AEMLO: AutoEncoder-Guided Multi-label Oversampling

Ao Zhou, Bin Liu(✉), Jin Wang, Kaiwei Sun, and Kelin Liu

Key Laboratory of Data Engineering and Visual Computing, Chongqing University of Posts and Telecommunications, Chongqing, China
{liubin,wangjin,sunkw}@cqupt.edu.cn

Abstract. Class imbalance significantly impacts the performance of multi-label classifiers. Oversampling is one of the most popular approaches, as it augments instances associated with less frequent labels to balance the class distribution. Existing oversampling methods generate feature vectors of synthetic samples through replication or linear interpolation and assign labels through neighborhood information. Linear interpolation typically generates new samples between existing data points, which may result in insufficient diversity of synthesized samples and further lead to the overfitting issue. Deep learning-based methods, such as AutoEncoders, have been proposed to generate more diverse and complex synthetic samples, achieving excellent performance on imbalanced binary or multi-class datasets. In this study, we introduce AEMLO, an AutoEncoder-guided Oversampling technique specifically designed for tackling imbalanced multi-label data. AEMLO is built upon two fundamental components. The first is an encoder-decoder architecture that enables the model to encode input data into a low-dimensional feature space, learn its latent representations, and then reconstruct it back to its original dimension, thus applying to the generation of new data. The second is an objective function tailored to optimize the sampling task for multi-label scenarios. We show that AEMLO outperforms the existing state-of-the-art methods with extensive empirical studies.

Keywords: Multi-label classification · Class imbalance · Oversampling · AutoEncoder

1 Introduction

In the field of multi-label classification (MLC), each instance can belong to multiple labels simultaneously. MLC is widely used in various fields, including image annotation [4], sound processing [16], biology [34] and text classification [14]. The issue of class imbalance in multi-label classification has gained prominence recently [28]. It is prevalent in real-world MLC problems and significantly affects classifier performance, as many algorithms assume data is balanced. Imbalanced datasets tend to bias learners towards majority labels [29].

A. Bifet et al. (Eds.): ECML PKDD 2024, LNAI 14941, pp. 107–124, 2024.
https://doi.org/10.1007/978-3-031-70341-6_7

1.1 Research Goal

Our goal is to address the class imbalance in multi-label datasets through the integration of a deep generative model within an encoder-decoder architecture. This strategy seeks to outperform conventional methods, such as sampling with linear interpolation or random replication, by dynamically creating instances that contain richer feature information.

1.2 Motivation

In recent years, innovative approaches have been developed to tackle the issue of imbalance in multi-label learning [28], including sampling methods [6,17], classifier adaption [9], and ensemble techniques [18,27]. Sampling methods, in particular, aim to balance the dataset before the training phase, offering flexibility and compatibility with any multi-label classifier. To ensure effective sampling, several studies have concentrated on identifying specific samples and refining decision boundaries. For example, MLSOL [17] assigns a higher selecting probability to the sample suffering severe local imbalance. MLBOTE [29] refines the boundary samples related to high imbalance labels and employs different sampling strategies. Traditional oversampling techniques often rely on basic linear interpolation or replication for creating feature vectors of synthesized samples, with label vectors typically generated through majority voting or replication.

The Autoencoder (AE) and Generative Adversarial Network (GAN), as exemplary generative models, have shown substantial potential in data generation, restoration, and augmentation [8,12,15,19,22]. An Autoencoder compresses data into a latent space using an encoder and reconstructs it by a decoder. Its objective is to minimize reconstruction errors, enabling efficient feature extraction and noise reduction [13,15]. Although Autoencoder and GAN are used to address the imbalance problem and generate minority samples, they primarily cater to single-label datasets and face several challenges when applied to multi-label datasets. First, AE and GAN require training samples with identical labels (same class in the single-label dataset or same label set in the multi-label dataset). However, in multi-label data, the number of samples with a complete label set is often too limited to effectively train deep learning models. Secondly, although multi-label datasets can be divided into several binary datasets via the *One vs All* strategy, For each binary dataset, we can learn and reconstruct new feature vectors for multi-label data through end-to-end models, but we can not determine appropriate complete label set for each feature vector (Fig. 1).

1.3 Summary

In this work, we introduce an innovative approach crafted to tackle the class imbalance issue in multi-label datasets named **A**uto**E**ncoder-guided **M**ulti-**L**abel **O**versampling (AEMLO). The core of AEMLO's design lies in two essential elements:

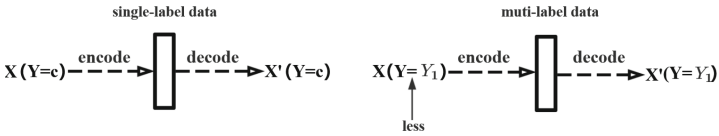


Fig. 1. Using Autoencoders to train data.

1. The basic encoder-decoder architecture is designed to encode data into a lower-dimensional space and subsequently reconstruct it, making it suitable for oversampling applications.
2. A specialized objective function for multi-label imbalance data sampling.

Our approach incorporates the sampling process into a deep encoder-decoder framework that has been pre-trained, providing a holistic solution for the creation of low-dimensional data representations and synthetic instances through an end-to-end methodology. By augmenting the original training set with instances generated via AEMLO, we further train various traditional multi-label classifiers and conduct comparisons against several multi-label sampling techniques. Experimental results consistently demonstrate the superiority of our method. Our code can be found in https://github.com/CquptZA/AEMLO.

2 Related Work

2.1 Multi-label Classification

Formally, let $\mathcal{X} \in \mathbb{R}^d$ represent the d-dimensional feature space, and let $L = \{l_1, l_2, \ldots, l_q\}$ denote a set of q predefined labels. In multi-label classification, our objective is to construct a mapping function $h : \mathcal{X} \rightarrow L$ based on a given multi-label training dataset $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, where each sample $\mathbf{x}_i \in \mathcal{X}$ is associated with a binary label vector $\mathbf{y}_i \in \{0, 1\}^q$. Here, $\mathbf{y}_i$ is a binary vector where each element denotes whether the associated label from L is relevant (1) or not relevant (0) to $\mathbf{x}_i$.

In Multi-Label Classification (MLC), methods are split into three types based on how they handle label correlations. First-order strategies like MLkNN [35] and BR [3] treat labels independently, offering simplicity and efficiency. Second-order methods, such as CLR [11], analyze pairwise label correlations for improved interaction understanding. For complex scenarios with intricate label relationships, high-order methods, like RAkEL [31] and ECC [23], are more effective. RAkEL tackles this by dividing labels into subsets for diverse interaction modeling. ECC sequentially links classifiers, allowing each to learn from the predictions of its predecessors.

2.2 Multi-label Imbalance Learning

Let $N_j^1(N_j^0)$ denote the number of instances with "1" ("0") class of label l_j. $IRlbl$ and ImR are the two measures to evaluate the imbalance level of individual labels

[6,32]. Let $N^1_{max} = max\{N^1_j\}^q_{j=1}$ be the number of "1"s in the most frequent label, $IRlbl_j$ and ImR are defined as:

$$IRlbl_j = N^1_{max}/N^1_j \quad ImR_j = max(N^1_j, N^0_j)/min(N^1_j, N^0_j) \tag{1}$$

The larger the $IRlbl_j$ and ImR_j, the higher the imbalance level of l_j. Then, $MeanIR$ calculates the average imbalance ratio ($IRlbl$) across all labels in a dataset, defined by: $\frac{1}{q}\sum^q_{j=1} IRlbl_j$, where q is the total number of labels. The higher $MeanIR$, the imbalance of the dataset. By considering the $IRlbl$ and $MeanIR$, we can calculate imbalance indicators such as the coefficient of variation of $IRlbl$ ($CVIR$) and concurrency level ($SCUMBLE$) [28].

The imbalanced approaches proposed for MLC can be divided into three categories: sampling methods, classifier adaptation [9,32,33], and ensemble approaches [18,27]. Compared to the other two methods, the sampling method is more universal, as it creates (deletes) instances related to minority (majority) labels to construct a balanced training set that can be used to train any classifier without suffering from bias. Sampling methods involve undersampling and oversampling techniques. Undersampling reduces the presence of majority labels by either randomly removing instances or employing heuristic approaches to selectively eliminate samples. For example, LPRUS and MLRUS [25] aim to alleviate imbalances by respectively targeting the most frequent label sets or individual labels for removal. Conversely, oversampling techniques such as LPROS and the MLSMOTE [6] focus on augmenting the dataset with new instances associated with minority labels, either through duplication or the generation of synthetic samples. Recent developments include the REMEDIAL [5] method, which adjusts label and feature spaces to lessen label co-occurrence and improve sampling. Integrating this method with techniques such as MLSMOTE can further optimize dataset balancing [7]. MLSOL [17] specifically generates instances to focus on local imbalances in datasets. On the other hand, MLTL [21] refines datasets by removing instances that obscure class boundaries, Another notable method, MLBOTE [29], categorizes instances based on their boundary characteristics and applies different sampling rates.

2.3 Deep Sampling Method

Traditional sampling techniques struggle to effectively expand the training set for complex models. This has sparked interest in generative models and their potential to mimic oversampling strategies [2,10]. Utilizing an encoder-decoder setup, artificial instances can be effectively introduced into an embedding space. AE [13,15] and GAN [12] have been effectively employed to capture the underlying distribution of data and further applied to generate data for oversampling purposes. AE is designed to learn efficient data codings in an unsupervised manner. Essentially, they aim to capture the most salient features of the data by compressing the input into a lower-dimensional latent space and then reconstructing it back to the original dimensionality. The core objective of an AE is defined by the reconstruction error, which quantifies the difference between

the original data and its reconstruction. Unlike Variational Autoencoder (VAE), which incorporates the Kullback-Leibler (KL) divergence to regulate the latent space, standard AE relies solely on the reconstruction loss. This encourages the model to develop a compressed representation that retains as much of the original information as possible, enabling the AE to generate reconstructions that are as close to the input. For example. DeepSMOTE [8] integrates traditional SMOTE methods into encoding and decoding architectures similar to AE. VAE strive to maximize a variational lower bound on the data's log-likelihood. Typically, they are formulated by merging a reconstruction loss with the KL divergence. The KL divergence serves as an indirect penalty for the reconstruction loss, steering the model towards a more faithful replication of the data distribution [22]. By penalizing the reconstruction loss, the model is motivated to refine its replication of the data, thereby enabling it to produce outputs rooted in the input's latent distribution. GAN has significantly advanced the field of computer vision by framing image generation as a competitive game between a generator and a discriminator network [19,20,37]. Despite their remarkable achievements, GAN requires the deployment of two separate networks, can encounter training difficulties, and are susceptible to mode collapse [8].

3 Multi-label AutoEncoder Oversampling

3.1 Method Description and Overview

The multi-label AutoEncoder oversampling framework, as described in Algorithm 1, is divided into the training process and the instance generation phase.

In the training process, as shown in Fig. 2, the model is designed to learn and optimize four distinct mapping functions: the feature encoding function $\mathbf{F}_{ex}$, label encoding function $\mathbf{F}_{ey}$, feature decoding function $\mathbf{F}_{dx}$, and label decoding function $\mathbf{F}_{dy}$. The model is trained end-to-end with mini-batches and the Adam optimizer, where batch size n encompasses the feature vector $\mathbf{x}_i$ and binary label vector $\mathbf{y}_i$ of the i-th sample, respectively. The matrices $\mathbf{X}$ and $\mathbf{Y}$ aggregate the input features and labels for all samples in the batch. The framework ingests a feature matrix $\mathbf{X}$ and its corresponding label matrix $\mathbf{Y}$, aiming to output reconstructed versions of $\mathbf{X}'$ and $\mathbf{Y}'$. Meanwhile, The other goal of our model is to identify an optimal latent space $\mathcal{L}$, where the Deep Canonical Correlation Analysis (DCCA) component [1] enhances the correlation between $\mathbf{X}$ and $\mathbf{Y}$. Therefore, the model's objective function is defined as:

$$\Theta = \min_{\mathbf{F}_{ex},\mathbf{F}_{ey},\mathbf{F}_{dx},\mathbf{F}_{dy}} \Phi(\mathbf{F}_{ex},\mathbf{F}_{ey}) + \alpha\Psi(\mathbf{F}_{ex},\mathbf{F}_{dx}) + \beta\Gamma(\mathbf{F}_{ey},\mathbf{F}_{dy}) \tag{2}$$

where $\Phi(\mathbf{F}_{ex},\mathbf{F}_{ey})$ denotes the latent space loss, $\alpha\Psi(\mathbf{F}_{ex},\mathbf{F}_{dx})$ and $\beta\Gamma(\mathbf{F}_{ey},\mathbf{F}_{dy})$ signify the reconstruction losses. Here, α and β serve to balance these components, respectively. In Sect. 3.2, we will explain every term of the objective function in details. At the end of each epoch, we enter a validation phase, adjusting the threshold for binary label conversion by maximizing the F-measure of each label on the validation set.

Algorithm 1: Using Encoder and Decoder for Multi-Label Sampling

Input: Original dataset $D = (\mathbf{X}, \mathbf{Y})$, sample count num, parameter α, β, latent space dimension l

Output: Balanced training set D'

```
/* train the Encoder and Decoder                                    */
for e ← 1 to epochs do
    for batch(X̂, Ŷ) ← B do
        F_ex((X̂)), F_ey((Ŷ));            /* encode batch data to ℒ */
        F_dx((X̂)), F_dy((Ŷ));          /* decode batch data from ℒ */
        Define the loss function by Eq 2;
        Compute gradients and update parameters with Adam;
    Update T ;          /* validate and optimize the bipartition
    threshold for each label */
/* generate instances                                               */
while num > 0 do
    x_s ← select form M;                  /* choose seed instance */
    F_ex(x_s) ;                                          /* encode */
    x_g ← F_dx(F_ex(x_s))   y_g ← F_dy(F_ex(x_s));       /* decode */
    y_g ← T;                                          /* rounding */
    D'=D' ∪ (x_g, y_g) ;
    num ← num − 1 ;
return D'
```

After the model training is completed, we proceed with instance generation. Let num represent the required number of instances to be generated, and p denote the sampling rate. Further details on the sampling process can be found in Sect. 3.3.

3.2 Loss Function

Joint Embedding. To calculate $\Phi(\mathbf{F}_{ex}, \mathbf{F}_{ey})$ defined in Eq. 2, we employ the DCCA to embed features and labels into a shared latent space simultaneously and rewrite the correlation-based $\Phi(\mathbf{F}_{ex}, \mathbf{F}_{ey})$ as the following deep version:

$$\Phi(\mathbf{F}_{ex}(\mathbf{X}), \mathbf{F}_{ey}(\mathbf{Y})) = \|\mathbf{F}_{ex}(\mathbf{X}) - \mathbf{F}_{ey}(\mathbf{Y})\|_F^2 = \mathrm{Tr}(\mathbf{C}_1^T \mathbf{C}_1) + \lambda \mathrm{Tr}(\mathbf{C}_2^T \mathbf{C}_2 + \mathbf{C}_3^T \mathbf{C}_3) \tag{3}$$

where

$$\begin{aligned} \mathbf{C}_1 &= \mathbf{F}_{ex}(\mathbf{X}) - \mathbf{F}_{ey}(\mathbf{Y}), \\ \mathbf{C}_2 &= \mathbf{F}_{ex}(\mathbf{X})\mathbf{F}_{ex}(\mathbf{X})^T - \mathbf{I}, \\ \mathbf{C}_3 &= \mathbf{F}_{ey}(\mathbf{Y})\mathbf{F}_{ey}(\mathbf{Y})^T - \mathbf{I}, \\ constraint &: \mathbf{F}_{ex}(\mathbf{X})\mathbf{F}_{ex}(\mathbf{X})^T = \mathbf{F}_{ey}(\mathbf{Y})\mathbf{F}_{ey}(\mathbf{Y})^T = \mathbf{I} \end{aligned} \tag{4}$$

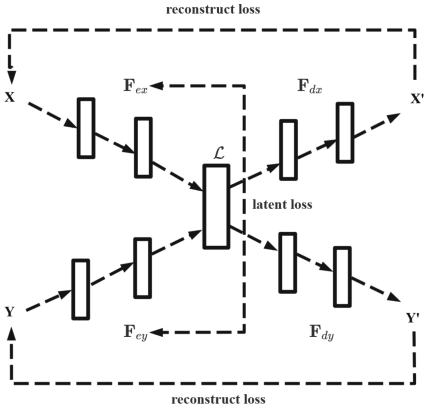


Fig. 2. The architecture of the proposed autoencoder learns the latent space $\mathcal{L}$ through the function of $\mathbf{F}_{ex}$ and $\mathbf{F}_{ey}$, and decouples $\mathcal{L}$ through $\mathbf{F}_{dx}$ and $\mathbf{F}_{dy}$.

$\mathbf{C}_1$ quantifies the discrepancy between the feature and label embeddings, while $\mathbf{C}_2$ and $\mathbf{C}_3$ assess how each embedded space diverges from orthonormality. The goal is to minimize these discrepancies to align the embeddings of $\mathbf{X}$ and $\mathbf{Y}$ closely, ensuring they remain orthonormal as dictated by the constraint. The identity matrix $\mathbf{I} \in \mathbb{R}^{l \times l}$, serves as a benchmark for achieving this orthonormality, where l denotes the latent space dimension. Integrating DCCA in our sampling framework not only enables a unified embedding of features and labels but also allows for their precise reconstruction from shared space through the functions $\Psi(\mathbf{F}_{ex}(\mathbf{X}), \mathbf{F}_{dx}(\mathbf{X}))$ for features and $\Gamma(\mathbf{F}_{ey}(\mathbf{Y}), \mathbf{F}_{dy}(\mathbf{Y}))$ for labels.

Feature Reconstruction. The function Ψ is composed of two distinct components: the feature reconstruction error, $\mathcal{M}$, and the instance similarity metric, $\mathcal{S}$. It is defined as:

$$\Psi(\mathbf{F}_{ex}(\mathbf{X}), \mathbf{F}_{dx}(\mathbf{X})) = \mathcal{M} + \lambda\mathcal{S} \tag{5}$$

where λ is a regularization parameter that balances the contribution of the similarity metric $\mathcal{S}$ relative to the reconstruction error $\mathcal{M}$.

The reconstruction error $\mathcal{M}$, quantified as mean squared error, is calculated as:

$$\mathcal{M} = \sum_{i=1}^{n} (\mathbf{x}'_i - \mathbf{x}_i)^2 \tag{6}$$

with $\mathbf{x}'_i$ representing the reconstruction of the input $\mathbf{x}_i$, generated by the $\mathbf{F}_{dx}$ applied to the encoded representation $\mathbf{F}_{ex}(\mathbf{x}_i)$.

The similarity metric $\mathcal{S}$ ensures that the proximity between original instances is maintained after reconstruction, thereby conserving the integrity of the feature space. This metric is formulated as:

$$\mathcal{S} = \frac{1}{n(n-1)} \sum_{i,j=1, i \neq j}^{n} \left((\mathbf{x}_i - \mathbf{x}_j)^2 - (\mathbf{x}'_i - \mathbf{x}'_j)^2 \right)^2 \tag{7}$$

which measures the squared differences in distances between all pairs of original instances $(\mathbf{x}_i, \mathbf{x}_j)$ and their corresponding reconstructed pairs $(\mathbf{x}'_i, \mathbf{x}'_j)$, ensuring the model preserves instance similarity in its learned feature space. The combination of $\mathcal{M}$ and $\mathcal{S}$ ensures an optimal balance between high-fidelity feature reconstruction and the preservation of relative distances among data within the feature space.

Label Reconstruction. The function Γ encapsulates ranking loss to help the model retrieve label vectors from the shared embedding space:

$$\Gamma(\mathbf{F}_{ey}(\mathbf{Y}), \mathbf{F}_{dy}(\mathbf{Y})) = \sum_{i=1}^{n} \left(\frac{E_i}{|Y_i| \times |\bar{Y}_i|} \right) \tag{8}$$

where E_i is defined as the set of label pairs (y_{ij}, y_{ik}) that satisfy the condition $f(\mathbf{x}_i, y_{ij}) \leq f(\mathbf{x}_i, y'_{ij})$, with these label pairs belonging to the Cartesian product of the set of positive labels Y_i and the set of negative labels $\bar{Y}_i$. Here, Y_i represents the set of positive labels, while $\bar{Y}_i$ represents the set of negative labels.

3.3 Generate Instances and Post-processing

Let $L_s = \{l_j \mid ImR_j > 10, IRlbl_j > MeanIR\}$[1] be the set comprising m minority labels [29] and $M = \{(\mathbf{x}_i, \mathbf{y}_i) \mid y_{ij} = 1, l_j \in L_s\}$ be the minority instance set associated any labels in L_s. Then, we randomly select a seed sample $(\mathbf{x}_i, \mathbf{y}_i)$ from M to initiate the sampling process through forward inference. As shown in Fig. 3, the process encodes the feature vector $\mathbf{x}_s$ into a latent space by $\mathbf{F}_{ex}(\mathbf{x}_s)$, then decodes it to the feature and label vectors of the new instance by $\mathbf{F}_{dx}(\mathbf{F}_{ex}(\mathbf{x}_s))$ and $\mathbf{F}_{dy}(\mathbf{F}_{ex}(\mathbf{x}_s))$, respectively. Specifically, we employ a predefined threshold set T to transform the decoded numerical label vector into a binary label vector. After the process, we remove any instances where the generated label vector is entirely zeros to ensure each instance contributes meaningfully to the dataset.

4 Experiments and Analysis

4.1 Datasets

We evaluate our proposed model across 9 benchmark multi-label datasets spanning diverse domains, such as text, images, and bioinformatics [30]. Each dataset is characterized by a set of statistics and imbalance metrics, which include the

[1] Here, 10 is a hyperparameter. We refer to the suggestions in [29] for the selection.

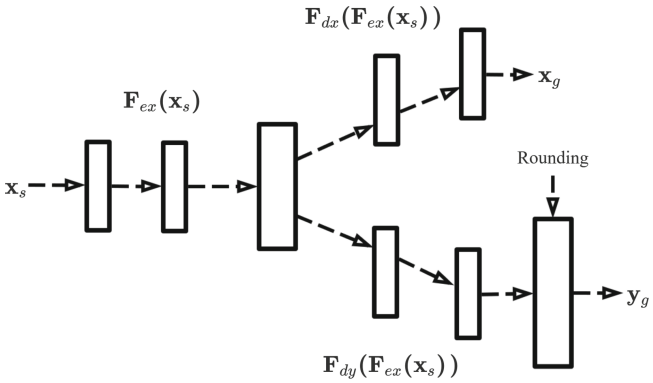


Fig. 3. The process of generating instances

number of instances (n), feature dimensions (d), labels (q), average label cardinality per instance $Card(q)$, and label density $Den(q)$. A comprehensive explanation of these statistical measures and imbalance metrics is available in [28,36]. In the experiment, 20% training data is split as the validation set, which is used to establish thresholds for accurate prediction of final labels (Table 1).

Table 1. Characteristics of the experimental datasets

Dataset	Domain	n	d	q	$Card(q)$	$Den(q)$	MeanIR	CVIR
bibtex	text	7395	1836	159	2.40	0.02	12.50	0.41
enron	text	1702	1001	53	3.38	0.06	73.95	1.96
Languagelog	text	1460	1004	75	15.93	0.21	5.39	0.78
yeast	biology	2417	103	14	4.24	0.30	7.20	1.88
rcv1	text	6000	472	101	2.88	0.03	54.49	2.08
rcv2	text	6000	472	101	2.63	0.03	45.51	1.71
rcv3	text	6000	472	101	2.61	0.03	68.33	1.58
cal500	music	502	68	174	26.04	0.15	20.58	1.09
Corel5k	images	5000	499	374	3.52	0.01	189.57	1.53

4.2 Experiment Setup

In AEMLO, $\mathbf{F}_{ex}$ and $\mathbf{F}_{ey}$ are comprised of two fully connected layers, whereas $\mathbf{F}_{dx}$ and $\mathbf{F}_{dy}$ adopt a single fully connected layer structure. Each layer within these components incorporates 512 neurons and incorporates a leaky ReLU activation function to introduce nonlinearity. The parameters α and β of objective function are explored within the range of $[2^{-4}, 2^{-3}, \cdots, 2^{4}]$.

Compared Sampling Methods. We compare our proposed sampling method with the following sampling methods.

- **MLSMOTE**: MLSMOTE extends the classical SMOTE method to multi-label data.[$Parameter : Ranking, k = 5$]
- **MLSOL**: MLSOL considers local label imbalance and employs weight vectors and type matrices for seed instance selection and synthetic instance generation. [$Parameter : p \in (0.1, 0.3, 0.5, 0.7, 0.9)\,, k = 5$]
- **MLROS**: MLROS executes replicating instances associated with minority labels. [$Parameter : p \in (0.1, 0.3, 0.5, 0.7, 0.9)$]
- **MLRUS**: MLRUS executes removing instances associated with majority labels. [$Parameter : p \in (0.1, 0.2, 0.3)$]
- **MLTL**: MLTL identifies and removes Tomek-Links in multi-label data by considering the set of instances associated with each minority label. [$Parameter : k = 5$]
- **MLBOTE**: MLBOTE divides instances into three categories, and determines specific instance weights and sampling rates for each group.

Base Multi-lable Classifiers. We use all sampling methods on the following five multi-label classifiers.

- **Binary Relevance** [3]: BR transforms the multi-label classification problem into multiple independent binary classification tasks, each of which corresponds to one label and trains a binary classifier. Base binary classifier: SVM.
- **Multi-label k-Nearest Neighbors** [35]: MLkNN is an extension of the k-Nearest Neighbors (kNNs) algorithm for multi-label classification. hyperparameter configuration: k=10.
- **Random k-labELsets** [31]: RAkEL divides the entire label set into several random subsets containing at least three labels and encodes each subset as a multi-class dataset by treating each label combination as a class. hyperparameter configuration: k = 3, n = 2q, base binary classifier: C4.5 Decision Tree.
- **Ensemble of Classifier Chain** [23]: ECC is an approach that extends the Classifier Chain further in an ensemble framework. hyperparameter configuration: N = 5, base binary classifier: C4.5 Decision Tree.
- **Calibrated Label Ranking** [11]: CLR transforms the multi-label learning problem into the label ranking problem. Base binary classifier: SVM

Evaluation Metrics. To assess the efficacy of the batch method in multi-label classification, three commonly utilized evaluation metrics are adopted, comprising Macro-F, Macro-AUC, Ranking Loss. Please refer to [36] for detailed definitions of these metrics (Figs. 4, 5 and 6).

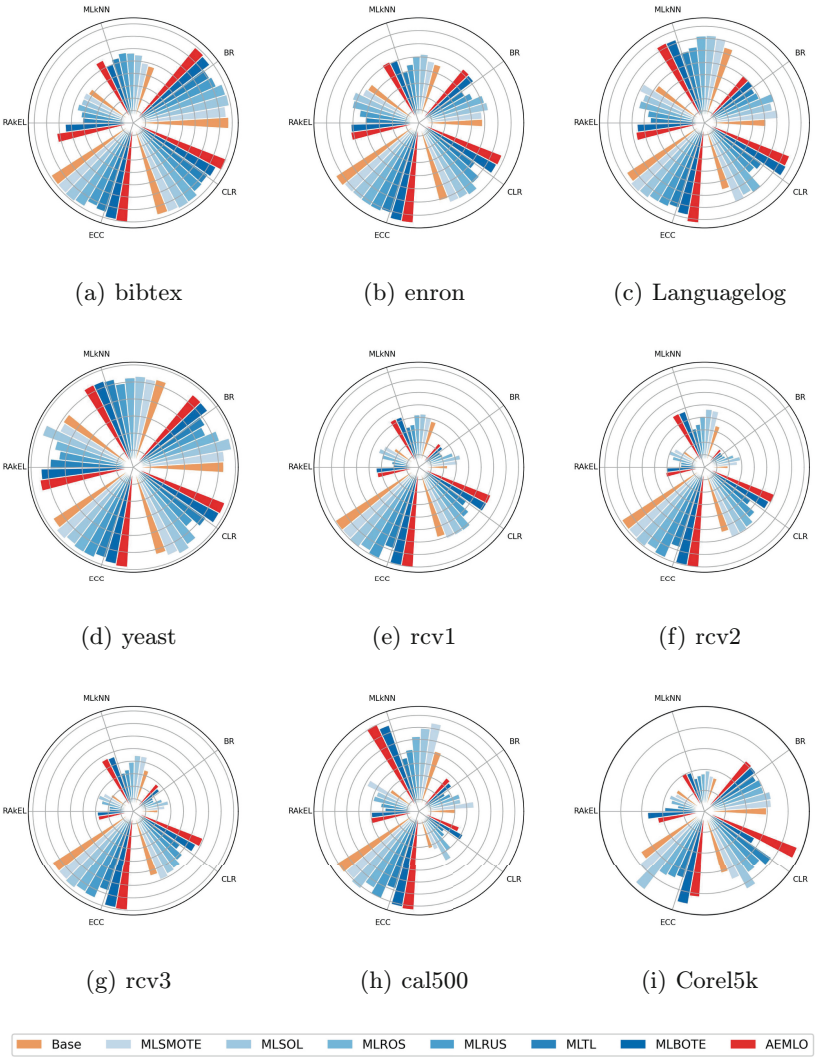


Fig. 4. The performance of the multi-label sampling methods in terms of Macro-F across five different classification methods.

4.3 Experimental Analysis

Table 2 presents the average rankings of each base classifier combined with sampling methods across all datasets. Additionally, the Friedman test was utilized to verify the significant superiority/inferiority of our method compared to other sampling approaches across three evaluation metrics in five basic multi-label classification methods. The detailed results of the comparative sampling methods using five fundamental learners on the Macro-F, Macro-AUC, and Ranking Loss

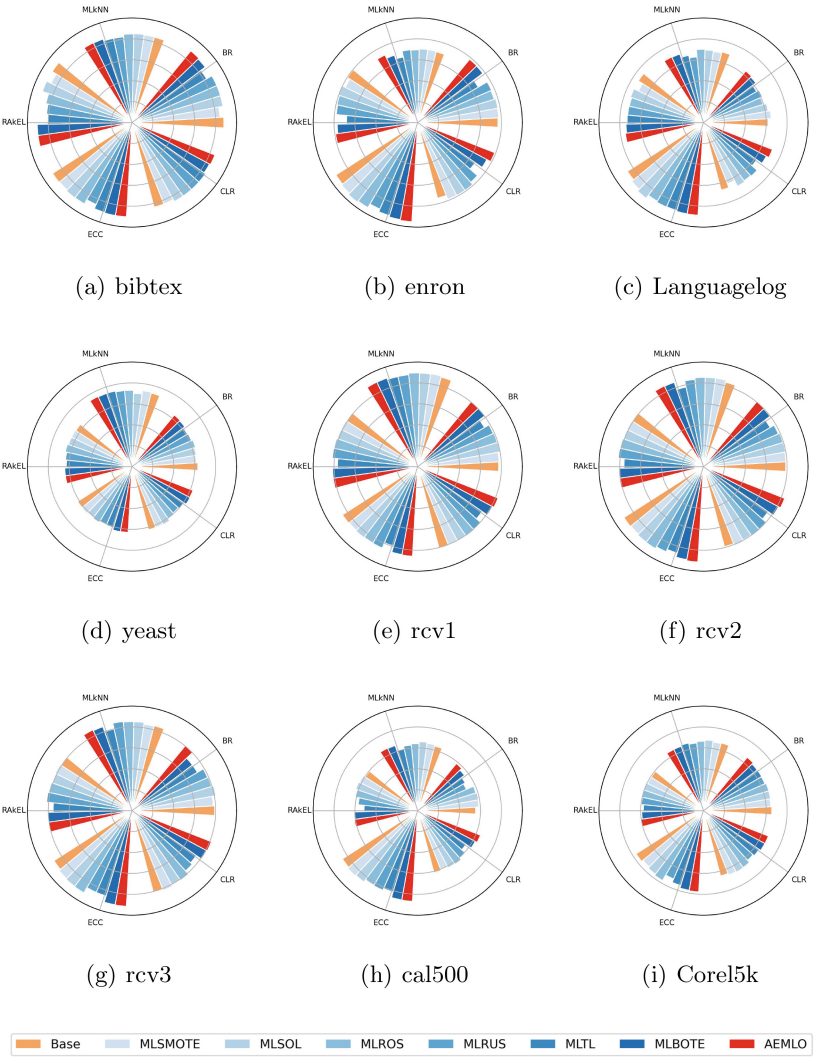


Fig. 5. The performance of the multi-label sampling methods in terms of Macro-AUC across five different classification methods.

are shown in Github. The Origin represents training directly using the training set without any sampling methods. The results indicate that the AEMLO method achieves the highest average ranking in almost all metrics, securing the most significant victories without any substantial losses. It is observed that MLBOTE and MLSOL outperform MLSMOTE, reflecting that refining rule selection for seed instances is more effective than oversampling with all minority seeds directly. An interesting observation is that the performance of MLTL and MLRUS is even worse than that of the original dataset. This is primarily

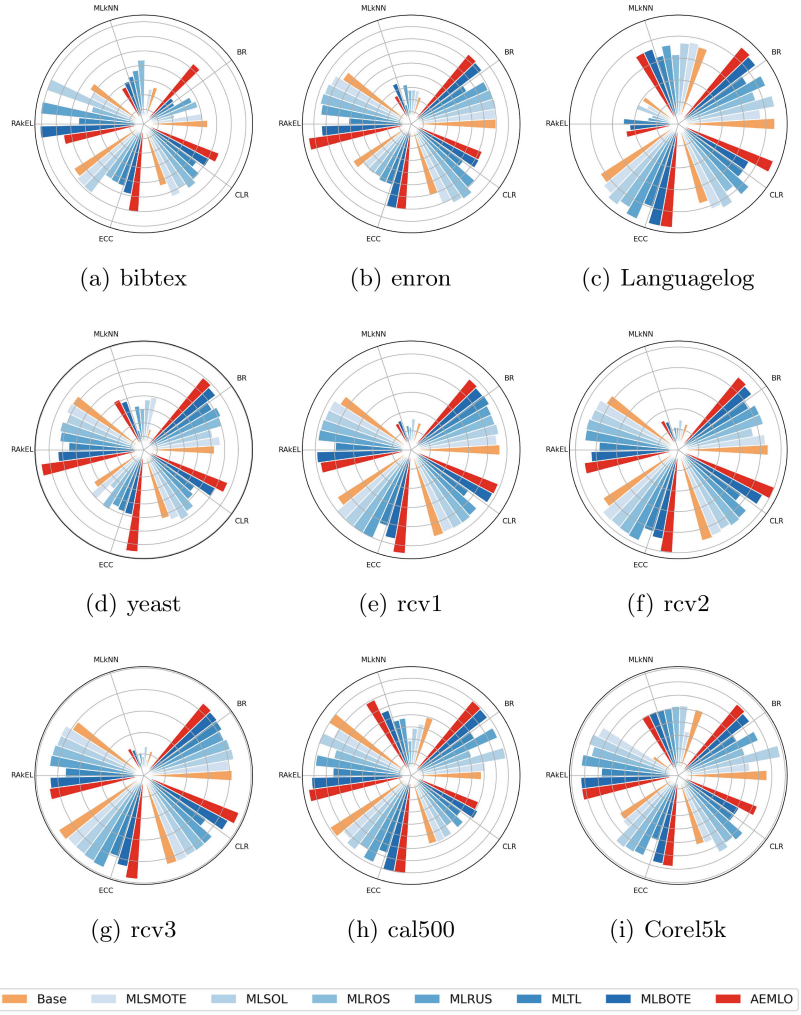


Fig. 6. The performance of the multi-label sampling methods in terms of Ranking Loss across five different classification methods.

attributed to the removal of critical instances, leading to the loss of important information.

Autoencoders excel at generating new samples by learning compressed representations of input data. However, a subtle challenge arises when the feature space of these generated samples diverges from that of the origin samples. This divergence may pose difficulties for the MLkNN, which relies heavily on the distances between samples within the feature space to identify nearest neighbors. As such, any significant discrepancy in the feature distribution between generated and origin samples could potentially impact the MLkNN to accurately

Table 2. The mean ranking of different sampling methods evaluated with five base learners across three metrics is presented. The notation (n1/n2) indicates adjustments from the Friedman test at a 5% significance level, signifying that the method in question significantly outperforms n1 methods and is outdone by n2 methods. The top-performing method is emphasized in bold, with lower rankings indicating superior performance.

		Origin	MLSMOTE	MLSOL	MLROS	MLRUS	MLTL	MLBOTE	AEMLO
Macro-F	BR	6.33(0/3)	5.11(0/3)	2.89(4/1)	4.22(1/0)	6.00(0/3)	7.33(0/5)	2.78(4/0)	**1.33(6/0)**
	MLkNN	5.33(0/4)	3.11(3/0)	**1.56(4/0)**	3.89(2/0)	6.89(0/4)	7.67(0/5)	4.56(1/1)	3.00(3/0)
	RAkEL	5.67(0/3)	4.11(2/2)	2.67(2/0)	4.89(2/2)	7.22(0/5)	7.00(0/5)	2.44(3/0)	**2.00(4/0)**
	ECC	5.11(0/2)	4.22(0/0)	3.56(2/0)	5.67(0/2)	5.44(0/2)	6.67(0/3)	2.78(3/0)	**2.56(4/0)**
	CLR	6.33(0/3)	3.89(2/0)	2.78(3/0)	3.00(2/0)	6.89(0/4)	7.11(0/5)	3.89(2/0)	**2.11(3/0)**
	Avg(Total)	5.75(0/15)	4.09(7/5)	2.69(15/1)	4.33(7/4)	6.49(0/18)	7.16(0/23)	3.29(13/1)	**2.20(20/0)**
Macro-AUC	BR	5.00(0/2)	4.78(0/1)	3.00(3/0)	4.11(1/0)	5.78(0/3)	7.56(0/5)	4.33(1/1)	**1.44(7/0)**
	MLkNN	5.11(0/3)	4.22(2/0)	3.44(3/0)	3.78(2/0)	6.11(0/4)	6.78(0/4)	4.44(1/0)	**2.11(3/0)**
	RAkEL	5.33(0/2)	4.33(1/0)	**2.78(2/0)**	3.56(1/0)	4.33(1/0)	7.67(0/6)	4.89(0/0)	3.11(2/0)
	ECC	5.22(0/2)	5.02(0/2)	3.33(3/0)	4.22(1/0)	6.89(0/4)	7.00(0/4)	2.56(3/0)	**1.56(5/0)**
	CLR	6.00(0/3)	5.00(0/1)	3.16(2/0)	3.67(2/1)	5.56(0/2)	7.44(0/4)	3.33(2/0)	**1.67(5/0)**
	Avg(Total)	5.33(0/12)	4.71(3/4)	2.98(13/0)	3.87(7/0)	5.73(1/13)	7.29(0/23)	3.91(7/1)	**2.18(22/0)**
Ranking Loss	BR	4.89(0/1)	6.22(0/2)	4.44(0/0)	4.00(1/0)	5.33(0/2)	6.11(0/3)	3.00(3/0)	**2.00(4/0)**
	MLkNN	4.56(0/2)	5.44(0/3)	3.56(3/0)	5.33(0/2)	4.00(0/2)	6.89(0/5)	**2.89(3/0)**	3.33(2/0)
	RAkEL	6.44(0/3)	5.22(0/1)	4.33(1/0)	4.44(0/0)	2.78(2/0)	7.56(0/4)	3.11(2/0)	**2.11(3/0)**
	ECC	5.00(0/1)	4.56(0/0)	4.78(0/0)	3.67(1/0)	4.67(0/0)	6.33(0/2)	4.11(0/0)	**2.89(2/0)**
	CLR	5.22(0/2)	4.00(1/1)	4.22(1/1)	4.11(1/0)	5.00(0/2)	7.67(0/5)	3.89(3/0)	**1.89(5/0)**
	Avg(Total)	5.22(0/9)	5.09(1/7)	4.27(5/1)	4.31(3/2)	4.36(2/6)	6.91(0/19)	3.40(11/0)	**2.44(16/0)**

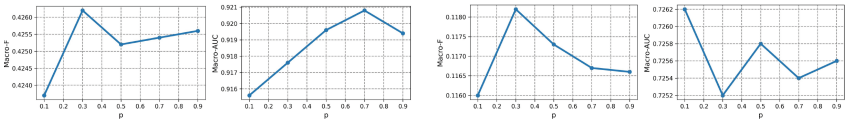
classify unseen samples. The enhanced performance of BR and CLR methods on augmented datasets can be attributed to the robustness of SVM and its adeptness at navigating complex decision boundaries. Specifically, SVM is particularly effective at managing the intricacies introduced into the feature space by data synthesized through Autoencoders.

4.4 Parameter Analysis

We investigate the influence of various parameter settings on the performance of ALMLO. We select smaller enron and larger Corel5k as two representative datasets in the parameter analysis.

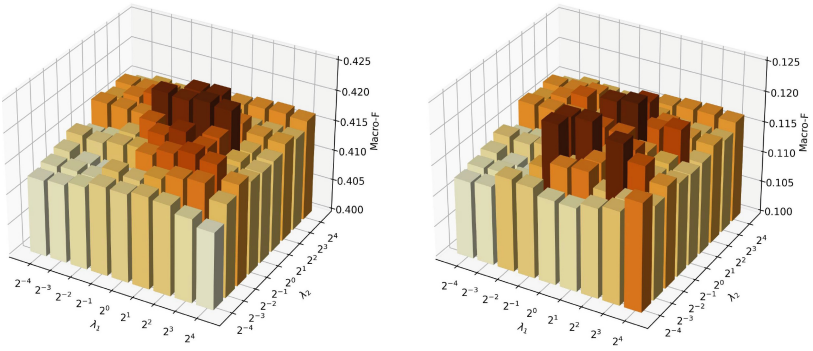
As shown in Fig. 7(a), the impact of varying sampling rate p on Macro-F and Macro-AUC scores (based on MLkNN) shows a trend of initial fluctuation, followed by stabilization. In contrast, in Fig. 7(b) exhibits a higher sensitivity to p, with significant volatility in Macro-F and inconsistent variations in Macro-AUC. These observations suggest that the optimal selection of p may be highly dependent on dataset characteristics.

Figure 8 illustrates ALMLO's performance sensitivity to variations in α and β, highlighting the importance of balancing feature reconstruction loss with label relevance loss during optimization.



(a) enron (b) Corel5k

Fig. 7. Performance of sampling rate in terms of Macro-F and Macro-AUC.

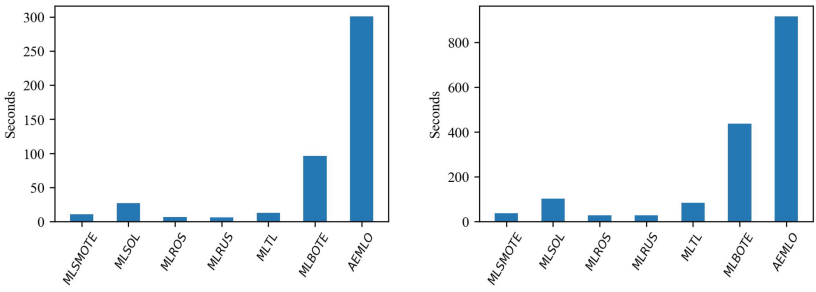


(a) enron (b) Corel5k

Fig. 8. Performance of AEMLO with varying parameter configurations in terms of Macro-F.

4.5 Sampling Time

Figure 9 shows the time efficiency of different sampling methods, with the epoch set as 100 for AEMLO. It is evident that AEMLO, as a deep learning approach, requires training before sampling, resulting in a higher time expenditure.



(a) enron (b) Corel5k

Fig. 9. Sampling time of different sampling method.

5 Conclusion

In this paper, we introduce AEMLO, an innovative oversampling model devised for addressing data imbalance in multi-label learning by integrating canonical correlation analysis with the encoder-decoder paradigm. AEMLO emerges as an effective oversampling solution for training deep architectures on imbalanced data distributions. It acts as a data-level solution for class imbalance, synthesizing instances to balance the training set and thus enabling the training of any classifiers without bias. AEMLO exhibits the pivotal characteristics crucial for a successful sampling algorithm in the multi-label learning domain: the ability to manipulate features and labels, i.e., to learn low-dimensional joint embeddings from feature and label representations and transform them into an original-dimensional space, along with generating new feature representations and their corresponding label subsets. This is facilitated through the utilization of an encoder/decoder framework. Extensive experimental studies demonstrate the capability of AEMLO to handle imbalanced multi-label datasets in various domains and collaborate with diverse multi-label classifiers.

Acknowledgements. This work was supported by the National Natural Science Foundation of China (62302074) and the Science and Technology Research Program of Chongqing Municipal Education Commission (KJQN202300631).

References

1. Andrew, G., Arora, R., Bilmes, J., Livescu, K.: Deep canonical correlation analysis. In: International Conference on Machine Learning, pp. 1247–1255. PMLR (2013)
2. Bellinger, C., Drummond, C., Japkowicz, N.: Manifold-based synthetic oversampling with manifold conformance estimation. Mach. Learn. **107**, 605–637 (2018)
3. Boutell, M.R., Luo, J., Shen, X., Brown, C.M.: Learning multi-label scene classification. Pattern Recogn. **37**(9), 1757–1771 (2004)
4. Cabral, R., Torre, F., Costeira, J.P., Bernardino, A.: Matrix completion for multi-label image classification. In: Advances in Neural Information Processing Systems, vol. 24 (2011)
5. Charte, F., Rivera, A., del Jesus, M.J., Herrera, F.: Resampling multilabel datasets by decoupling highly imbalanced labels. In: Onieva, E., Santos, I., Osaba, E., Quintián, H., Corchado, E. (eds.) HAIS 2015. LNCS (LNAI), vol. 9121, pp. 489–501. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19644-2_41
6. Charte, F., Rivera, A.J., del Jesus, M.J., Herrera, F.: Mlsmote: approaching imbalanced multilabel learning through synthetic instance generation. Knowl.-Based Syst. **89**, 385–397 (2015)
7. Charte, F., Rivera, A.J., del Jesus, M.J., Herrera, F.: REMEDIAL-HwR: tackling multilabel imbalance through label decoupling and data resampling hybridization. Neurocomputing **326**, 110–122 (2019)
8. Dablain, D., Krawczyk, B., Chawla, N.V.: Deepsmote: fusing deep learning and smote for imbalanced data. IEEE Trans. Neural Netw. Learn. Syst. **34**(9), 6390–6404 (2022)

9. Daniels, Z., Metaxas, D.: Addressing imbalance in multi-label classification using structured hellinger forests. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 31 (2017)
10. Fajardo, V.A., et al.: On oversampling imbalanced data with deep conditional generative models. Expert Syst. Appl. **169**, 114463 (2021)
11. Fürnkranz, J., Hüllermeier, E., Loza Mencía, E., Brinker, K.: Multilabel classification via calibrated label ranking. Mach. Learn. **73**, 133–153 (2008)
12. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, vol. 27 (2014)
13. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006)
14. Jiang, T., Wang, D., Sun, L., Yang, H., Zhao, Z., Zhuang, F.: Lightxml: transformer with dynamic negative sampling for high-performance extreme multi-label text classification. In: AAAI, pp. 7987–7994 (2021)
15. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
16. Liang, J., Phan, H., Benetos, E.: Learning from taxonomy: multi-label few-shot classification for everyday sound recognition. In: ICASSP, pp. 771–775. IEEE (2024)
17. Liu, B., Blekas, K., Tsoumakas, G.: Multi-label sampling based on local label imbalance. Pattern Recogn. **122**, 108294 (2022)
18. Liu, B., Tsoumakas, G.: Making classifier chains resilient to class imbalance. In: Asian Conference on Machine Learning, pp. 280–295. PMLR (2018)
19. Mariani, G., Scheidegger, F., Istrate, R., Bekas, C., Malossi, C.: Bagan: data augmentation with balancing GAN. In: International Conference on Machine Learning (2018)
20. Mullick, S.S., Datta, S., Das, S.: Generative adversarial minority oversampling. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1695–1704 (2019)
21. Pereira, R.M., Costa, Y.M., Silla, C.N., Jr.: MLTL: a multi-label approach for the tomek link undersampling algorithm. Neurocomputing **383**, 95–105 (2020)
22. Razavi, A., Van den Oord, A., Vinyals, O.: Generating diverse high-fidelity images with VQ-VAE-2. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
23. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009. LNCS (LNAI), vol. 5782, pp. 254–269. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04174-7_17
24. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. Mach. Learn. **85**, 333–359 (2011)
25. Charte, F., Rivera, A.J., del Jesus, M.J., Herrera, F.: Addressing imbalance in multilabel classification: measures and random resampling algorithms. Neurocomputing **163**, 3–16 (2015)
26. Sechidis, K., Tsoumakas, G., Vlahavas, I.: On the stratification of multi-label data. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011. LNCS (LNAI), vol. 6913, pp. 145–158. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23808-6_10
27. Tahir, M.A., Kittler, J., Bouridane, A.: Multilabel classification using heterogeneous ensemble of multi-label classifiers. Pattern Recogn. Lett. **33**(5), 513–523 (2012)

28. Tarekegn, A.N., Giacobini, M., Michalak, K.: A review of methods for imbalanced multi-label classification. Pattern Recogn. **118**, 107965 (2021)
29. Teng, Z., Cao, P., Huang, M., Gao, Z., Wang, X.: Multi-label borderline oversampling technique. Pattern Recogn. **145**, 109953 (2024)
30. Tsoumakas, G., Spyromitros-Xioufis, E., Vilcek, J., Vlahavas, I.: Mulan: a java library for multi-label learning. J. Mach. Learn. Res. **12**, 2411–2414 (2011)
31. Tsoumakas, G., Vlahavas, I.: Random k-labelsets: an ensemble method for multilabel classification. In: Kok, J.N., Koronacki, J., Mantaras, R.L., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 406–417. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74958-5_38
32. Zhang, M.L., Li, Y.K., Yang, H., Liu, X.Y.: Towards class-imbalance aware multilabel learning. IEEE Trans. Cybern. **52**(6), 4459–4471 (2020)
33. Zhang, M.L.: ML-rbf: RBF neural networks for multi-label learning. Neural Process. Lett. **29**, 61–74 (2009)
34. Zhang, M.L., Zhou, Z.H.: Multilabel neural networks with applications to functional genomics and text categorization. IEEE Trans. Knowl. Data Eng. **18**(10), 1338–1351 (2006)
35. Zhang, M.L., Zhou, Z.H.: ML-KNN: a lazy learning approach to multi-label learning. Pattern Recogn. **40**(7), 2038–2048 (2007)
36. Zhang, M.L., Zhou, Z.H.: A review on multi-label learning algorithms. IEEE Trans. Knowl. Data Eng. **26**(8), 1819–1837 (2013)
37. Zhu, B., Pan, X., vanden Broucke, S., Xiao, J.: A GAN-based hybrid sampling method for imbalanced customer classification. Inf. Sci. **609**, 1397–1411 (2022)