterion are used as the majority set. The proposed method is a hybrid approach since we perform oversampling on the minority class and undersampling on the majority class.

A. Notation

This section introduces the notation that is used in this article. The input data is a collection of training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$, where $\mathbf{x}_i \in \mathbb{R}^D$ is a feature vector, y_i is the corresponding label and m is the number of training examples. This work proposes to project each data sample $\mathbf{x}_i$ into a lower-dimensional space, and we assume that the projected data point for $\mathbf{x}_i$ is $\mathbf{z}_i \in \mathbb{R}^d$, where $d < D$. Besides, this work focuses on the binary classification task, so the prediction outcomes comprise positive (minority class) and negative (majority class) labels.

B. Embedding Network

Many approaches can learn the latent space, and this work proposes to extend autoencoder to develop our proposed method. Traditional autoencoder comprises two parts: 1) an encoder and 2) a decoder. The encoder aims to transform the original data into a low-dimensional latent representation, while the decoder maps the latent representation to the reconstruction of the original data. The goal is to optimize the encoder and decoder simultaneously by minimizing the reconstruction loss, as shown in (1), in which $\mathbf{x}$ is the input data, E is the encoder, D is the decoder, and $\mathscr{L}_R$ is the reconstruction loss

$$E^*, D^* = \arg\min_{E,D} \mathscr{L}_R(\mathbf{x}, D(E(\mathbf{x}))). \tag{1}$$

For problems involving labeled data, considering label information in the training process can improve the prediction performance. Thus, we further consider label information to learn the latent space, and the goal is to preserve the property of class proximity. Fig. 2 presents the structure of our proposed method.

The proposed approach involves three loss functions. The first is the reconstruction loss, which is used by the autoencoder. We use the mean-square error (MSE) as the reconstruction loss as shown in (2), in which $\mathbf{x}_i$ represents the original data point, $\hat{\mathbf{x}}_i$ denotes the reconstructed data point, and b is the batch size. Using reconstruction loss helps to learn a space

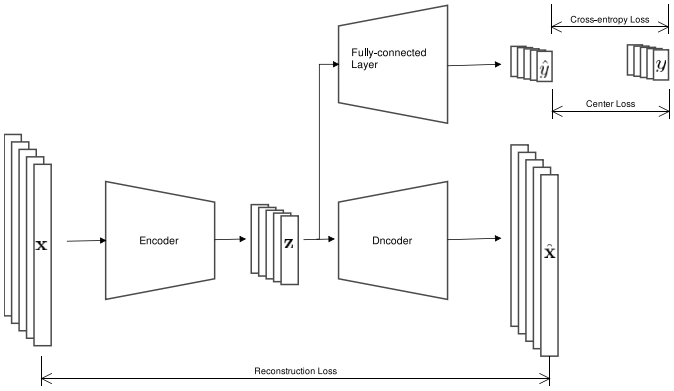


Fig. 2. Schematic of supervised autoencoder.

where data points can be reconstructed from the latent space, giving the basis to eliminate the noise in the data

$$\mathscr{L}_R = \frac{1}{b}\sum_{i=1}^{b} \left\| \mathbf{x}_i - \hat{\mathbf{x}}_i \right\|^2. \tag{2}$$

Besides reconstruction loss, this work proposes to use between-class and within-class losses to model class proximity, in which between-class loss aims to make the data points of different classes as far as possible, and within-class loss makes the data of the same class close. It is apparent that many loss functions can denote between-class and within-class losses once the selected loss functions can satisfy the aforementioned principles. In this work, we propose to use the cross-entropy loss as shown in (3) to represent the between-class loss, in which y and $\hat{y}$ represent the label and the prediction, respectively

$$\mathscr{L}_{CE} = -\frac{1}{b}\sum_{i=1}^{b} \left[y_i \times \log \hat{y}_i + (1 - y_i) \times \log(1 - \hat{y}_i) \right]. \tag{3}$$

The cross-entropy loss gives more penalties to the data points that are classified into incorrect classes, which is why we propose to use cross-entropy loss to denote the between-class loss. We use center loss [34] to denote within-class loss and (4) presents the definition of center loss, in which $c_{y_i} \in \mathbb{R}^d$ denotes the center of the y_i class in the latent space. The center loss gives more penalties to the data points that are far away from the class center, forcing the data points of the same class

to move toward the class center

$$\mathscr{L}_C = \frac{1}{2}\sum_{i=1}^{b}\left|\left|\mathbf{z}_i - c_{y_i}\right|\right|_2^2. \tag{4}$$

The proposed method combines three loss functions to learn the representations in the latent space, and the goal is to make the latent space separable for the projected data points. The dimension of the latent space is much less than that of the original feature space, so the proposed method can alleviate the problem caused by a high-dimensional space. Equation (5) lists the loss function $\mathscr{L}$ used by our proposed method

$$\mathscr{L} = \mathscr{L}_{CE} + \mathscr{L}_C + \mathscr{L}_R. \tag{5}$$

C. Density-Based Filtering

Once the embedding network training is completed, all data points can be projected into the latent space. Next, we propose to use density as a criterion to select high-quality data points. The low-density data point is in a region that is far from other data points, so it may be noise. In contrast, the high-density data point is more likely to be representative of the cluster. Thus, this work defines data points that are in the high-density area as high-quality ones.

We use KDE to calculate the density of points in the latent space, in which KDE is a nonparametric method to estimate the probability density function of a random variable. The free parameter of KDE is the kernel function that specifies the shape of the distribution placed at each point. Many kernel functions, such as the Gaussian kernel, tophat kernel, and linear kernel, have been devised over decades. As mentioned in the previous section, the center loss is used in our proposed method, forcing the data points to move toward the class center. Among these kernels, the Gaussian KDE is the one comprising this property, which is why we select the Gaussian kernel as the kernel function. Besides, the Gaussian kernel has strong computational advantages [35]. The Gaussian kernel function comprises a hyperparameter h, which affects the smoothness of the density.

The value of bandwidth h can be determined by the rule of thumb, cross-validation, and plug-in methods. This work determines the value of h using Scott's rule [36], meaning that h is proportional to $m^{(-1/[d+4])}$, in which m is the number of samples and d is the number of dimensions.

The proposed method calculates the densities for the majority and minority classes, respectively. Next, we assess the quality of the data using the quartile of density. For the majority samples, we keep the samples whose densities are larger than the second quartile of the majority density $\mathbf{Q}_2$. Regarding minority samples, we aim to select the minority samples that are representative of the minority class to generate synthetic examples, so we set a higher criterion than that of the majority samples by using the third quartile, $\mathbf{Q}_3$, of the minority density. As a result, we use 25% of the minority samples as the selected samples in the subsequent generation process, but the remaining 75% of the minority samples are still kept for the classifier training process. On the other hand, we use 50% of the majority of samples to train the classifier.

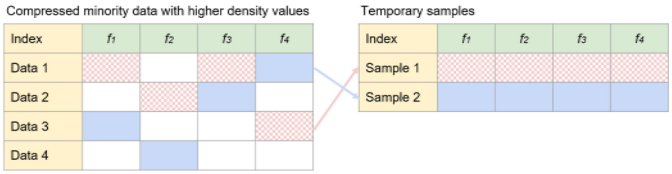


Fig. 3. Generation process of temporary samples.

D. Generation of Synthetic Samples

To generate valid and diverse synthetic samples, we propose to use a feature-level sampling technique to generate samples. Given the feature domain $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d]$ for all the latent features of the selected samples in the minority class, in which $\mathbf{v}_1 = \{\mathbf{v}_{11}, \mathbf{v}_{12}, \ldots, \mathbf{v}_{1m}\}$ defines the value set for the first latent feature, $\mathbf{v}_2 = \{\mathbf{v}_{21}, \mathbf{v}_{22}, \ldots, \mathbf{v}_{2m}\}$ is the value set for the second latent feature, and so on.

In the beginning, the proposed method samples the first latent feature from the set $\mathbf{v}_1$ with a replacement for the synthetic sample. Following the same procedure, we can get the remaining latent features for the synthetic sample. This is a sampling process, so we can get enormous synthetic samples using the above method.

Fig. 3 illustrates the generating process of two synthetic samples. In this example, the number of selected minority samples is 4 and the dimension of the latent space is 4. We simplify the problem to assume that the domain for each feature f_i $(1 \leq i \leq 4)$ comprises four feature values. Note that this is only for demonstration purposes and our proposed method does not have this constraint. To generate a synthetic sample, we randomly select a feature value from its domain sequentially until all features of the synthetic sample are generated. This process can ensure that the feature values for each synthetic sample are valid as they are sampled from their corresponding domains. This is a random process, so we can enlarge the diversity of the synthetic samples.

E. Verification of Synthetic Samples

Once the generation process is completed, quality verification for the synthetic sample is performed to ensure that the synthetic sample is of good quality. This work proposes to use density as the criterion, so we use the point of the highest density of a class to denote the center of the class and the point of the lowest density as the boundary point of the class.

Fig. 4 demonstrates the verification process that involves two criteria for verification. We introduce $\mathbf{c}^M$ and $\mathbf{c}^N$ to represent the centers for the majority and minority classes, respectively. We define the radius of the selected minority class as r^N, which is the distance between the center and the boundary point of the minority class.

Given a synthetic sample $\mathbf{z}^S$, we use two criteria as shown below to verify its quality. The first criterion (6) is to ensure that the synthetic sample $\mathbf{z}^S$ is closer to the center of the minority class than that of the majority class. Furthermore, the second criterion (7) is to ensure that the synthetic sample $\mathbf{z}^S$ is within the radius of the selected minority class. We keep the synthetic samples satisfying the two criteria as the training

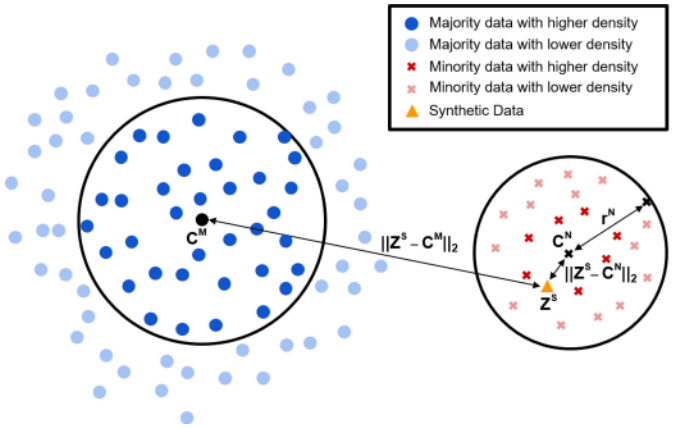


Fig. 4. Verification of temporary samples.

samples

$$\left|\left|\mathbf{z}^S - \mathbf{c}^N\right|\right|_2 < \left|\left|\mathbf{z}^S - \mathbf{c}^M\right|\right|_2 \tag{6}$$

$$\left|\left|\mathbf{z}^S - \mathbf{c}^N\right|\right|_2 < r^N. \tag{7}$$

F. Algorithm for DDHS

Algorithm 1 shows the algorithm of DDHS. The inputs comprise training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ and the oversampling rate p. The output of the algorithm is a processed dataset that is used to train the classifier in the following step. First, the embedding network is trained to learn the discriminative space from the training data, and we use the trained embedding network to project the data into a latent space to obtain the projected data point $\mathbf{z}_i \in \mathbb{R}^d$ (lines 1–4). In the latent space, DDHS uses density-based filtering (DBF) to select the high-quality minority samples and remove the low-quality majority samples (lines 5 and 6). Then, DDHS performs the data generation process and verifies the synthetic data (lines 7–14). The algorithm outputs a new dataset for model training once the above steps are completed.

G. Time Complexity Analysis

The embedding network is an essential step during time complexity analysis, so we focus on the analysis of the embedding network. As shown in Fig. 2, the proposed model comprises an autoencoder and a fully connected layer. The autoencoder comprises an encoder and a decoder, each of which comprises three layers in the implementation. Assume that the first three hidden layers of the encoder comprise n_1, n_2, and n_3 neurons, respectively. Besides, the number of neurons for the fully connected layer is n_f, which is equal to n_3 in our implementation. In particular, the hidden layers of the decoder are in the reverse order of the encoder network. Given the number of data samples m, the number of epochs e, and the dimension of the data sample D, the time complexity for the forward pass in an epoch is $O(2m(Dn_1+n_1n_2+n_2n_3)+m(Dn_1+n_1n_2+n_2n_3+n_3n_f))$. The time complexity of backpropagation is like the forward pass, and n_f is less than n_2. Moreover, the number of neurons for each layer can be considered constant. In this case, the time complexity is about $O(emD)$.

Algorithm 1: DDHS Algorithm

Input: Training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$, where $\mathbf{x}_i \in \mathbb{R}^D$, and over-sampling rate p
Output: Processed dataset

1 Let $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^m$ ($\mathbf{X}^M \leftarrow$ the majority set; $\mathbf{X}^N \leftarrow$ the minority set)
2 Use the training data to train the embedding network with the loss $\mathscr{L}$ defined in Eq. (5), which comprises cross-entropy loss, center loss, and reconstruction loss, and uses the backpropagation algorithm to update the network parameters until convergence
3 Project each training sample $\mathbf{x}_i$ to the latent space with the embedding network and the corresponding sample in the latent space is called $\mathbf{z}_i \in \mathbb{R}^d$
4 Let $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^m$ ($\mathbf{Z}^M \leftarrow$ the projected majority set; $\mathbf{Z}^N \leftarrow$ the projected minority set)
5 Estimate the density of $\mathbf{Z}^M$ and $\mathbf{Z}^N$ by KDE with Gaussian kernel, in which the bandwidth h is determined by Scott's Rule
6 Sort the density and let $\mathbf{Z}^{MH}$ comprise the majority samples whose density larger than $\mathbf{Q}_2$ and $\mathbf{Z}^{NH}$ comprise the minority samples whose density larger than $\mathbf{Q}_3$
7 Collect feature domain $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d]$ from $\mathbf{Z}^{NH}$
8 Let $\mathbf{S}$ be an empty set and $n^s = p \times |\mathbf{Z}^{MH}| - |\mathbf{Z}^N|$
9 **while** $|\mathbf{S}| \leq n^s$ **do**
10 Initialize $\mathbf{s}$ to be an empty vector of size d
11 **for** $j = 1$ *to* d **do**
12 $\mathbf{s}[j] \leftarrow$ randomly sample a feature value from $\mathbf{v}_j$ with replacement
13 **if** $\mathbf{s}$ *pass the verification listed in Eq.* (6)&(7) **then**
14 $\mathbf{S} = \mathbf{S} \cup \mathbf{s}$
15 return $\mathbf{S} \cup \mathbf{Z}^N \cup \mathbf{Z}^{MH}$

IV. Experiments

We conduct several experiments to evaluate our proposed method. Besides, we compare our proposed method with 12 methods.

A. Datasets

We conduct experiments on 13 datasets, all of which are publicly available. The summary of the datasets is listed in Table I, in which the imbalance ratio is the ratio between the number of minority samples and the number of majority samples. The mnist dataset is a modified MNIST image dataset that contains the number zeros from the original MNIST dataset, with the number sixes added as outliers. Furthermore, 100 features of the original 784 (number of pixels) were randomly sampled. In the experiments, we randomly select 80% of the dataset as the training set and the remaining as the test set. Furthermore, we randomly select 20% of the training set as the validation set for model tuning. Note that we use

TABLE I
SUMMARY OF THE DATASETS

Name	Data Size	Numbers of Features	Imbalanced Ratio
Shuttle	49,097	9	7.7% (3511:45586)
Magic	19,020	11	54.23% (6688:12332)
House [1]	22,784	16	42.04% (6744:16040)
Letter	20,000	16	3.81% (734:19266)
Satellite	6,435	36	46.28% (2036:4399)
Spambase	4,601	57	65.03% (1813:2788)
Optdigits	5,216	64	2.96% (150:5066)
Mnist	7,603	100	10.14% (700:6903)
Crime	1,994	100	8.13% (150:1844)
Sylva [2]	14,395	108	6.56% (886:13509)
Nomao	34,465	118	39.98% (9844:24621)
Webpage [3]	34,780	300	2.9% (981:33799)
Isolet	7,797	617	8.34% (600:7197)

Magic, Letter, Spambase, Crime, Nomao, and Isolet can be downloaded from UCI machine learning repository.
Shuttle, Satellite, Optdigits, and mnist can be downloaded from Outlier Detection DataSets [4]

seven datasets that involve low- and high-dimensional datasets in Section IV and the remaining datasets in Section V.

B. Comparison Methods

In the experiments, we compare our proposed method with 11 comparison methods and one baseline method that does not use any sampling method or preprocessing technique to alter the class distribution. The proposed method is a data-level algorithm, so the comparison methods are data-level methods. The comparison methods include SMOTE [6], Oversampling (Over), Undersampling (Under), Borderline SMOTE 1 (B-SMOTE 1) [12], Borderline SMOTE 2 (B-SMOTE 2) [12], Tomek link [11], SMOTE Tomek [37], Safe-level SMOTE (S-SMOTE) [13], Gaussian SMOTE (G-SMOTE) [38], NRAS oversampling [14], and MBS [10].

C. Experimental Settings

This work separates the experiments into two parts to make our results reliable and objective. We randomly split the data into training and test sets with ten different random seeds to alleviate the effect caused by the random splitting. Besides data splitting, most data-level methods involve the sampling technique, so we use ten different random seeds to generate data samples to alleviate the stochastic characteristic of sampling methods. Therefore, we repeat each experiment 100 times and use the average of 100 experimental results as the final evaluation result.

Our proposed method and other comparison methods belong to data-level algorithms, so any classifier can be used to assess the classification performance. This article uses linear support vector machine (SVM) [39] and logistic regression as classifiers for two main reasons. First, they are linear methods without using additional feature combinations or kernel tricks, so they are appropriate for assessing the quality of the generated datasets. Second, they are popular classifiers in practical settings.

In experiments, all methods need to set up hyperparameters to determine oversampling or undersampling ratios. We set the ratio of the oversampling and undersampling methods at 100%, which means that the number of minority samples increases to be the same as that of the majority samples for the oversampling approach, while the number of majority samples decreases to the number of minority samples for the undersampling method. We set k as 5 for the methods that are based on the SMOTE algorithm. Regarding the proposed method, the dimension of the latent space is also a hyperparameter, and we use the grid search technique to search for the dimension, in which the search spaces for high- and low-dimensional datasets are {2, 4, 8, 16} and {2, 4, 8}, and the search results are 16 and 2 for high- and low-dimensional datasets, respectively. We use dimension 16 as the threshold to determine whether a dataset is a high- or low-dimensional dataset. The proposed embedding network comprises an encoder and a decoder, both of which comprise a 3-layer neural network. For high-dimensional datasets, the input and output feature sizes for the 3-layer encoder network are (input size, 256), (256, 64), and (64, 16). Regarding the low-dimensional datasets, the input and output feature sizes for the 3-layer encoder network are (input size, 64), (64, 16), and (16, 2). Following the conventional design, the size of the decoder network is in the reverse order of the encoder network. The activation function used by all hidden layers is ReLU, and the activation function for the output layer is a sigmoid function.

D. Evaluation Metric

This work uses the AUC and the area under the precision–recall curve (AUPRC) as evaluation metrics. The AUC can provide an overall model performance when considering all cutoff points on the receiver operating characteristic (ROC) curve. Similarly, the AUPRC can provide an overall model performance by considering precision–recall pairs for different probability thresholds, as precision–recall is a useful measure when the classes are very imbalanced.

E. Experimental Results

The first experiment is conducted with linear SVM as the classifier. The experimental results are listed in Tables II and III, in which the performance result is denoted by the mean and standard deviation of the 100 results.

The experimental results show that the proposed method yields the best performance in most datasets. For the datasets for which the proposed method is not the best, the proposed method still works well. The differences between our proposed method and the best comparison methods among these datasets are minor. It is worth mentioning that our proposed method works very well in an extremely high-dimensional dataset, namely, Isolet.

We conducted the second experiment with logistic regression. Tables IV and V present the results, showing that DDHS outperforms the comparison methods in all datasets. Thus, our proposed method can work well for both linear SVM and logistic regression. The main reason is that once the class distribution is balanced and the data samples are in a separable space, linear classifiers can still work very well.

Subsequently, we use the Wilcoxon rank-sum test to evaluate whether our proposed method is significantly better than

TABLE II
EXPERIMENTAL RESULTS USING LINEAR SVM (AUC)

Dataset	DDHS	SMOTE	Overs	Under	B-SMOTE 1	B-SMOTE 2	Tomek Link
Shuttle	**0.9819±0.0029**	0.9754 ± 0.0043	0.9739 ± 0.0044	0.9730 ± 0.0043	0.9290 ± 0.0475	0.9408 ± 0.0390	0.9715 ± 0.0047
House	**0.8381±0.0107**	0.8138 ± 0.0050	0.8132 ± 0.0050	0.8112 ± 0.0062	0.8169 ± 0.0046	0.8155 ± 0.0050	0.7562 ± 0.0079
Satellite	0.8201±0.0204	0.8168 ± 0.0130	0.8170 ± 0.0133	0.8167 ± 0.0134	0.7801 ± 0.0136	0.7817 ± 0.0141	0.8174 ± 0.0109
Optdigits	**0.9965±0.0059**	0.9842 ± 0.0131	0.9852 ± 0.0139	0.9913 ± 0.0067	0.9818 ± 0.0145	0.9769 ± 0.0187	0.9820 ± 0.0146
Mnist	**0.9645±0.0084**	0.9560 ± 0.0085	0.9567 ± 0.0085	0.9538 ± 0.0093	0.9525 ± 0.0075	0.9495 ± 0.0081	0.9119 ± 0.0206
Sylva	0.9938±0.0030	0.9942 ± 0.0024	**0.9954±0.0017**	0.9927 ± 0.0021	0.9914 ± 0.0034	0.9745 ± 0.0059	0.9909 ± 0.0030*
Isolet	**0.9642±0.0065**	0.9160 ± 0.0120	0.9183 ± 0.0108	0.9437 ± 0.0084	0.9120 ± 0.0149	0.9063 ± 0.0213	0.9125 ± 0.0098

Dataset	DDHS	SMOTE Tomek	S-SMOTE	G-SMOTE	NRAS	MBS	Baseline
Shuttle	**0.9819±0.0029**	0.9754 ± 0.0043	0.9718 ± 0.0044	0.9771 ± 0.0036	0.9791 ± 0.0035	0.9731 ± 0.0046	0.9714 ± 0.0047
House	**0.8381±0.0107**	0.8137 ± 0.0049	0.8057 ± 0.0046	0.7985 ± 0.0058	0.8028 ± 0.0059	0.8064 ± 0.0056	0.7404 ± 0.0063
Satellite	0.8201±0.0204	0.8167 ± 0.0130	0.8185 ± 0.0124	0.8076 ± 0.0128	0.8175 ± 0.0128	**0.8227±0.0112**	0.8169 ± 0.0112
Optdigits	**0.9965±0.0059**	0.9842 ± 0.0131	0.9844 ± 0.0144	0.9945 ± 0.0043	0.9840 ± 0.0121	0.9860 ± 0.0110	0.9820 ± 0.0146
Mnist	**0.9645±0.0084**	0.9560 ± 0.0085	0.9285 ± 0.0171	0.9585 ± 0.0084	0.9470 ± 0.0130	0.9196 ± 0.0170	0.9113 ± 0.0210
Sylva	0.9938 ± 0.0030	0.9942 ± 0.0024	0.9919 ± 0.0033	0.9920 ± 0.0017	0.9928 ± 0.0024*	0.9906 ± 0.0043*	0.9904 ± 0.0043*
Isolet	**0.9642±0.0065**	0.9158 ± 0.0118	0.9103 ± 0.0175	0.9527 ± 0.0080	0.9255 ± 0.0171	0.9178 ± 0.0142	0.9135 ± 0.0108

* No significant difference between the proposed method and the comparison method in performance by using Wilcoxon Rank-sum test

TABLE III
EXPERIMENTAL RESULTS USING LINEAR SVM (AUPRC)

Dataset	DDHS	SMOTE	Overs	Under	B-SMOTE 1	B-SMOTE 2	Tomek Link
Shuttle	**0.9803±0.0036**	0.9766 ± 0.0039	0.9760 ± 0.0039	0.9751 ± 0.0042	0.9362 ± 0.0583	0.9466 ± 0.0451	0.9765 ± 0.0042
House	**0.8347±0.0099**	0.7227 ± 0.0113	0.7204 ± 0.0111	0.7168 ± 0.0122	0.7228 ± 0.0110	0.7220 ± 0.0112	0.7204 ± 0.0113
Satellite	**0.8782±0.0185**	0.8609 ± 0.0149	0.8609 ± 0.0151	0.8602 ± 0.0150	0.8527 ± 0.0139	0.8536 ± 0.0142	0.8596 ± 0.0149
Optdigits	**0.9993±0.0015**	0.9946 ± 0.0080	0.9942 ± 0.0085	0.9935 ± 0.0081	0.9957 ± 0.0064	0.9845 ± 0.0134	0.9967 ± 0.0066
Mnist	**0.9492±0.0219**	0.9304 ± 0.0158	0.9260 ± 0.0168	0.9137 ± 0.0207	0.9224 ± 0.0177	0.9175 ± 0.0175	0.9379 ± 0.0149
Sylva	**0.9808±0.0091**	0.9783 ± 0.0100	0.9771 ± 0.0108	0.9733 ± 0.0118	0.9650 ± 0.0139	0.9055 ± 0.0243	0.9790 ± 0.0108
Isolet	0.9419 ± 0.0000	0.9310 ± 0.0004	0.9288 ± 0.0004	0.8810 ± 0.0173	0.9207 ± 0.0008	0.8476 ± 0.0109	0.9302 ± 0.0001

Dataset	DDHS	SMOTE Tomek	S-SMOTE	G-SMOTE	NRAS	MBS	Baseline
Shuttle	**0.9803±0.0036**	0.9766 ± 0.0039	0.9759 ± 0.0040	0.9763 ± 0.0041	0.9760 ± 0.0039	0.9736 ± 0.0048	0.9765 ± 0.0042
House	**0.8347±0.0099**	0.7226 ± 0.0111	0.7158 ± 0.0114	0.7027 ± 0.0110	0.7183 ± 0.0115	0.7168 ± 0.0114	0.7204 ± 0.0117
Satellite	**0.8782±0.0185**	0.8609 ± 0.0149	0.8603 ± 0.0147	0.8436 ± 0.0151	0.8597 ± 0.0152	0.8492 ± 0.0151	0.8596 ± 0.0148
Optdigits	**0.9993±0.0015**	0.9946 ± 0.0080	0.9963 ± 0.0062	0.9987 ± 0.0028	0.9924 ± 0.0092	0.9960 ± 0.0077	0.9967 ± 0.0066
Mnist	**0.9492±0.0219**	0.9304 ± 0.0157	0.9263 ± 0.0206	0.9267 ± 0.0201	0.9306 ± 0.0213	0.9285 ± 0.0213	0.9379 ± 0.0144
Sylva	**0.9808±0.0091**	0.9783 ± 0.0100	0.9716 ± 0.0137	0.9773 ± 0.0106	0.9803 ± 0.0101	0.9798 ± 0.0090	0.9807 ± 0.0090
Isolet	0.9419 ± 0.0000	0.9310 ± 0.0004	0.8508 ± 0.0076	**0.9451±0.0033**	0.9197 ± 0.0055	0.9277 ± 0.0038	0.9355 ± 0.0001

TABLE IV
EXPERIMENTAL RESULTS USING LOGISTIC REGRESSION (AUC)

Dataset	DDHS	SMOTE	Over	Under	B-SMOTE 1	B-SMOTE 2	Tomek Link
Shuttle	**0.9917±0.0034**	0.9870 ± 0.0025	0.9865 ± 0.0026	0.9818 ± 0.0036	0.9834 ± 0.0052	0.9832 ± 0.0036	0.9816 ± 0.0037
House	**0.9932±0.0031**	0.9909 ± 0.0027	0.9910 ± 0.0026	0.9883 ± 0.0024	0.9895 ± 0.0034	0.9805 ± 0.0062	0.9916 ± 0.0029
Satellite	**0.9149±0.0145**	0.8909 ± 0.0102	0.8907 ± 0.0102	0.8888 ± 0.0102	0.8881 ± 0.0094	0.8886 ± 0.0095	0.8876 ± 0.0103
Optdigits	**1.0000±0.0000**	0.9999 ± 0.0003	0.9998 ± 0.0004	0.9998 ± 0.0004	0.9997 ± 0.0003	0.9993 ± 0.0010	0.9999 ± 0.0002
Mnist	**0.9944±0.0027**	0.9911 ± 0.0031	0.9909 ± 0.0030	0.9897 ± 0.0037	0.9904 ± 0.0025	0.9897 ± 0.0022	0.9907 ± 0.0038
Sylva	**0.9991±0.0004**	0.9991 ± 0.0003*	0.9991 ± 0.0004*	0.9988 ± 0.0004	0.9985 ± 0.0004	0.9958 ± 0.0011	0.9990 ± 0.0004
Isolet	**0.9909±0.0000**	0.9890 ± 0.0001	0.9889 ± 0.0001	0.9874 ± 0.0014	0.9878 ± 0.0001	0.9814 ± 0.0009	0.9892 ± 0.0000

Dataset	DDHS	SMOTE Tomek	S-SMOTE	G-SMOTE	NRAS	MBS	Baseline
Shuttle	**0.9917±0.0034**	0.9870 ± 0.0025	0.9828 ± 0.0037	0.9864 ± 0.0026	0.9871 ± 0.0024	0.9813 ± 0.0039	0.9816 ± 0.0037
House	**0.9932±0.0031**	0.9909 ± 0.0027	0.9864 ± 0.0030	0.9897 ± 0.0032	0.9877 ± 0.0043	0.9910 ± 0.0030	0.9917 ± 0.0029
Satellite	**0.9149±0.0145**	0.8909 ± 0.0102	0.8896 ± 0.0103	0.8806 ± 0.0097	0.8897 ± 0.0103	0.8765 ± 0.0104	0.8875 ± 0.0103
Optdigits	**1.0000±0.0000**	0.9999 ± 0.0003	0.9999 ± 0.0002	0.9999 ± 0.0001	0.9998 ± 0.0004	0.9998 ± 0.0003	0.9999 ± 0.0002
Mnist	**0.9944±0.0027**	0.9911 ± 0.0031	0.9897 ± 0.0039	0.9903 ± 0.0038	0.9898 ± 0.0046	0.9892 ± 0.0044	0.9908 ± 0.0037
Sylva	**0.9991±0.0004**	0.9991 ± 0.0003*	0.9988 ± 0.0004	0.9990 ± 0.0004	0.9991 ± 0.0004	0.9990 ± 0.0004	0.9991 ± 0.0004*
Isolet	**0.9909±0.0000**	0.9890 ± 0.0001	0.9828 ± 0.0007	0.9885 ± 0.0004	0.9871 ± 0.0004	0.9888 ± 0.0002	0.9895 ± 0.0000

* No significant difference between the proposed method and the comparison method in performance by using Wilcoxon Rank-sum test

other comparison methods. We set the significance level α at 0.05, so we would reject the null hypothesis at a confidence level of 95%, concluding whether there is a significant difference in the results between the two groups. Specifically, our proposed DDHS is the reference, and we apply the Wilcoxon rank-sum test to all comparison methods to verify

TABLE V
EXPERIMENTAL RESULTS USING LOGISTIC REGRESSION (AUPRC)

Dataset	DDHS	SMOTE	Overs	Under	B-SMOTE 1	B-SMOTE 2	Tomek Link
Shuttle	**0.9803±0.0035**	0.9753±0.0040	0.9757±0.0040	0.9720±0.0047	0.9335±0.0415	0.9376±0.0270	0.9722±0.0048
House	**0.8347±0.0101**	0.7128±0.0112	0.7107±0.0112	0.6928±0.0121	0.7138±0.0112	0.7132±0.0114	0.7032±0.0113
Satellite	**0.8798±0.0174**	0.8565±0.0150	0.8565±0.0150	0.8540±0.0152	0.8521±0.0141	0.8538±0.0142	0.8548±0.0150
Optdigits	**0.9995±0.0010**	0.9973±0.0050	0.9964±0.0058	0.9964±0.0056	0.9935±0.0069	0.9862±0.0149	0.9978±0.0044
Mnist	**0.9571±0.0165**	0.9307±0.0158	0.9247±0.0169	0.9131±0.0226	0.9207±0.0173	0.9152±0.0172	0.9357±0.0169
Sylva	**0.9809±0.0091**	0.9787±0.0097	0.9787±0.0098	0.9729±0.0127	0.9659±0.0113	0.9262±0.0199	0.9774±0.0109
Isolet	**0.9434±0.0217**	0.9326±0.0140	0.9317±0.0138	0.8923±0.0212	0.9181±0.0199	0.8586±0.0353	0.9385±0.0133

Dataset	DDHS	SMOTE Tomek	S-SMOTE	G-SMOTE	NRAS	MBS	Baseline
Shuttle	**0.9803±0.0035**	0.9753±0.0040	0.9739±0.0048	0.9757±0.0040	0.9747±0.0042	0.9741±0.0048	0.9722±0.0048
House	**0.8347±0.0101**	0.7123±0.0112	0.7045±0.0112	0.6986±0.0110	0.7048±0.0113	0.7081±0.0107	0.7038±0.0115
Satellite	**0.8798±0.0174**	0.8565±0.0150	0.8557±0.0150	0.8455±0.0151	0.8553±0.0152	0.8485±0.0151	0.8549±0.0150
Optdigits	**0.9995±0.0010**	0.9973±0.0050	0.9975±0.0041	0.9986±0.0030	0.9964±0.0058	0.9973±0.0052	0.9978±0.0044
Mnist	**0.9571±0.0165**	0.9307±0.0158	0.9270±0.0203	0.9251±0.0199	0.9329±0.0195	0.9280±0.0197	0.9361±0.0166
Sylva	**0.9809±0.0091**	0.9787±0.0097	0.9735±0.0110	0.9768±0.0111	0.9786±0.0102	0.9761±0.0102	0.9787±0.0099
Isolet	**0.9434±0.0217**	0.9326±0.0140	0.8862±0.0217	0.9250±0.0165	0.9271±0.0163	0.9349±0.0137	0.9397±0.0134

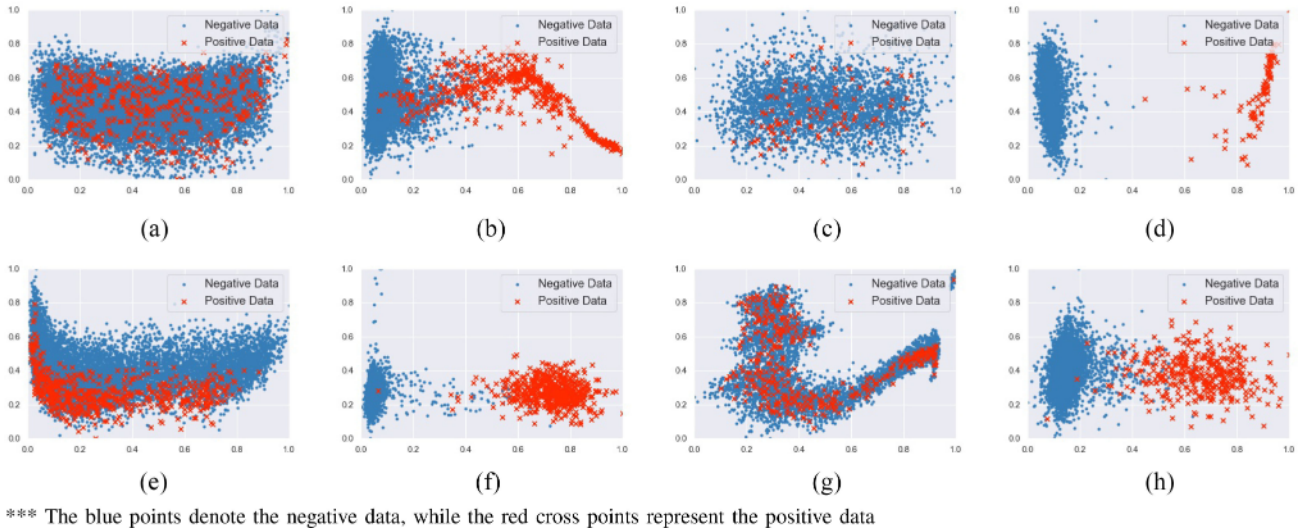


Fig. 5. Effect of embedding network in the four datasets. (a) Letter (Initial). (b) Letter (Final) (c) Optdigits (Initial). (d) Optdigits (Final). (e) Sylva (Initial). (f) Sylva (Final). (g) Isolet (Initial). (h) Isolet (Final).

whether there is a performance difference between DDHS and the comparison method. The results of the Wilcoxon rank sum are presented in Tables II and IV. For the comparison method that has an asterisk, there is no significant difference in performance between it and the proposed DDHS. The results show that the proposed method can significantly outperform other alternatives in most datasets.

We use principal component analysis (PCA) to project the data points into a two-dimensional space, and the purpose is to visualize the data points with scatter plots during model training. To make all datasets have the same range, we transform the data values into the range of 0 and 1 after performing PCA. We select one low-dimensional dataset and three high-dimensional datasets to present the visualization results. It is noted that we use PCA to visualize the data scatter rather than classification.

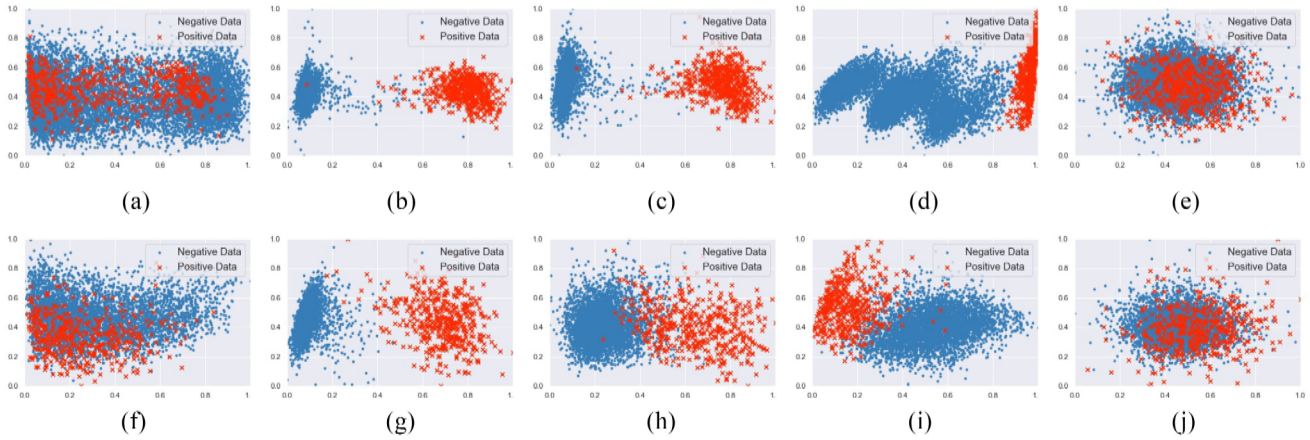
Fig. 5 shows the visualization results before and after the proposed embedding network in the four datasets. Figures show that the two classes overlap initially; however, the proposed embedding network can separate data points in different classes into two classes at the end of the training. The data samples in both the majority and minority classes become denser in the end. We conclude that the proposed embedding network can benefit from the three loss functions to learn a compact and separable latent space.

V. DISCUSSION

Besides the experimental results presented in the previous section, we conduct extensive experiments to discuss and analyze the proposed method.

A. Loss Functions in Embedding Network

The proposed method requires learning a latent space, and we propose an embedding network that involves three loss functions to achieve the goal. Therefore, we conduct experiments with different loss combinations to investigate which



*** The blue points denote the negative data, while the red cross points represent the positive data

Fig. 6. Effects of loss functions on DDHS in (a)–(e) Sylva and (f)–(j) Mnist datasets. (a) and (f) Initial Stage. (b) and (g) $L_{CE} + L_C + L_R$. (c) and (h) $L_{CE} + L_C$. (d) and (i) $L_{CE} + L_R$. (e) and (j) $L_C + L_R$.

TABLE VI
AUCs FOR DIFFERENT LOSS FUNCTION COMBINATIONS

	$(L_{CE}+L_C+L_R)$	$L_{CE}+L_R$	L_C+L_R	$L_{CE}+L_C$
Letter	0.9546±0.0062	**0.9560±0.0002**	0.5000±0.0000	0.9380±0.0109
Magic	**0.7960±0.0048**	0.7797±0.0059	0.4992±0.0009	0.6877±0.0266
Satellite	**0.8466±0.0040**	0.8046±0.0124	0.5444±0.0282	0.8114±0.0092
Mnist	**0.9719±0.0010**	0.9473±0.0011	0.6217±0.0114	0.9482±0.0079
Namao	**0.9542±0.0003**	0.9392±0.0026	0.7614±0.0037	0.9515±0.0005
Isolet	**0.9637±0.0107**	0.9425±0.0137	0.4991±0.0010	0.9505±0.0023

combination leads to the best results. This work presents results from performance and visualization perspectives.

We select six datasets (Letter, Magic, Satellite, Mnist, Nomao, and Isolet) to assess the performance of each combination and use three additional loss combinations to conduct experiments. The first is the proposed method with cross-entropy loss and reconstruction loss ($L_{CE} + L_R$); the second is with center loss and reconstruction loss ($L_C + L_R$); and the last is with cross-entropy loss and center loss ($L_{CE} + L_C$). Notably, the proposed method considers three loss functions ($L_{CE} + L_C + L_R$). Table VI shows the experimental results.

Table VI indicates that the proposed method with all loss functions yields the best performance except the Letter dataset, but the performance difference between the combinations with $L_{CE} + L_C + L_R$ and $L_{CE} + L_R$ is very minor in the Letter dataset. It can be seen from Table VI that L_{CE} is the most essential loss function among the three loss functions, as cross-entropy loss can fully use the label information. The proposed method can also benefit from L_C to improve performance, as it can help the model learn a representation that is close to the class center. The reconstruction loss contributes the least to the performance, which is consistent with the intuition, as the reconstruction loss does not consider the information on the labels. However, the results show that the reconstruction loss can help to improve the performance of the model. We conjecture that the reconstruction loss can pose an additional constraint to force the model to learn a representation that can reconstruct the original input from the latent representation, giving a base to reduce data noise.

Besides performance evaluation, we use scatter plots to visualize the classification results with different loss combinations. Fig. 6(a)–(e) presents the visualization results in the Sylva dataset, while Fig. 6(f)–(j) are the visualization results for the Mnist dataset. It can be seen from the figures that the data points of different classes can be well separated when the loss functions involve L_{CE}, L_C, and L_R. Moreover, the density of the data points in the same class is dense. We conclude that the proposed method can benefit from using L_{CE} and L_C to denote the between-class and within-class losses, since the between-class loss is to ensure that the data points in different classes are separated as far as possible, while the within-class loss is to ensure that the data points of the same class should be close to each other.

The visualization results also show that L_{CE} is the dominant loss among the three loss functions, as the visualization results that involve L_{CE} can separate most data points very well. We can learn a compact representation by using L_C, as the center loss enables the data points to move toward the class center. Therefore, the above experimental results indicate that the proposed method can benefit from using L_{CE} and L_C as between-class and within-class losses.

Considering the three losses enables our proposed embedding network to learn separable and dense feature representations in a low-dimensional latent space. We argue that subsequent synthetic data generation and model training can benefit from the latent space to generate good-quality synthetic samples.

B. Ablation Study of Density-Based Filtering

This work proposes to use density as a criterion to select high-quality data samples. This section performs an ablation study to investigate the influence brought by the proposed DBF on performance, as the ablation study provides a means to evaluate the contribution of a certain component to the entire system.

In the experiments, we conduct experiments by using the proposed DDHS with the DBF process and the proposed DDHS without the DBF process. We select six datasets

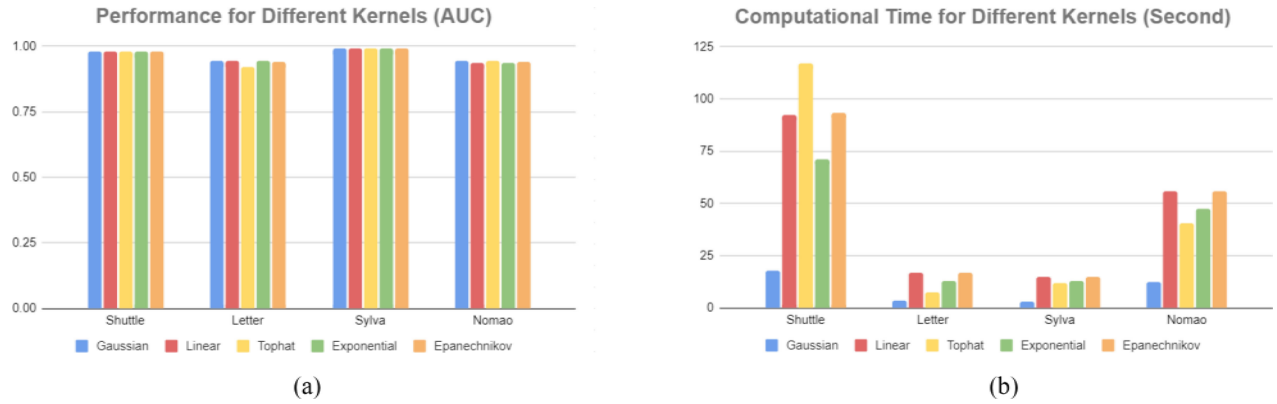


Fig. 7. Performance (AUC) and computational time (second) for different kernels. (a) Performance. (b) Computational Time.

TABLE VII
PERFORMANCE IMPACT OF DBF (AUC)

	DDHS	DDHS w/o DBF
Shuttle	**0.9849±0.0006**	0.9769±0.0017
Letter	**0.9617±0.0018**	0.8726±0.0156
Mnist	**0.9599±0.0025**	0.8976±0.0139
Namao	**0.9528±0.0011**	0.9476±0.0013
Webpage	**0.9449±0.0003**	0.8574±0.0101
Isolet	**0.9562±0.0053**	0.9107±0.0047

TABLE VIII
COMPARISON WITH PCA AND ITS VARIANTS (AUC)

	DDHS	PCA	Kernel PCA	Supervised PCA
Magic	**0.8941±0.0074**	0.8208±0.0086	0.8208±0.0086	0.8208±0.0086
House	**0.9932±0.0031**	0.7797 ±0.0057	0.7797 ±0.0057	0.7797 ±0.0057
Satellite	**0.9149±0.0145**	0.8751±0.0098	0.8751±0.0098	0.8751±0.0098
Spambase	**0.9663±0.0061**	0.7958±0.0180	0.7958±0.0180	0.7958±0.0180
Sylva	**0.9991±0.0004**	0.9955±0.0007	0.9955±0.0007	0.9955±0.00072

(Shuttle, Letter, Mnist, Namao, Webpage, and Isolet) to conduct the experiments, and the experimental results are presented in Table VII. The results demonstrate that the proposed method can benefit from the DBF process, as the DDHS with the DBF process can significantly outperform those without the DBF process. Therefore, the results show that using the DBF process to select data samples can boost the model performance.

C. Comparison With PCA and Its Variants

The proposed work transforms the data points into a lower-dimensional space, and this process is the same as dimensionality reduction. PCA is one of the most well-known methods of dimensionality reduction by projecting data points into a lower-dimensional space that can preserve as much of the data variation as possible. Kernel PCA [40] extends PCA by using the kernel trick, so that the kernel PCA can map data points to a high-dimensional space and perform the linear operations of PCA in the new space. Both PCA and kernel PCA are unsupervised learning methods, meaning that they do not use labels during the dimensionality reduction process. In contrast, supervised PCA [41] generalizes PCA by aiming to deal with supervised learning problems, such as regression and classification, with high-dimensional input data. Unlike PCA which does not consider labels, supervised PCA searches for a subspace in which the dependency between input features and the label is maximized.

We conduct experiments on several datasets to compare the proposed DDHS with PCA and its variants. The experimental results are listed in Table VIII, in which we use linear logistic regression as a classifier. The results show that the proposed method outperforms PCA and its variants. We conclude that the reasons leading to the results are labels and a nonlinear transformation. Both PCA and kernel PCA project data points onto a new space without using label information, so the subsequent classification task cannot always benefit from the new space. Supervised PCA is a supervised learning method that relies on an orthogonal projection matrix to find the subspace, which is a linear transformation. In contrast, the proposed method comprises three loss functions to use label information and data reconstruction to learn a discriminative space. Besides, we use deep neural networks to perform the nonlinear transformation process. Thus, the proposed method can yield better results than the comparison methods in the experiments.

D. Comparison of Kernel Density Estimation

This work uses the Gaussian kernel as the KDE kernel function, but other kernel functions can be used to replace the Gaussian kernel. Therefore, we conduct experiments on four datasets to assess the proposed method when using different kernels, in which the datasets are Shuttle, Letter, Sylva, and Nomao. The comparison kernel functions involve linear, tophat, exponential, and Epanechnikov kernels.

The experiments use performance and computational time as evaluation metrics. The experimental results are listed in Fig. 7. The results in Fig. 7(a) indicate that the average performance of the Gaussian kernel is slightly better than that of other kernels, while the computational time is much better than other alternatives, as shown in Fig. 7(b). Therefore, we can conclude that the Gaussian kernel is more suitable than the other four kernels in the experiments.

TABLE IX
EXPERIMENTAL RESULTS FOR ENSEMBLE METHODS

Dataset	DDHS-boosting	Isolation Forest	RUSBoost	SMOTEBoost	SMOTEBagging	SPE	Decision Tree
Spambase	0.9214±0.0051	0.4950 ± 0.0021	0.9326 ± 0.0084	0.9309 ± 0.0084	0.9302 ± 0.0070	**0.9383±0.0075**	0.9104 ± 0.0119
Optdigits	0.9958±0.0058	0.5844 ± 0.0767	0.9855 ± 0.0136	0.9463 ± 0.0265	0.9414 ± 0.0283	0.9728 ± 0.0148	**0.9997±0.0005**
Mnist	**0.9598±0.0076**	0.6980 ± 0.0279	0.9146 ± 0.0330	0.9156 ± 0.0169	0.9050 ± 0.0203	0.9494 ± 0.0132	0.9026 ± 0.0170
Crime	**0.8295±0.0407**	0.6240 ± 0.0542	0.7021 ± 0.0544	0.6602 ± 0.0395	0.6591 ± 0.0389	0.8201 ± 0.0348	0.6658 ± 0.0405
Sylva	**0.9921±0.0024**	0.5002 ± 0.0009	0.9396 ± 0.0410	0.9680 ± 0.0079	0.9794 ± 0.0074	0.9781 ± 0.0077	0.9592 ± 0.0090
Nomao	0.9390±0.0059	0.5599 ± 0.0073	0.9425 ± 0.0024	0.9512 ± 0.0054	0.9513 ± 0.0024	**0.9579±0.0029**	0.9335 ± 0.0036
Webpage	**0.9405±0.0085**	0.4812 ± 0.0048	0.8119 ± 0.0487	0.8472 ± 0.0177	0.7311 ± 0.0163	0.9162 ± 0.0090	0.7654 ± 0.0000
Isolet	**0.9623±0.0053**	0.4632 ± 0.0095	0.8667 ± 0.0258	0.8119 ± 0.0267	0.8113 ± 0.0237	0.9414 ± 0.0132	0.8031 ± 0.0000

E. Ensemble Learning and Cost-Sensitive Methods

Ensemble learning is an important research topic in algorithm-level methods for dealing with the class imbalance problem. Besides, cost-sensitive learning is another popular technique in algorithm-level methods. The proposed method is a data-level approach, and it is easy to combine ensemble learning to extend our proposed method. This work integrates our proposed DDHS with AdaBoost and devises an algorithm called DDHS-boosting, in which DDHS is invoked in each iteration of the training phase to balance the class distribution before training a classifier.

To assess the performance of DDHS-boosting, we compare DDHS-boosting with five state-of-the-art methods, each of which is an ensemble learning for imbalanced data problems. Comparison methods comprise Isolation Forest [42], RUSBoost [23], SMOTEBoost [22], SMOTEBagging [21], and SPE [24]. As for cost-sensitive learning, we use a cost-sensitive decision tree as the comparison method, in which the model adjusts the weights inversely proportional to the frequency of the class in the input data. It is worth mentioning that SPE is a method that focuses on extremely high-dimensional datasets, so we use the datasets that comprise over 50 features to conduct experiments. The purpose is to assess whether the proposed DDHS-boosting can work well on extremely high-dimensional datasets. The experimental settings are the same as those used in the previous section.

Table IX shows the experimental results, indicating that DDHS-boosting yields promising results. As for these comparison methods, SPE is a recently published work in the top conference, and their experiments showed that SPE can achieve robust performance even under highly overlapping classes and extremely skewed distribution. We empirically show that the proposed DDHS can be easily adapted to the boosting framework and show promising results in high-dimensional datasets.

F. Training Time and Prediction Time

We apply our proposed method to five large-scale datasets, including Isolet, mnist, spambase, optdigits, and crime, and measure the run time of DDHS. The experiments are carried out on a computer with an Intel Xeon CPU E5-2620 2.1 GHZ, 64-GB RAM, and an Nvidia GeForce 1080Ti (11 GB) GPU. For each dataset, we repeat the experiments 20 times and record the training time and prediction time. The average training times for the Isolet, mnist, spambase, optdigits, and crime datasets are 288, 174, 179, 131, and 48 s, respectively. The prediction times for the Isolet, mnist, spambase, optdigits, and crime datasets are 0.016, 0.014, 0.013, 0.009, and 0.011 s, respectively. The results show that our proposed model is efficient and can be applied to large-scale datasets.

VI. CONCLUSION

This article proposes a method called DDHS to cope with the class imbalance problem. The proposed method is a data-level method, and we combine deep learning to develop the proposed method. Central to the proposed method is to use an embedding network to learn a low-dimensional separable latent space. Subsequently, we perform density-based data filtering and use a feature-level method to generate and validate diverse synthetic samples in the latent space. The design of the embedding network is to ensure that the data points in the latent space can preserve the property of class proximity, and we propose to use cross-entropy and center losses to denote between-class and within-class losses. Besides, we use the reconstruction loss as a constraint to enable the learning process to obtain a good feature representation. The experimental results show that the proposed DDHS yields promising results. We also show that it is easy to adapt DDHS to the boosting framework and normally outperform other ensemble learning approaches. The future work will be to extend the proposed method to deal with time-series data, as time-series data are important in many application domains.

ACKNOWLEDGMENT

The authors are grateful to the National Center for High-Performance Computing for computer time and facilities.

REFERENCES

[1] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.
[2] Y.-H. Liu, C.-L. Liu, and S.-M. Tseng, "Deep discriminative features learning and sampling for imbalanced data problem," in *Proc. IEEE Int. Conf. Data Min. (ICDM)*, 2018, pp. 1146–1151.
[3] M. Havaei *et al.*, "Brain tumor segmentation with deep neural networks," *Med. Image Anal.*, vol. 35, pp. 18–31, Jan. 2017.
[4] C.-L. Liu and X.-W. Wu, "Fast recommendation on latent collaborative relations," *Knowl. Based Syst.*, vol. 109, pp. 25–34, Oct. 2016.
[5] C.-L. Liu and Y.-C. Chen, "Background music recommendation based on latent factors and moods," *Knowl. Based Syst.*, vol. 159, pp. 158–170, Nov. 2018.

[6] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jan. 2002.

[7] L. Lusa and R. Blagus, "Evaluation of SMOTE for high-dimensional class-imbalanced microarray data," in *Proc. 11th Int. Conf. Mach. Learn. Appl.*, vol. 2, 2012, pp. 89–94.

[8] Z. Xu, D. Shen, T. Nie, and Y. Kou, "A hybrid sampling algorithm combining M-SMOTE and ENN based on random forest for medical imbalanced data," *J. Biomed. Informat.*, vol. 107, Jul. 2020, Art. no. 103465.

[9] P. Jeatrakul, K. W. Wong, and C. C. Fung, "Classification of imbalanced data by combining the complementary neural network and SMOTE algorithm," in *Neural Information Processing. Models and Applications*, K. W. Wong, B. S. U. Mendis, and A. Bouzerdoum, Eds. Heidelberg, Germany: Springer, 2010, pp. 152–159.

[10] 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.

[11] N. Thai-Nghe, D. Nghi, and L. Schmidt-Thieme, "Learning optimal threshold on resampling data to deal with class imbalance," in *Proc. IEEE RIVF Int. Conf. Comput. Telecommun. Technol.*, 2010, pp. 71–76.

[12] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," in *Advances in Intelligent Computing*, D.-S. Huang, X.-P. Zhang, and G.-B. Huang, Eds. Heidelberg, Germany: Springer, 2005, pp. 878–887.

[13] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, "Safe-level-SMOTE: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem," in *Advances in Knowledge Discovery and Data Mining*, T. Theeramunkong, B. Kijsirikul, N. Cercone, and T.-B. Ho, Eds. Heidelberg, Germany: Springer, 2009, pp. 475–482.

[14] W. A. Rivera, "Noise reduction a priori synthetic over-sampling for class imbalanced data sets," *Inf. Sci.*, vol. 408, pp. 146–161, Oct. 2017.

[15] F. Kamalov, "Kernel density estimation based sampling for imbalanced class distribution," *Inf. Sci.*, vol. 512, pp. 1192–1201, Feb. 2020.

[16] M. Gao, X. Hong, S. Chen, and C. J. Harris, "Probability density function estimation based over-sampling for imbalanced two-class problems," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2012, pp. 1–8.

[17] L. Cao and Y.-K. Zhai, "An over-sampling method based on probability density estimation for imbalanced datasets classification," in *Proc. Int. Conf. Intell. Inf. Process.*, 2016, pp. 1–6.

[18] Y. Sun, M. S. Kamel, A. K. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Pattern Recognit.*, vol. 40, no. 12, pp. 3358–3378, 2007.

[19] X. Gu, F.-L. Chung, H. Ishibuchi, and S. Wang, "Imbalanced TSK fuzzy classifier by cross-class Bayesian fuzzy clustering and imbalance learning," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 8, pp. 2005–2020, Aug. 2017.

[20] Z.-H. Zhou and X.-Y. Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 1, pp. 63–77, Jan. 2006.

[21] S. Wang and X. Yao, "Diversity analysis on imbalanced data sets by using ensemble models," in *Proc. IEEE Symp. Comput. Intell. Data Min.*, 2009, pp. 324–331.

[22] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "SMOTEBoost: Improving prediction of the minority class in boosting," in *Knowledge Discovery in Databases*, N. Lavrač, D. Gamberger, L. Todorovski, and H. Blockeel, Eds. Heidelberg, Germany: Springer, 2003, pp. 107–119.

[23] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "RUSBoost: A hybrid approach to alleviating class imbalance," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 40, no. 1, pp. 185–197, Jan. 2010.

[24] Z. Liu *et al.*, "Self-paced ensemble for highly imbalanced massive data classification," in *Proc. IEEE 36th Int. Conf. Data Eng. (ICDE)*, 2020, pp. 841–852.

[25] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Netw.*, vol. 106, pp. 249–259, Oct. 2018.

[26] G. He, B. Li, H. Wang, and W. Jiang, "Cost-effective active semi-supervised learning on multivariate time series data with crowds," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, Sep. 17, 2020, doi: 10.1109/TSMC.2020.3019531.

[27] G. Wang, K. W. Wong, and J. Lu, "AUC-based extreme learning machines for supervised and semi-supervised imbalanced classification," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 51, no. 12, pp. 7919–7930, Dec. 2021.

[28] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, 2006.

[29] S. H. Khan, M. Hayat, M. Bennamoun, F. A. Sohel, and R. Togneri, "Cost-sensitive learning of deep feature representations from imbalanced data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 8, pp. 3573–3587, Aug. 2018.

[30] 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.

[31] Z. Li, F. Nie, X. Chang, Y. Yang, C. Zhang, and N. Sebe, "Dynamic affinity graph construction for spectral clustering using multiple features," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 12, pp. 6323–6332, Dec. 2018.

[32] Z. Li, F. Nie, X. Chang, L. Nie, H. Zhang, and Y. Yang, "Rank-constrained spectral clustering with flexible embedding," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 12, pp. 6073–6082, Dec. 2018.

[33] Z. Li, L. Yao, X. Chang, K. Zhan, J. Sun, and H. Zhang, "Zero-shot event detection via event-adaptive concept relevance mining," *Pattern Recognit.*, vol. 88, pp. 595–603, Apr. 2019.

[34] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 499–515.

[35] B. W. Silverman, "Using kernel density estimates to investigate multimodality," *J. Royal Stat. Soc. B Methodol.*, vol. 43, no. 1, pp. 97–99, 1981.

[36] D. W. Scott, *Multivariate Density Estimation: Theory, Practice, and Visualization*. Hoboken NJ, USA: Wiley, 2015.

[37] G. E. Batista, A. L. Bazzan, and M. C. Monard, "Balancing training data for automated annotation of keywords: A case study," in *Proc. WOB*, 2003, pp. 10–18.

[38] H. Lee, J. Kim, and S. Kim, "Gaussian-based SMOTE algorithm for solving skewed class distributions," *Int. J. Fuzzy Logic Intell. Syst.*, vol. 17, no. 4, pp. 229–234, 2017.

[39] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[40] S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch, "Kernel PCA and de-noising in feature spaces," in *Proc. NIPS*, vol. 11, 1998, pp. 536–542.

[41] E. Barshan, A. Ghodsi, Z. Azimifar, and M. Z. Jahromi, "Supervised principal component analysis: Visualization, classification and regression on subspaces and submanifolds," *Pattern Recognit.*, vol. 44, no. 7, pp. 1357–1371, 2011.

[42] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. 8th IEEE Int. Conf. Data Min.*, 2008, pp. 413–422.

Chien-Liang Liu (Member, IEEE) received the M.S. and Ph.D. degrees in computer science from the Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, in 2000 and 2005, respectively.

He is currently a Full Professor with the Department of Industrial Engineering and Management, National Chiao Tung University, Hsinchu, Taiwan. His research interests include machine learning, data mining, deep learning, and big data analytics.

Yu-Hua Chang received the M.S. degree in industrial engineering and management from National Yang Ming Chiao Tung University, Hsinchu, Taiwan, in 2021.

His research interests include machine learning, data mining, and deep learning.