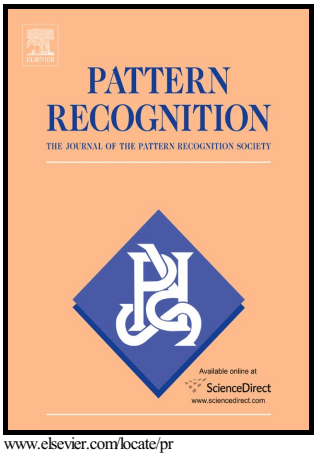


www.elsevier.com/locate/pr

Dual Autoencoders Features for Imbalance Classification Problem

Wing W.Y. Ng, Guangjun Zeng, Jiangjun Zhang, Daniel S. Yeung, Witold Pedrycz

PII: S0031-3203(16)30130-3
DOI: http://dx.doi.org/10.1016/j.patcog.2016.06.013
Reference: PR5767

To appear in: *Pattern Recognition*

Received date: 15 September 2015
Revised date: 16 June 2016
Accepted date: 17 June 2016

Cite this article as: Wing W.Y. Ng, Guangjun Zeng, Jiangjun Zhang, Daniel S. Yeung and Witold Pedrycz, Dual Autoencoders Features for Imbalance Classification Problem, *Pattern Recognition* http://dx.doi.org/10.1016/j.patcog.2016.06.013

Dual Autoencoders Features for Imbalance Classification Problem

Wing W.Y. Ng[a], Guangjun Zeng[a], Jiangjun Zhang[a,*], Daniel S. Yeung[a], Witold Pedrycz[b,c,d]

[a] *Machine Learning and Cybernetics Research Center, School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China*
[b] *Department of Electrical and Computer Engineering, University of Alberta, Canada*
[c] *Department of Electrical and Computer Engineering Faculty of Engineering, King Abdulaziz University Jeddah, 21589, Saudi Arabia*
[d] *Systems Research Institute, Polish Academy of Sciences Warsaw, Poland*

Abstract

Many classification problems encountered in real-world applications exhibit a profile of imbalanced data. Current methods depend on data resampling. In fact, if the feature set provides a clear decision boundary, resampling may not be needed to solve the imbalanced classification problem. Therefore, this work proposes a feature learning method based on the autoencoder to learn a set of features with better classification capabilities of the minority and the majority classes to address the imbalanced classification problems. Two sets of features are learned by two stacked autoencoders with different activation functions to capture different characteristics of the data and they are combined to form the Dual Autoencoding Features. Samples are then classified in the new feature space learned in this manner instead of the original input space. Experimental results show that the proposed method outperforms current resampling-based methods with statistical significance for imbalanced pattern classification problems.

Keywords: Imbalanced Classification, Feature Learning, Stacked Autoencoder

1. Introduction

Imbalanced pattern classification problems occur in many machine learning problems, e.g., network security and medical imaging applications. The disease detection problem classifying "healthy" versus "sick" is an instance of imbalanced classification tasks. In such problems, the number of samples in the minority class is much smaller than that of the majority class. This leads to the problem that high average classification accuracy over all classes can be achieved by even misclassifying all samples of the minority class. Undersampling the majority class and oversampling the minority class are two most commonly used ways to relieve the imbalance problems among classes. However, undersampling-based methods easily loose or deteriorate information about samples (e.g., data distribution) while oversampling-based methods may

*Corresponding author
Email addresses: wingng@ieee.org (Wing W.Y. Ng), zeng.gj@mail.scut.edu.cn (Guangjun Zeng), jjzhangscut@gmail.com (Jiangjun Zhang), danyeung@ieee.org (Daniel S. Yeung), wpedrycz@ualberta.ca (Witold Pedrycz)

suffer from overfitting and becomes sensitive to overlapping between minority and majority classes [1]. The key challenge of dealing with imbalanced pattern classification problems is to create a classifier to classify samples in both majority and minority classes correctly, e.g. yielding high AUC, F_1-score, and G-Mean values. Therefore, re-balancing the two classes is only one of the methods to deal with imbalanced pattern classification problems.

In contrast to balancing the majority and the minority classes by resampling, projecting samples from the input space onto a feature space with a better classification representation between the majority and the minority classes yields a good classification for both classes [2]. The autoencoder [3] yields useful feature representation when its number of neurons positioned at the hidden is larger than the number of input features [3]. By using these learned features as inputs, a classifier yields better classification results in comparison to those when using the original input features. Therefore, we propose the Dual Autoencoding Features (DAF), a feature learning method based on the stacked autoencoder, to relieve the imbalance issue in a pattern classification problem. The DAF is a new application of stacked autoencoders and also provides a new angle to solve the imbalanced classification problem. Two stacked autoencoders using different activation functions are used to learn features from the input space with imbalanced data to capture different characteristics of the data. Then, they are concatenated to form the DAF. The DAF is used to train a classifier for imbalanced pattern classification problems. The major contributions of this study include:

1. to our best knowledge, this paper proposes the first feature learning-based method for dealing with imbalance pattern classification using stacked autoencoders. We provide a new viewpoint for solving a imbalanced pattern classification problem by projecting the input space onto a learned feature space with better representation using stacked autoencoders;
2. the concatenation of learned features from two different stacked autoencoders further enhances the classification accuracy by providing a better representation with different characteristics of the data to the classifier to learn. More specifically, the *sigmoid* activation function is less sensitive to input changes and provides robust representation while the *tanh* activation function provides is more sensitive and provides detailed information of the data.

Section 2 elaborates on some related works for imbalanced classification problems and autoencoders. The DAF is proposed in Section 3. Experimental results reported for 14 UCI datasets are presented and discussed in Section 4. Section 5 concludes this work.

2. Related Works

2.1. Imbalanced Classification Problems

Given an imbalanced training dataset consisting of N samples: $D = \{x^{(1)}, x^{(2)}, ..., x^{(N)}\}$, where $x^{(i)}$ denotes the input vector of the i^{th} sample. In this work, without losing generality, we focus on two-class

problems and assume the negative (positive) class to be the majority (minority) class. In imbalanced classification problems, the number of samples in the majority class (N_n) is far larger than the number of samples in the minority class (N_p) and $N = N_p + N_n$. In many cases, $N_n \gg N_p$ and misclassifying all minority samples to be the majority class may still yield high overall classification accuracy. However, classifiers trained using standard methods (e.g. minimization of overall training error) do not perform any meaningful classification and yield very poor generalization capability on the minority class.

A straightforward solution is to resample either the majority class or the minority class in order to obtain relatively balanced distributions in both classes. The major advantages of the resampling-based method include classifier-independent and possible reuse by different classifiers. The random oversampling (ROS) [4] and the random undersampling (RUS) [4] are two widely applied resampling methods. The ROS duplicates randomly selected samples from the minority class to balance the distributions of the two classes. In contrast, the RUS randomly removes samples from the majority class to gain the balanced dataset. The major drawback of the RUS is the information loss owing to the random deletion of samples while the ROS overfits easily to the minority class because of the large number of artificially duplicated samples. The Synthetic Minority Oversampling Technique (SMOTE) [5] randomly creates artificial samples along a line joining a minority sample and a selected nearest neighbor. However, this approach may increase the overlapping between classes. Variants of the SMOTE are proposed to overcome this limitation, for instances the Borderline-SMOTE [6] and the Safe-level-SMOTE [7]. The Diversified Sensitivity-based Undersampling [8] preserves the structure of the training dataset by clustering the majority class samples and balances the distribution of classes by selecting the most sensitive samples from both clusters of the majority class and all samples in the minority class.

Random-based methods suffer from the variance of performances caused by the random selection of samples. Therefore, ensemble-based methods are proposed to relieve this problem. The RUSBoost [9], the SMOTEBoost [10], the EUSBoost [11], the Inverse RUS (IRUS) [12], the EasyEnsemble and the BalanceCascade [13] are instances of ensemble-based methods. The EasyEnsemble randomly selects multiple subsets of samples from the majority class with replacement. Then, an ensemble of classifiers is trained using the Adaboost with combinations of sample subsets and all samples in the minority class. In contrast, the BalanceCascade performs bagging in a supervised manner to train the base classifier of the ensemble. In iteration, samples from the majority class being correctly classified by the current base classifier are removed to avoid over-learning in the future. The SMOTEBoost integrates the SMOTE and the standard boosting procedure to improve the classification performance on the minority samples by increasing weights of misclassified minority samples. The RUSBoost randomly removes samples in the majority class instead of oversampling in the SMOTEBoost. The IRUS severely undersamples the majority class such that the number of samples in the majority class is smaller than that of the minority class in the training set for each base classifier. Then, a composite classification boundary between the minority class and the majority class

is found by fusing those base classifiers. Instead of random undersampling, the EUSBoost uses evolutionary undersampling to enhance accuracies of base classifiers.

Random-based resampling methods are also combined with feature selection methods for dealing with imbalanced problems in [14, 15]. Samples in the majority class are split into several pseudo-classes and feature selection is performed on both the pseudo-classes and the minority class [14]. It makes the original imbalanced two-class problems into a more complicated multi-class classification problem. The SMOTE is also combined with a feature selection for training SVM to deal with imbalanced classification problems [15]. However, these two "feature selection" methods are essentially resampling-based methods. A feature ranking approach is proposed based on the probability density function of each feature in each class to select features for high-dimensional and small sample sized imbalanced classification problems [16]. The proposed method deals with the imbalanced problem by feature learning without resampling.

2.2. Autoencoders

The representation capability of a shallow neural network with a single hidden layer is limited [17, 3]. At the cost of more time consuming intensive training, deep neural networks yield better feature representations. Hence, the greedy layer-wise training for deep neural network is proposed to reduce the training time [18]. The autoencoder is an unsupervised learning method which learns features from the original input [19]. A deep architecture can be formed by stacking several autoencoders to improve the representation capability of learned features from the input data.

A single autoencoder consists of the input layer, the encoding layer, and the decoding layer. For each sample $x^{(i)}$ in the training dataset $D = \{x^{(1)}, x^{(2)}, ..., x^{(N)}\}$ where $x^{(i)}$ and N denote the n-dimension input vector of the i^{th} sample and the number of samples, respectively. The encoding layer is defined as:

$$f(x) = s_e(W_e x + b_e) \tag{1}$$

where W_e, b_e, and $s_e(\cdot)$ denote the weight matrix, the bias vector, and the activation function of the encoding layer, respectively. In a similar way, the decoding layer is defined as:

$$g(x) = s_d(W_d x + b_d) \tag{2}$$

where W_d, b_d, and $s_d(\cdot)$ denote the weight matrix, the bias vector, and the activation function of the decoding layer, respectively. Hence, the output of the autoencoder for the sample is defined as:

$$y = g(f(x)) \tag{3}$$

The aim of the autoencoder is to learn a feature representation in the encoding layer such that the outputs of the autoencoder reconstruct the inputs. Therefore the learning problem of the autoencoder is to find

a set of parameters $\theta = \{W_e, b_e, W_d, b_d\}$ to minimize the reconstruction error between the inputs and the outputs of the autoencoder:

$$\arg\min_{\theta} \sum_{i=1}^{N} L\left(x^{(i)}, g\left(f\left(x^{(i)}\right)\right)\right) \tag{4}$$

where

$$L(x, y) = \|y - x\|_2 \tag{5}$$

In order to avoid overfitting, the l_2 weight decay penalty is added to restrict the magnitudes of weights [20]. The optimization problem of the autoencoder is rewritten as follows:

$$\arg\min_{\theta} \frac{1}{2} \sum_{i=1}^{N} L\left(x^{(i)}, g\left(f\left(x^{(i)}\right)\right)\right) + \frac{1}{2}\lambda \|W\|_2 \tag{6}$$

where λ and W denote the regularization parameter and the matrix consisting of two weight matrices (W_e and W_d) . Usually, the parameter (θ) of an autoencoder is optimized using the back-propagation method [21] which is the same as the training of a standard multilayer perceptron neural network with one hidden layer, except the desired outputs of the autoencoder are the inputs.

When a linear function is used as the activation function, a single autoencoder with the number of hidden neurons (M) smaller than the number of inputs neurons (n) reduces to the Principle Component Analysis (PCA) [3]. Usually, a nonlinear function is used as the activation function for feature learning. When the $M \leq n$, the autoencoder performs feature reduction similar to the PCA. In contrast, an autoencoder with $M > n$ potentially learns the identity function. Fortunately, in practice, the autoencoder exploits statistical regularities in the dataset and learns useful features [3]. Therefore, in dimensional reduction tasks, M is selected to be smaller than n. In contrast, $M > n$ yields a better set of features for feature learning tasks.

The autoencoder can also be stacked together to derive deeper and more abstract features to support better representation of patterns. In stacked autoencoders, the outputs of the encoding layer of an autoencoder are fed to the next autoencoder as their inputs [22]. Each autoencoder learns in the same way as a single autoencoder and the first autoencoder uses the original input from the data as inputs. In contrast, stacking autoencoders with linear activation function is not useful because the composition of a linear operation yields another linear operation [17].

2.3. Autoencoders for Feature Learning

There are several variants of autoencoders with different regularization terms being added to the objective function for training. The denoising autoencoder corrupts the input by randomly dropping-out some input values of randomly selected samples to enhance the robustness of the autoencoder [23]. Furthermore, the stacked denoising autoencoder [22] stacks several denoising autoencoders to generate a higher representation of data. The contractive autoencoder adds the Frobenius norm of the Jacobian matrix of the autoencoder

activations with respect to the input to the objective function [24]. It yields robust features on the hidden layer by a contraction in the localized input space.

Feature learning with autoencoders are widely used in different application domains. For instances, the autoencoder [24], the denoising autoencoder [23], and the stacked denoising autoencoder [22] learn features from images for improving image recognition precisions. Features learned by the sparse autoencoder are applied to detect human body in depth images [25]. Patches with the size of 16x16 from the full and normalized images are randomly selected to train autoencoders for feature learning. Convolution and pooling are performed on sets of features learned from different patches and the final set of features created is used to train the classifier for human detection in images. On the other hand, both the autoencoder [26, 27] and the denoising autoencoder [28] are used to learn features for speech recognitions. Recursive autoencoders are used to learn features for sentiment distribution prediction [29] and model building for semantic compositionality over a sentiment treebank [30] in natural language processing. An ITG-based translation with vector space representations for variable-sized phrases generated by the recursive autoencoder is proposed in [31].

3. Dual Autoencoding Features

3.1. Framework of DAF

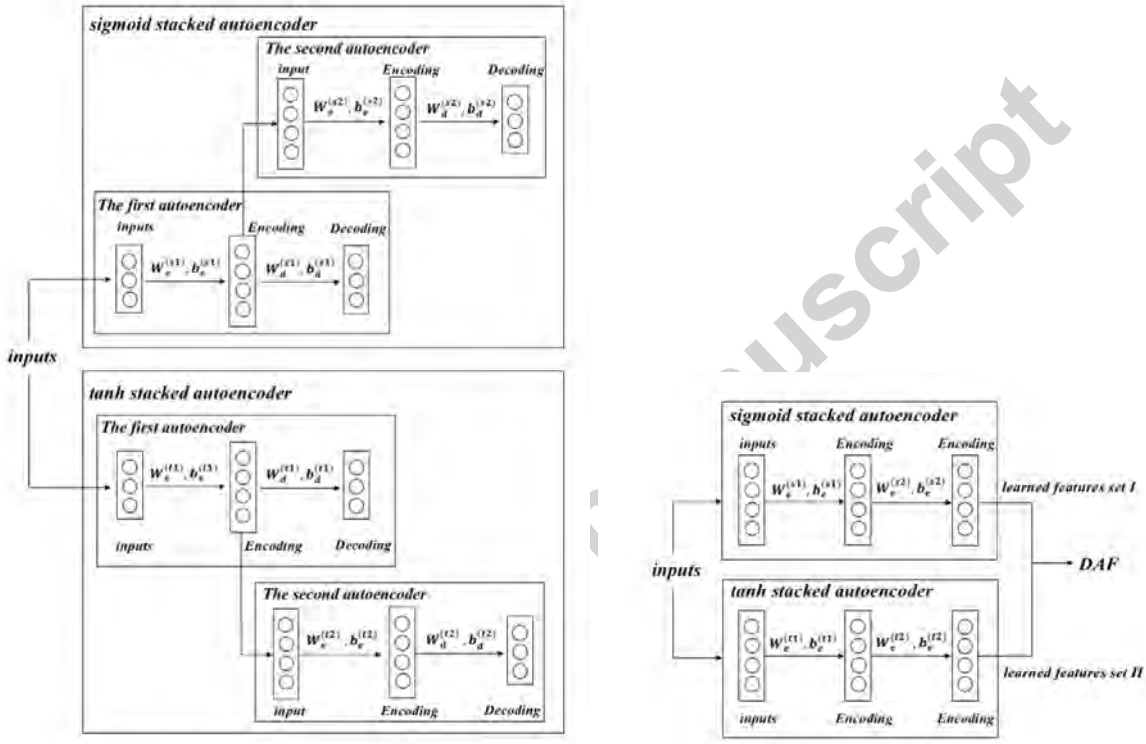
The Dual Autoencoding Features (DAF) combines feature learned from two stacked autoencoders which use the *sigmoid* and the *tanh* functions as activation functions, respectively. Figure 1 shows the overall procedure of the DAF over all datasets. Figure 1a shows the feature learning procedures while Figure 1b displays the feature encoding procedures for samples. The *sigmoid* and the *tanh* functions are defined as follows:

$$sigmoid\,(x) = \frac{1}{1+e^{-x}} \tag{7}$$

$$tanh\,(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{8}$$

The DAF uses 2-layer stacked autoencoders to learn features in this work. The DAF does not limit the number of autoencoders being stacked. However, a single layer autoencoder may not be powerful enough to learn useful features [3, 19, 32] while stacked autoencoders with more than two layers may not yield much better representation at the cost of slower training time. The DAF trains two 2-layer stacked autoencoders independently using different activation functions. The greedy layer-wise training algorithm without the fine-tuning step [18] is applied to train each two-layer stacked autoencoder.

As shown in Figure 1a, each of the 2-layer stacked autoencoders consists of two autoencoders. In each, the first autoencoder takes input from the dataset and encodes them using the encoding layer in it. Then, the second autoencoder uses the outputs of the encoding layer of the first autoencoder as input and learns



(a) Feature learning phase (b) Feature encoding phase

Figure 1: Procedures of (a) the feature learning and (b) the feature encoding of the DAF

to further encode it to a learned feature set. With the two 2-layer stacked autoencoders using different activation functions learning different sets of learned features, they are concatenated to form the DAF. Whenever a sample arrives, the sample is encoded as shown in Figure 1b using connection weights and biases learned in the feature learning phase to transform the input features of the sample to the DAF.

3.2. Training of DAF

The training of each autoencoder uses the standard back-propagation algorithm. Similar to other feed-forward network training, all weights are randomly initialized and biases are set to zero at the beginning of the training. Samples are then fed to the autoencoder in the forward pass to generate output values (y) of the decoding (i.e. output) layer. Then, these values are used to compute the gradients for the weights and the bias updates in the decoding layer as follows:

$$\frac{\partial J}{\partial W_d} = \delta_d \left(g\left(f\left(x\right)\right)\right)^T \tag{9}$$

$$\frac{\partial J}{\partial b_d} = \delta_d \tag{10}$$

where $\delta_d = \partial 0.5 \times \left(\left(y-x\right)^2 + \lambda \left\|W\right\|_2\right) / \partial z$ and $z = W_d f\left(x\right) + b_d$. The gradients for the weights and the bias updates in the encoding layer are computed as follows:

$$\frac{\partial J}{\partial W_e} = \delta_e \left(f\left(x\right)\right)^T \tag{11}$$

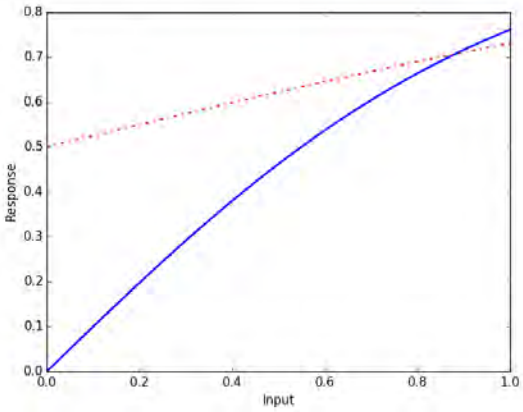
$$\frac{\partial J}{\partial b_e} = \delta_e \tag{12}$$

where $\delta_e = W_e \delta_d f'$ and f' denotes the partial derivative of $f\left(x\right)$. The weights and the biases of autoencoders of the DAF are updated using Equations (9), (10), (11), and (12) by the L-BFGS [33] method. They are updated repeatedly until converge.

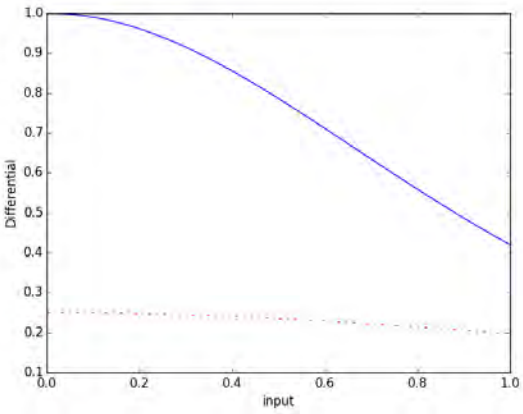
3.3. Discussion of Autoencoders with Different Activations Functions

Autoencoders with different activation functions yield different responses to inputs and different learned features. Both the *tanh* and the *sigmoid* functions are widely used activation functions and the *tanh* function can be regarded as a scaled *sigmoid* function. Activation responses and differentials of both the *tanh* and the *sigmoid* functions in the range of $[0, 1]$ are shown in Figures 2a and 2b, respectively. Only the input range of [0, 1] is shown because input features being fed to autoencoders are usually scaled to $[0, 1]$. Figure 2b shows that the *tanh* function is more sensitive to changes in inputs (i.e. yielding larger partial derivatives) and has a wider range of activation values in comparison to the *sigmoid* function. The partial derivatives of the *sigmoid* and the *tanh* functions are defined as follows:

$$sigmoid'\left(x\right) = \frac{e^{-x}}{\left(1+e^{-x}\right)^2} \tag{13}$$



(a) Activation responses of the *sigmoid* (dotted line) and *tanh* (solid line) with respect to inputs in range of $[0, 1]$



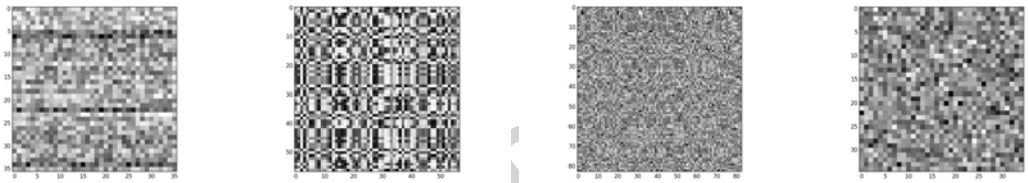
(b) Defferentials of the *sigmoid* (dotted line) and *tanh* (solid line) with respect to inputs in range of $[0, 1]$

Figure 2: Activation responses and partial differential of the *sigmoid* (dotted line) and *tanh* (solid line) with respect to inputs positioned in the range of $[0, 1]$

$$tanh'(x) = \frac{4}{e^{2x} + e^{-2x}} \tag{14}$$

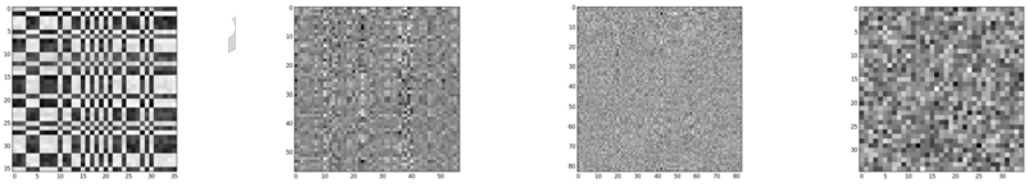
Figure 2b shows that the sensitivities of the *tanh* function are always larger than that of the *sigmoid* function in the range $[0, 1]$. With a high sensitivity, the output of the *tanh* function varies whenever a small change appears in inputs. Therefore, in feature learning, the *tanh* function is more suitable in capturing detailed and local information for representing the data. In contrast, a classifier with low sensitivity yields better robustness and ignores detailed changes in inputs [34]. Hence, the *sigmoid* function is more robust to noise and provides relative global representation to the data. Both global and local information are important for improving classification accuracy. Therefore, the DAF combines sets of features learned from both the *sigmoid* autoencoder and the *tanh* autoencoder to gain benefits from both of them.

Figures 3 and 4 show weight matrices of the encoding layer of the second autoencoder of two-layer stacked autoencoders with *sigmoid* and *tanh* activation functions, respectively, for four UCI datasets. Overall, values of weight matrices of the *sigmoid* stacked autoencoders are smaller (darker in figures) than that of the *tanh* stacked autoencoders'. Hence, features learned by *sigmoid* stacked autoencoders are less sensitive to input changes.



(a) Seeds (b) Satimage (c) Segment (d) Ionos

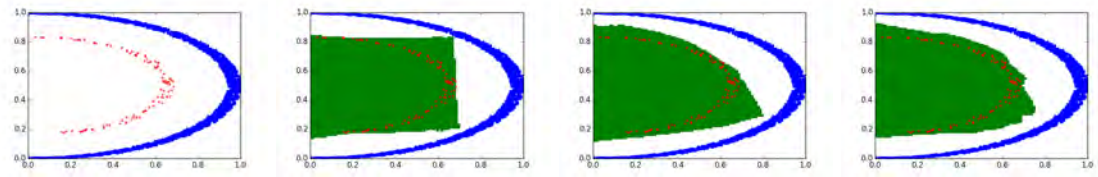
Figure 3: Weight matrices of the second encoding layer of the two-layer stacked autoencoder with *sigmoid* activation function of four UCI datasets



(a) Seeds (b) Satimage (c) Segment (d) Ionos

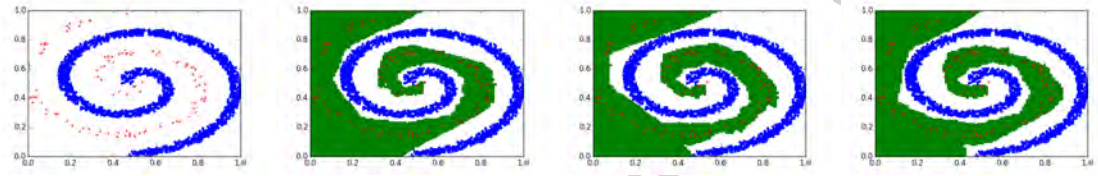
Figure 4: Weight matrices of the second encoding layer of the two-layer stacked autoencoder with *tanh* activation function of four UCI datasets

Experiments using two artificial datasets (A1 and A2) are performed to further demonstrate the differences among the combination of two activation functions in the DAF, the autoencoder using the *sigmoid*



(a) Dataset distribution (b) Sigmoid (c) Tanh (d) DAF

Figure 5: Dataset Distribution of the Dataset A1 and decision boundaries of the DAF, the Sigmoid and the Tanh autoencoders



(a) Dataset distribution (b) Sigmoid (c) Tanh (d) DAF

Figure 6: Dataset Distribution of the Dataset A2 and decision boundaries of the DAF, the Sigmoid and the Tanh autoencoders

Table 1: Average F_1-Score of the DAF, the Sigmoid and the Tanh autoencoders for A1 and A2

Method	A1	A2
DAF	0.9985	0.9733
SIGMOID	0.9930	0.9682
TANH	0.9981	0.9643

function only, and the autoencoder using the *tanh* function only. All three autoencoders use the same number of hidden neurons. In each artificial dataset, 2000 and 100 samples are generated for the majority and the minority classes, respectively. Figures 5 and 6 show the distribution of samples and decision boundaries yielded by the three autoencoders for artificial datasets A1 and A2, respectively. The *tanh* yields a better decision boundary for A1 (Figure 5 (c)) while the *sigmoid* yields a better decision boundary for A2 (Figure 6 (b)). In contrast, the DAF yields good decision boundaries for both artificial datasets. Moreover, Table 1 shows that the DAF yields the best $F_1 - Score$ among all methods for both A1 and A2 where the $F_1 - Score$ is defined in Equation (15). These show that different activation functions perform well in different situations. The DAF combining two different activation functions enjoy the good performances of both activation functions for different situations.

Algorithm 1 shows the feature learning procedures of the DAF while Algorithm 2 shows the feature encoding procedures for a given sample set. For a given imbalanced classification problem, the training dataset is used to learn the DAF using Algorithm 1 and then samples in both the training and the testing sets are encoded using Algorithm 2 to get the DAF for both sets. The DAF of the training dataset is used to train the classifier for solving the imbalanced classification problem.

Algorithm 1 Feature learning of the DAF

Require: N, n, l, m_j, and D denote the number of samples, the number of features, and the number of autoencoders, the number of neurons on the encoding layer of the j^{th} autoencoders, and the $n \times N$ input feature matrix, respectively

Ensure: Two stacked autoencoders with *sigmoid* and *tanh* activation function, respectively.

1: Scale each input feature to the range of $[0, 1]$.

2: Set $\varphi = D$, and $m_0 = n$.

3: **for** $j = 1$ to l **do**

3.1: Train an autoencoder with the *sigmoid* activation function using φ and Equation (6) with m_j neurons on the encoding layer and $m_{(j-1)}$ neurons on both the input and the decoding layer.

3.2: Compute the outputs of the encoding layer for all training sample to get $m_j \times N$ matrix H_j.

3.3: Set $\varphi = H_j$

4: **end for**

5: Repeat Step 3 to train another stacked autoencoder by replacing the *sigmoid* activation function by the *tanh* function.

3.4. Computational Complexity of DAF

Suppose we have m training samples and n hidden neurons in each autoencoders of DAF. Our experiments in Section 4.2 show that a autoencoder [3] in the same type using only one type of activation requires

Algorithm 2 Feature encoding of the DAF

Require: X denotes the $n \times N_X$ input feature matrix of a given set of samples.

Ensure: $2m_l \times N_X$ learned feature matrix H_{DAF}

1: Get two stacked autoencoders with *sigmoid* and *tanh* activation functions by Algorithm 1.
2: Compute the values of the encoding layer of the l_{th} *sigmoid* stacked autoencoder for all samples in X to get $m_l \times N_X$ matrix $H_{sigmoid}$
3: Compute the values of the encoding layer of the l_{th} *tanh* stacked autoencoder for all samples in X to get $m_l \times N_X$ matrix H_{tanh}
4: Concatenate $H_{sigmoid}$ and H_{tanh} by rows to get the learned feature matrix H_{DAF}.

$2n$ hidden neurons to yield similar (but worse) performance of the DAF. The computation of gradients of both weights and bias need to perform matrix multiplication which has a complexity of $O\left(n^k\right)$ where k is a constant varying by different matrix multiplication algorithms. The DAF has a matrix of 2 $m \times n$ to be optimized so the complexity is only $O\left(2n^k\right)$. In contrast, a autoencoder using $2n$ hidden neurons with a matrix of $m \times (2n)$ has a complexity of $O\left((2n)^k\right)$. Therefore, the DAF yields similar or better performance over autoencoder using a single type of activation function with smaller computational complexity. Furthermore, the training time of the DAF depends on the training time of the autoencoders being used. More efficient autoencoders, e.g. [35], may further improve the training time of the DAF.

4. Experiments

The DAF is compared with existing methods on 14 UCI datasets [36]. Table 2 shows the characteristics of the 14 UCI datasets. For multi-class datasets, one of the classes is selected as the minority class while all the other classes are combined to form the majority class. The second column of Table 2 shows the minority classes of the 14 UCI datasets. The DAF is compared with single stacked autoencoders in Section 4.2 to show the effectiveness of the combination of feature sets from two stacked autoencoders in the DAF. Section 4.3 shows the comparisons between the DAF and existing methods including resampling-based and feature projection methods. Experiments on six artificial datasets are shown in Section 4.1 to demonstrate the decision boundaries learned by classifiers trained using different methods.

The AUC, the F_1-score, and the G-Mean are used to evaluate the performances of these methods [8] [37] [38]. The AUC measures the area under receiver operating characteristic curve. The F_1-score is computed as follows:

$$F_1 = 2\frac{P \times R}{P + R} \tag{15}$$

$$P = \frac{TP}{TP + FP} \tag{16}$$

Table 2: Characteristic of 14 UCI Datasets

dataset	mc	feature#	samples#	ratio
breast_w	2	9	350	0.346
ionos	1	34	176	0.358
ionosphere	2	34	113	0.358
newthyroid	3	5	108	0.139
tic_tac_toe	2	9	313	0.346
vote	2	16	218	0.385
wine	3	13	90	0.267
card	1	51	346	0.445
ecoli	3	7	170	0.153
segment	1	19	1155	0.143
satimage	3	36	3219	0.211
waveform	2	40	2501	0.331
vowel	1	10	495	0.091
seeds	1	7	210	0.333

$$R = TPR = \frac{TP}{TP + FN} \tag{17}$$

where TP, FP, and FN denote the number of true positive samples, the number of false positive samples, and the number of false negative samples, respectively. In this work, the minority class is regarded as the positive class, thus TP is equal to the number of correctly classified samples in the minority class. The G-Mean measures the precisions of both the majority and the minority classes and is computed as follow:

$$G = \sqrt{TPR \times TNR} \tag{18}$$

$$TNR = \frac{TN}{TN + FP} \tag{19}$$

where TN denotes the number of true negative samples.

All methods, except the RUSBoost, use the random forest (RF) as the classifier [39]. The implementation combining classifiers by the average of their probabilistic prediction, instead of majority vote, is used in our experiments [40]. However, the RUSBoost performs ensemble of decision trees training together with the resampling, so it does not use the RF as classifier. Default values are used for parameters of methods in comparison. The DAF uses $m_1 = m_2$ and these values of each experiment are selected between one to three times of the number of inputs via empirical validation.

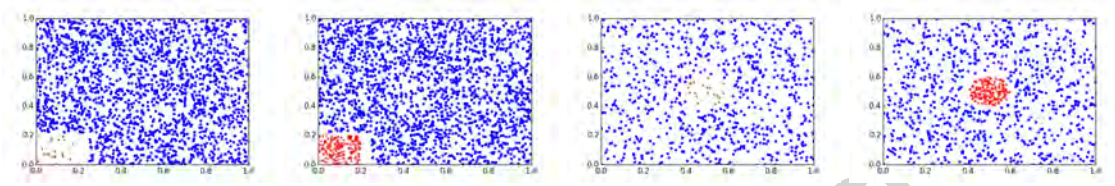
For each dataset, 30 independent runs are performed with a 50-50 random split of training and testing sets. Averages and standard deviations over the 30 runs of the AUC, the F_1-score, and the G-Mean are reported. The t-test is performed to verify the statistical significances of our experimental results. The *, the &, the #, and the @ indicate that the DAF outperforms another method with 99%, 95%, 90%, and 80% statistical significances, respectively.

4.1. Experiments on Artificial Datasets

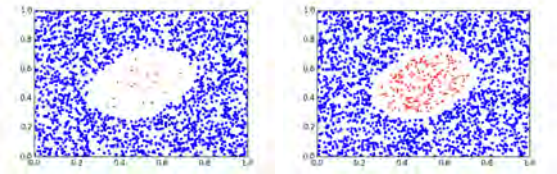
In this section, we examine the decision boundaries learned by classifiers trained by different methods using six artificial datasets. The aim is to show the benefit of feature learning over resampling-based and feature projection methods. Figure 7 shows the six artificial datasets. There are three sets of artificial datasets and each consists of two datasets sampled from the same distribution but with different imbalance ratios. The imbalance ratio of Artificial Datasets 1-a, 2-a, and 3-a (1-b, 2-b, and 3-b) is 1:100 (1:10). The red "x" (blue ".") represents a sample in the minority (majority) class.

Figures 8, 9, 10, 11, 12, 13 show decision boundaries of classifiers learned using different methods for Artificial Datasets 1-a, 1-b, 2-a, 2-b, 3-a, and 3-b, respectively. In these figures, samples in the shaded-area are classified as the minority by the classifier. The DAF yields the best decision boundaries in all the six experiments. Decisions boundaries of classifiers trained using undersampling-based methods (i.e.

the IRUS, the RUS, and the RUSBoost) usually expanded to the majority class and therefore they easily misclassify majority samples as minority's. In contrast, oversampling-based methods (i.e. the ROS and the SMOTE) yield decision boundaries in the area of minority class instead of the boundaries between two classes. However, owing to the small size of the minority class, the poor decision boundaries of oversampling-based methods yield less bad influence to the AUC, the F_1-score and the G-Mean values in comparison with undersampling-based methods. In these experiments using simple 2-dimensional datasets, both feature projection methods yield a rotation of the original features and decision boundaries biased to the majority class. The original RF method yields poor decision boundaries and heavily biased to the majority class when the imbalance ratio is 1:100.



(a) Artificial Dataset 1-a (b) Artificial Dataset 1-b (c) Artificial Dataset 2-a (d) Artificial Dataset 2-b



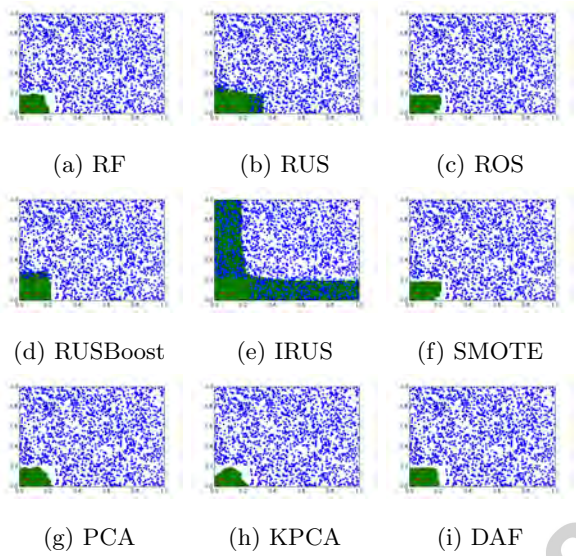
(e) Artificial Dataset 3-a (f) Artificial Dataset 3-b

Figure 7: Distributions of six artificial datasets

Overall, the DAF learns good feature representation to the data and the classifier trained using the learned features yields the best decision boundary. This validates that a good feature set relieves the imbalance issue in the pattern classification problems.

4.2. Comparison with Single Stacked Autoencoder

The DAF is compared with the S1x1, the S1x2, the T1x1, and the T1x2. The "S1" and the "T1" denote the single two-layer stacked autoencoders with the *sigmoid* and the *tanh* activation functions, respectively. The "x1" and the "x2" indicate a single two-layer stacked autoencoder with m_l and $2m_l$ neurons, respectively, on the encoding layer where m_l denotes the number of neurons on the encoding layer for the DAF. The DAF combines learned features from the S1x1 and the T1x1. Therefore, comparing the DAF with the S1x1 and the T1x1 shows the effectiveness of the combination of feature sets from two stacked autoencoders with different activation functions. On the other hand, the DAF consists of $2m_l$ learned features and hence the S1x2 and the T1x2 are compared to prevent the difference in the number of learned features for classifiers.

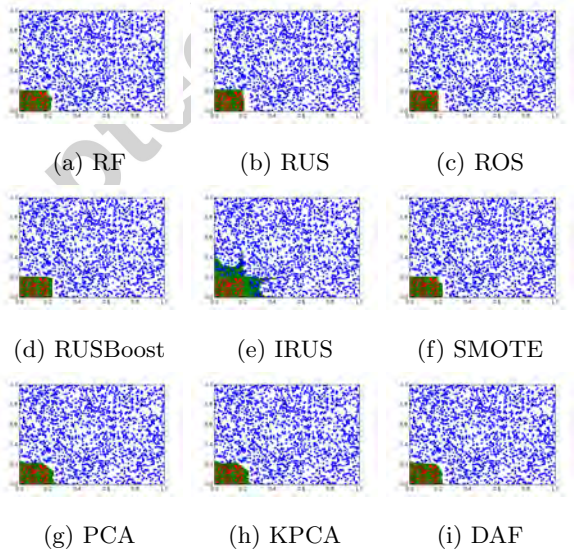


(a) RF (b) RUS (c) ROS

(d) RUSBoost (e) IRUS (f) SMOTE

(g) PCA (h) KPCA (i) DAF

Figure 8: Decision boundaries of classifiers trained using different methods for the Artificial Dataset 1-a

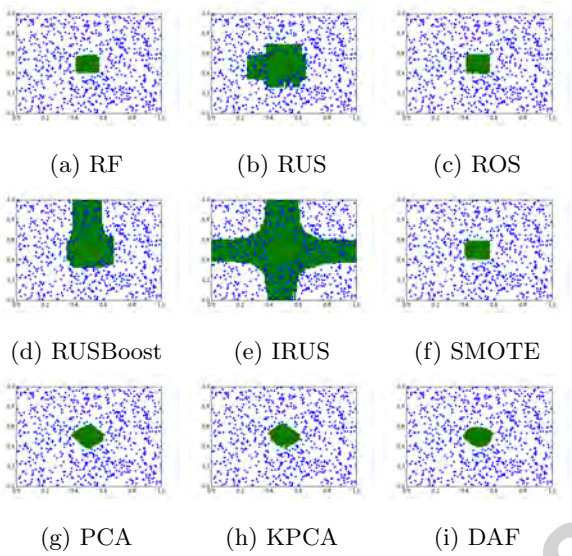


(a) RF (b) RUS (c) ROS

(d) RUSBoost (e) IRUS (f) SMOTE

(g) PCA (h) KPCA (i) DAF

Figure 9: Decision boundaries of classifiers trained using different methods for the Artificial Dataset 1-b

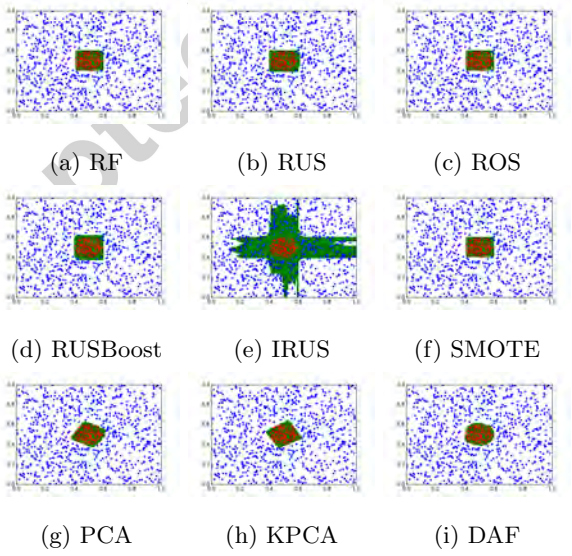


(a) RF (b) RUS (c) ROS

(d) RUSBoost (e) IRUS (f) SMOTE

(g) PCA (h) KPCA (i) DAF

Figure 10: Decision boundaries of classifiers trained using different methods for the Artificial Dataset 2-a

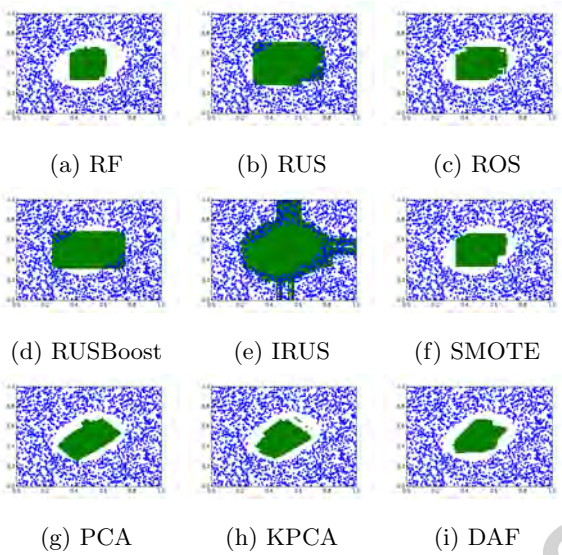


(a) RF (b) RUS (c) ROS

(d) RUSBoost (e) IRUS (f) SMOTE

(g) PCA (h) KPCA (i) DAF

Figure 11: Decision boundaries of classifiers trained using different methods for the Artificial Dataset 2-b

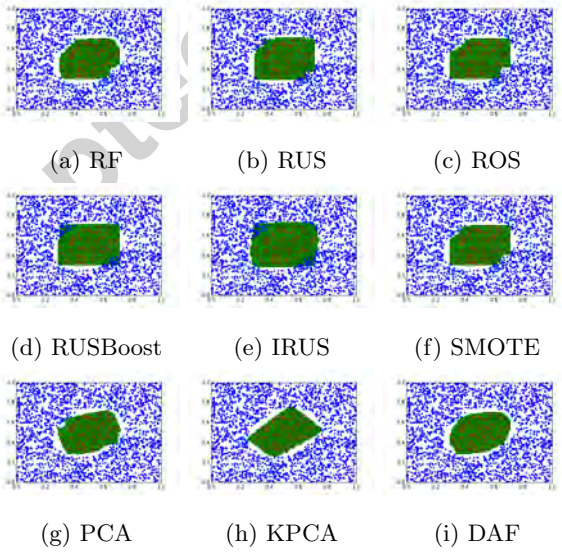


(a) RF (b) RUS (c) ROS

(d) RUSBoost (e) IRUS (f) SMOTE

(g) PCA (h) KPCA (i) DAF

Figure 12: Decision boundaries of classifiers trained using different methods for the Artificial Dataset 3-a



(a) RF (b) RUS (c) ROS

(d) RUSBoost (e) IRUS (f) SMOTE

(g) PCA (h) KPCA (i) DAF

Figure 13: Decision boundaries of classifiers trained using different methods for the Artificial Dataset 3-b

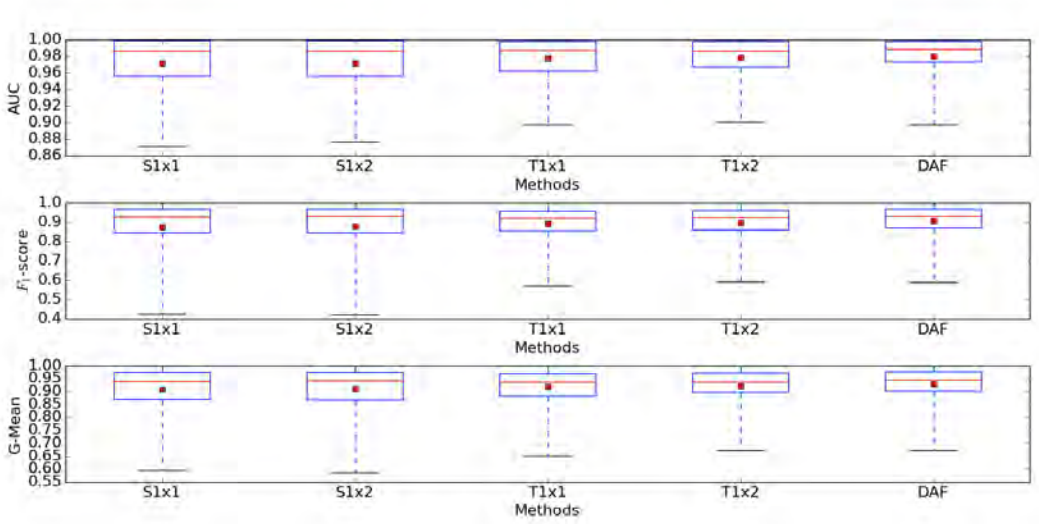


Figure 14: Boxplots of overall performance of the DAF and the four single stacked autoencoders over all datasets

Table 3 shows the overall performance of the DAF and the four different stacked autoencoders over 14 UCI datasets. The DAF yields the best average AUC, F_1-score and G-Mean values over all four single stacked autoencoders. The DAF outperforms both the S1x1 and the S1x2 with 99% statistical significances. In contrast, the DAF is statistically significantly better than the T1x1, but not T1x2 in the AUC and the F_1-score. These show that features learned by the *tanh* stacked autoencoders captures better representation of the data in comparison to the *sigmoid* stacked autoencoders. However, it may be sensitive to noise in data. Both the average and the standard deviation of AUC, F_1-score and G-Mean values of the T1x1 and the T1x2 are all worse than the DAF because the DAF uses both the sensitive and robust feature information from *tanh* and *sigmoid* stacked autoencoders. The boxplots in Figure 14 show that the DAF yields the best over performance with the best stability in the AUC, the F_1-score and the G-Mean.

Table 3: Average and standard deviations of AUC, F_1-score G-Mean of the DAF and the four single stacked autoencoders over 14 UCI datasets

Metric	AUC	F_1-score	G-Mean
S1x1	0.972 ± 0.034*	0.876 ± 0.137*	0.909 ± 0.095*
S1x2	0.972 ± 0.034*	0.877 ± 0.137*	0.910 ± 0.094*
T1x1	0.977 ± 0.025@	0.893 ± 0.094#	0.918 ± 0.074&
T1x2	0.978 ± 0.024	0.897 ± 0.092	0.921 ± 0.072@
DAF	0.980 ± 0.024	0.905 ± 0.090	0.928 ± 0.070

4.3. Comparisons with Existing Methods

In this section, we compare the DAF with existing resampling-based and feature projection methods. The RUS [4], the ROS [4], the IRUS [12], the SMOTE [5] and the RUSBoost [9] are representative resampling-based methods which include varieties of representative methods in undersampling, oversampling and ensemble resampling methods. On the other hand, the PCA [41] and the Kernel PCA (KPCA) [42] are representative feature projection methods. The PCA is a widely used linear feature projection methods while the KPCA is a nonlinear modification of the PCA. Feature projection methods are compared to show the effectiveness of feature learning by the DAF over both linear and nonlinear feature projection methods. The RF using the original input features without any resampling is also compared and serves as the baseline for comparisons. The ensemble learning of the RF relieves the imbalance problem in certain extends [43].

Table 7 shows the overall AUC, F_1-score, and G-Mean over all datasets of different methods while Figure 15 shows the boxplots of overall performance of different methods. Both of them show that the DAF outperforms all other methods. In particular, the DAF outperforms the RUSBoost, the PCA and the KPCA with 99% statistical significances for all three metrics (the AUC, the F_1-score and the G-Mean) and outperforms the RF, the RUS and the IRUS with 99% statistical significances in at least one metric.

Tables 4, 5 and 6 show the AUC, the F_1-score and the G-Mean of experiments using different methods for the 14 UCI datasets, respectively. Among 336 experiments, the DAF outperforms other methods with 99% statistical significances in 276 experiments (i.e. 82.14% of experiments). This shows that the DAF is an effective method for dealing with imbalanced pattern classification problems in comparison to the baseline RF, current resampling-based methods, and feature projection methods. Both the PCA and the KPCA yield much worse performance in comparison to the DAF. This may be due to the single layer of feature projection in both the PCA and the KPCA while the DAF uses a two-layer feature projection in the encoding. Moreover, both feature projection methods reduce or maintain the number of features after projection while the encoding of the DAF increases the number of features in the learned set. On the other hand, the DAF yields better performance than resampling-based methods in 173 out of 210 experiments (82.38% of experiments). This shows that a set of good learned features yields a better imbalance pattern classification results than resampling. The DAF uses all given real training samples while the oversampling-based methods use a large number of artificial or duplicated samples to balance the two classes. This may yield a overfitting classifier and a misled distribution of the minority class. On the other hand, undersampling methods significantly distorted the distribution of the majority class which may reduce the classification generalization performance. Ensemble methods only reduce the variance of classifier performance, but still may suffer from aforementioned problems.

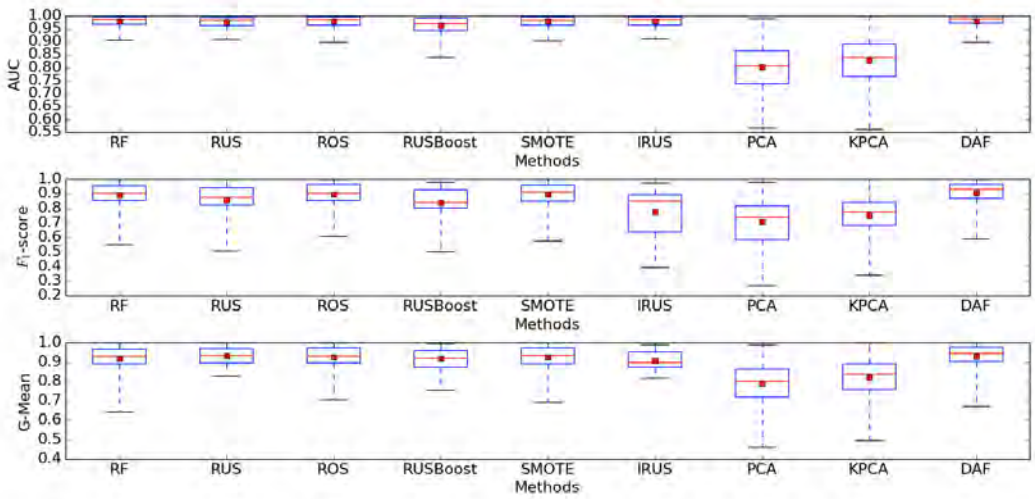


Figure 15: Boxplots of overall performance of the DAF and other methods over all datasets

Table 4: Average, standard deviation of AUC of different methods on 14 UCI datasets

Dataset	RF	ROS	RUS	IRUS	SMOTE	RUSBOOST	PCA	KPCA	DAF
breast_w	0.989 ± 0.003	0.989 ± 0.004	0.988 ± 0.004 @	0.990 ± 0.003	0.989 ± 0.004	0.982 ± 0.007 *	0.945 ± 0.019 *	0.933 ± 0.026 *	0.990 ± 0.003
ionos	0.972 ± 0.010 *	0.971 ± 0.010 *	0.964 ± 0.012 *	0.965 ± 0.011 *	0.970 ± 0.009 *	0.935 ± 0.021 *	0.819 ± 0.039 *	0.806 ± 0.036 *	0.980 ± 0.009
ionosphere	0.976 ± 0.007 *	0.976 ± 0.009 &	0.972 ± 0.009 *	0.972 ± 0.010 *	0.975 ± 0.010 *	0.942 ± 0.017 *	0.801 ± 0.041 *	0.796 ± 0.040 *	0.982 ± 0.009
newthyroid	0.998 ± 0.001 &	0.998 ± 0.002 &	0.996 ± 0.005 *	0.999 ± 0.002	0.998 ± 0.002 *	0.992 ± 0.015 &	0.899 ± 0.070 *	0.910 ± 0.076 *	0.999 ± 0.001
tic_tac_toe	0.997 ± 0.002 *	0.997 ± 0.002 *	0.991 ± 0.005 &	0.996 ± 0.003 &	0.997 ± 0.002 *	0.991 ± 0.006 *	0.670 ± 0.041 *	0.678 ± 0.030 *	0.994 ± 0.003
vote	0.991 ± 0.004	0.991 ± 0.004	0.991 ± 0.003	0.992 ± 0.003	0.991 ± 0.003	0.980 ± 0.010 *	0.871 ± 0.030 *	0.863 ± 0.027 *	0.991 ± 0.004
wine	0.999 ± 0.001	1.000 ± 0.000	1.000 ± 0.001	1.000 ± 0.000 @	1.000 ± 0.001	0.995 ± 0.005 *	0.914 ± 0.081 *	0.902 ± 0.107 *	1.000 ± 0.001
card	0.927 ± 0.010 &	0.924 ± 0.010 @	0.926 ± 0.009 &	0.930 ± 0.009 *	0.926 ± 0.009 &	0.897 ± 0.016 *	0.769 ± 0.029 *	0.767 ± 0.023 *	0.921 ± 0.010
ecoli	0.958 ± 0.021 @	0.954 ± 0.024	0.954 ± 0.023	0.958 ± 0.024	0.956 ± 0.021	0.931 ± 0.033 &	0.735 ± 0.067 *	0.798 ± 0.084 *	0.950 ± 0.025
segment	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000 *	1.000 ± 0.000 &	1.000 ± 0.000	0.997 ± 0.003 *	0.711 ± 0.064 *	0.917 ± 0.027 *	1.000 ± 0.000
satimage	0.953 ± 0.005 *	0.957 ± 0.005	0.946 ± 0.005 *	0.950 ± 0.005 *	0.957 ± 0.005	0.942 ± 0.005 *	0.671 ± 0.014 *	0.725 ± 0.021 *	0.957 ± 0.005
waveform	0.968 ± 0.002 *	0.964 ± 0.002 *	0.964 ± 0.003 *	0.966 ± 0.002 *	0.964 ± 0.002 *	0.951 ± 0.004 *	0.826 ± 0.030 *	0.857 ± 0.011 *	0.975 ± 0.002
vowel	0.997 ± 0.003 *	0.998 ± 0.003 &	0.995 ± 0.004 *	0.996 ± 0.004 *	0.998 ± 0.001 *	0.990 ± 0.007 *	0.788 ± 0.055 *	0.835 ± 0.041 *	0.999 ± 0.001
seeds	0.978 ± 0.003	0.976 ± 0.003 &	0.972 ± 0.004 *	0.976 ± 0.001 &	0.975 ± 0.003 *	0.957 ± 0.005 *	0.820 ± 0.036 *	0.833 ± 0.040 *	0.978 ± 0.005

Table 5: Average, standard deviation of F_1-score of different methods on 14 UCI datasets

Dataset	RF	ROS	RUS	IRUS	SMOTE	RUSBOOST	PCA	KPCA	DAF
breast_w	0.948 ± 0.013 &	0.954 ± 0.011	0.954 ± 0.010	0.953 ± 0.009 @	0.952 ± 0.011 @	0.937 ± 0.011 *	0.927 ± 0.020 *	0.912 ± 0.030 *	0.956 ± 0.010
ionos	0.899 ± 0.022 &	0.894 ± 0.019 *	0.888 ± 0.023 *	0.874 ± 0.023 *	0.901 ± 0.020 &	0.846 ± 0.039 *	0.769 ± 0.051 *	0.752 ± 0.048 *	0.915 ± 0.026
ionosphere	0.901 ± 0.022 *	0.899 ± 0.021 *	0.890 ± 0.021 *	0.880 ± 0.025 *	0.900 ± 0.024 *	0.850 ± 0.026 *	0.745 ± 0.054 *	0.738 ± 0.053 *	0.923 ± 0.024
newthyroid	0.919 ± 0.043 *	0.922 ± 0.062 &	0.891 ± 0.067 *	0.638 ± 0.035 *	0.925 ± 0.041 *	0.870 ± 0.060 *	0.828 ± 0.102 *	0.842 ± 0.124 *	0.956 ± 0.032
tic_tac_toe	0.967 ± 0.013 #	0.973 ± 0.011	0.915 ± 0.029 *	0.888 ± 0.023 *	0.968 ± 0.012 @	0.940 ± 0.017 *	0.567 ± 0.055 *	0.578 ± 0.039 *	0.972 ± 0.007
vote	0.944 ± 0.013	0.943 ± 0.013	0.943 ± 0.013	0.939 ± 0.011 #	0.944 ± 0.013	0.930 ± 0.018 *	0.842 ± 0.038 *	0.832 ± 0.034 *	0.944 ± 0.012
wine	0.976 ± 0.030	0.987 ± 0.020	0.979 ± 0.020	0.897 ± 0.027 *	0.989 ± 0.019	0.908 ± 0.048 *	0.874 ± 0.117 *	0.858 ± 0.158 *	0.984 ± 0.022
card	0.852 ± 0.014	0.853 ± 0.012	0.855 ± 0.011 #	0.857 ± 0.011 &	0.856 ± 0.015 #	0.818 ± 0.023 *	0.743 ± 0.034 *	0.741 ± 0.028 *	0.850 ± 0.011
ecoli	0.809 ± 0.054 &	0.833 ± 0.047	0.740 ± 0.047 *	0.593 ± 0.026 *	0.823 ± 0.066	0.718 ± 0.053 *	0.551 ± 0.103 *	0.651 ± 0.122 *	0.840 ± 0.062
segment	0.987 ± 0.007 &	0.985 ± 0.009	0.973 ± 0.008 *	0.876 ± 0.015 *	0.985 ± 0.008 @	0.956 ± 0.011 *	0.503 ± 0.106 *	0.858 ± 0.037 *	0.982 ± 0.007
satimage	0.616 ± 0.025 *	0.659 ± 0.018 *	0.544 ± 0.016 *	0.403 ± 0.006 *	0.657 ± 0.017 *	0.529 ± 0.012 *	0.406 ± 0.021 *	0.506 ± 0.032 *	0.642 ± 0.020
waveform	0.857 ± 0.007 *	0.853 ± 0.007 *	0.842 ± 0.005 *	0.800 ± 0.005 *	0.853 ± 0.006 *	0.824 ± 0.008 *	0.768 ± 0.039 *	0.809 ± 0.015 *	0.868 ± 0.007
vowel	0.930 ± 0.042 *	0.925 ± 0.039 *	0.754 ± 0.051 *	0.476 ± 0.013 *	0.933 ± 0.032 *	0.767 ± 0.043 *	0.634 ± 0.092 *	0.709 ± 0.059 *	0.959 ± 0.037
seeds	0.873 ± 0.018 &	0.859 ± 0.024 *	0.829 ± 0.024 *	0.809 ± 0.008 *	0.844 ± 0.017 *	0.810 ± 0.016 *	0.741 ± 0.047 *	0.754 ± 0.052 *	0.882 ± 0.009

Table 6: Average, standard deviation of G-Mean of different methods on 14 UCI datasets

Dataset	RF	ROS	RUS	IRUS	SMOTE	RUSBOOST	PCA	KPCA	DAF
breast_w	0.963 ± 0.012 *	0.968 ± 0.010 @	0.971 ± 0.009	0.973 ± 0.005	0.967 ± 0.010 &	0.957 ± 0.010 *	0.945 ± 0.019 *	0.932 ± 0.027 *	0.972 ± 0.008
ionos	0.916 ± 0.019 &	0.914 ± 0.015 *	0.912 ± 0.017 *	0.909 ± 0.018 *	0.920 ± 0.016 @	0.882 ± 0.032 *	0.814 ± 0.041 *	0.800 ± 0.040 *	0.928 ± 0.023
ionosphere	0.917 ± 0.018 *	0.918 ± 0.017 *	0.914 ± 0.017 *	0.913 ± 0.020 *	0.920 ± 0.019 *	0.886 ± 0.022 *	0.794 ± 0.044 *	0.791 ± 0.043 *	0.935 ± 0.021
newthyroid	0.943 ± 0.042 *	0.940 ± 0.055 *	0.964 ± 0.033 #	0.902 ± 0.016 *	0.950 ± 0.037 *	0.959 ± 0.030 &	0.892 ± 0.079 *	0.904 ± 0.086 *	0.979 ± 0.027
tic_tac_toe	0.970 ± 0.011	0.978 ± 0.009 &	0.945 ± 0.019 *	0.929 ± 0.016 *	0.972 ± 0.011	0.959 ± 0.013 *	0.660 ± 0.045 *	0.669 ± 0.032 *	0.973 ± 0.006
vote	0.956 ± 0.010	0.956 ± 0.010	0.957 ± 0.009	0.955 ± 0.008	0.955 ± 0.010	0.945 ± 0.014 *	0.870 ± 0.031 *	0.862 ± 0.028 *	0.956 ± 0.010
wine	0.979 ± 0.028 @	0.989 ± 0.018	0.992 ± 0.008	0.956 ± 0.013 *	0.990 ± 0.017	0.951 ± 0.029 *	0.910 ± 0.090 *	0.897 ± 0.119 *	0.987 ± 0.019
card	0.867 ± 0.012	0.867 ± 0.011	0.869 ± 0.010 #	0.869 ± 0.010 #	0.870 ± 0.013 #	0.833 ± 0.021 *	0.767 ± 0.029 *	0.766 ± 0.023 *	0.864 ± 0.010
ecoli	0.849 ± 0.050 *	0.880 ± 0.046	0.893 ± 0.030	0.854 ± 0.016 *	0.885 ± 0.057	0.885 ± 0.028	0.704 ± 0.091 *	0.777 ± 0.106 *	0.888 ± 0.052
segment	0.989 ± 0.007 &	0.991 ± 0.007 *	0.991 ± 0.004 *	0.976 ± 0.003 *	0.991 ± 0.007 *	0.986 ± 0.005 @	0.675 ± 0.085 *	0.915 ± 0.029 *	0.984 ± 0.007
satimage	0.702 ± 0.021 *	0.764 ± 0.017 *	0.873 ± 0.010 *	0.823 ± 0.004 *	0.770 ± 0.017 *	0.867 ± 0.009 *	0.615 ± 0.024 *	0.689 ± 0.029 *	0.730 ± 0.019
waveform	0.897 ± 0.006 *	0.900 ± 0.005 #	0.900 ± 0.004 &	0.868 ± 0.004 *	0.900 ± 0.005 @	0.883 ± 0.007 *	0.824 ± 0.031 *	0.856 ± 0.011 *	0.902 ± 0.005
vowel	0.940 ± 0.038 *	0.949 ± 0.035 &	0.963 ± 0.009	0.883 ± 0.007 *	0.961 ± 0.024	0.960 ± 0.014 @	0.763 ± 0.071 *	0.822 ± 0.050 *	0.969 ± 0.030
seeds	0.921 ± 0.013 *	0.907 ± 0.019 *	0.894 ± 0.019 *	0.883 ± 0.006 *	0.896 ± 0.013 *	0.876 ± 0.012 *	0.820 ± 0.037 *	0.831 ± 0.041 *	0.931 ± 0.006

Table 7: Overall results of AUC, F_1-score, G-Mean across all 14 UCI datasets

Methods	AUC	F_1-score	G-Mean
RF	0.979 ± 0.022	0.891 ± 0.095&	0.915 ± 0.075*
ROS	0.978 ± 0.023	0.896 ± 0.087@	0.925 ± 0.064
RUS	0.976 ± 0.0236&	0.857 ± 0.116*	0.931 ± 0.044
IRUS	0.978 ± 0.023	0.777 ± 0.171*	0.907 ± 0.046*
SMOTE	0.978 ± 0.022	0.895 ± 0.088#	0.925 ± 0.061
RUSBoost	0.963 ± 0.033*	0.836 ± 0.113*	0.916 ± 0.050*
PCA	0.803 ± 0.096*	0.707 ± 0.163*	0.790 ± 0.111*
DPCA	0.830 ± 0.088*	0.753 ± 0.131*	0.822 ± 0.096*
DAF	0.980 ± 0.024	0.905 ± 0.090	0.928 ± 0.070

5. Conclusion

In this work, we propose a new approach to cope with imbalanced pattern classification problems by feature learning using the Dual Autoencoding Features (DAF). Experimental results show that the DAF yields statistical significant improvement in the AUC, the F_1-score and the G-Mean in comparison to representative methods of resampling-based and feature projection methods. These results demonstrate that the DAF combing learned feature sets from the *sigmoid* and the *tanh* stacked autoencoders yields significantly better feature representation of samples and such that the classifier using the DAF yields better performance evaluated by the AUC, the F_1-score and G-Mean for dealing with the imbalanced pattern classification problems. The major advantage of the DAF is the combination of detailed and sensitive information learned by the *tanh* activation function and the robust information learned by the *sigmoid* activation function from the data. We also show that classifiers trained using the DAF yield the best decision boundaries in artificial datasets.

In our future works, other types of autoencoders could be added and examined for their capabilities and characteristics of improving the DAF. Furthermore, the dual autoencoders will be extended to multiple autoencoders. New challenges include the evaluation and analysis of different autoencoders for their suitabilities for imbalanced problems and the development of a method to determine the number of stacked autoencoders (i.e. number of learned feature sets) and the number of autoencoders per stack (i.e. the number of encoding layers). Among these issues, the determination of the optimal number of stacked layers is still an open problem in deep learning.

Acknowledgement

This work was supported by the National Natural Science Foundation of China under Grants 61572201 and 61272201, and the Fundamental Research Funds for the Central Universities (2015ZZ023).

References

[1] V. García, J. S. Sánchez, R. A. Mollineda, On the effectiveness of preprocessing methods when dealing with different levels of class imbalance, Knowledge-Based Systems 25 (1) (2012) 13–21.

[2] M. Wasikowski, X.-w. Chen, Combating the small sample class imbalance problem using feature selection, IEEE Transactions on Knowledge and Data Engineering 22 (10) (2010) 1388–1400.

[3] Y. Bengio, Learning deep architectures for AI, Foundations and Trends in Machine Learning 2 (1) (2009) 1–127.

[4] H. He, E. Garcia, Learning from Imbalanced Data, IEEE Transactions on Knowledge and Data Engineering 21 (9) (2009) 1263–1284.

[5] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, Journal of artificial intelligence research 16 (2002) 321–357.

[6] H. Han, W.-Y. Wang, B.-H. Mao, Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning, in: Advances in intelligent computing, 2005, pp. 878–887.

[7] C. Bunkhumpornpat, K. Sinapiromsaran, 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, 2009, pp. 475–482.

[8] W. Ng, J. Hu, D. Yeung, S. Yin, F. Roli, Diversified Sensitivity-Based Undersampling for Imbalance Classification Problems, IEEE Transactions on Cybernetics 45 (11) (2015) 2402–2412.

[9] C. Seiffert, T. Khoshgoftaar, J. Van Hulse, A. Napolitano, RUSBoost: A Hybrid Approach to Alleviating Class Imbalance, IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 40 (1) (2010) 185–197.

[10] N. V. Chawla, A. Lazarevic, L. O. Hall, K. W. Bowyer, Smoteboost: Improving prediction of the minority class in boosting, in: Proceedings of Knowledge Discovery in Databases, 2003, pp. 107–119.

[11] M. Galar, A. Fernández, E. Barrenechea, F. Herrera, EUSBoost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling, Pattern Recognition 46 (12) (2013) 3460–3471.

[12] M. A. Tahir, J. Kittler, F. Yan, Inverse random under sampling for class imbalance problem and its application to multi-label classification, Pattern Recognition 45 (10) (2012) 3738–3750.

[13] X.-Y. Liu, J. Wu, Z.-H. Zhou, Exploratory undersampling for class-imbalance learning, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 39 (2) (2009) 539–550.

[14] L. Yin, Y. Ge, K. Xiao, X. Wang, X. Quan, Feature selection for high-dimensional imbalanced data, Neurocomputing 105 (2013) 3–11.

[15] S. Maldonado, R. Weber, F. Famili, Feature selection for high-dimensional class-imbalanced data sets using Support Vector Machines, Information Sciences 286 (2014) 228–246.

[16] M. Alibeigi, S. Hashemi, A. Hamzeh, Dbfs: An effective density based feature selection scheme for small sample size and high dimensional imbalanced data sets, Data & Knowledge Engineering 81 (2012) 67–103.

[17] Y. Bengio, A. Courville, P. Vincent, Representation learning: A review and new perspectives, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (8) (2013) 1798–1828.

[18] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, Advances in neural information processing systems 19 (2007) 153.

[19] P. Baldi, Autoencoders, Unsupervised Learning, and Deep Architectures., ICML Unsupervised and Transfer Learning 27 (2012) 37–50.

[20] J. Moody, S. Hanson, A. Krogh, J. A. Hertz, A simple weight decay can improve generalization, Advances in neural information processing systems 4 (1995) 950–957.

[21] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, Cognitive modeling 5 (1988) 3.

[22] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion, The Journal of Machine Learning Research 11 (2010) 3371–3408.

[23] P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol, Extracting and Composing Robust Features with Denoising Autoencoders, in: Proceedings of the 25th International Conference on Machine Learning, 2008, pp. 1096–1103.

[24] S. Rifai, P. Vincent, X. Muller, X. Glorot, Y. Bengio, Contractive auto-encoders: Explicit invariance during feature extraction, in: Proceedings of the 28th International Conference on Machine Learning, 2011, pp. 833–840.

[25] S.-Z. Su, Z.-H. Liu, S.-P. Xu, S.-Z. Li, R. Ji, Sparse auto-encoder based feature learning for human body detection in depth image, Signal Processing 112 (2015) 43–52.

[26] J. Deng, Z. Zhang, F. Eyben, B. Schuller, Autoencoder-based Unsupervised Domain Adaptation for Speech Emotion Recognition, IEEE Signal Processing Letters 21 (9) (2014) 1068–1072.

[27] S. Chandar A P, S. Lauly, H. Larochelle, M. Khapra, B. Ravindran, V. C. Raykar, A. Saha, An Autoencoder Approach to Learning Bilingual Word Representations, in: Advances in Neural Information Processing Systems 27, 2014, pp. 1853–1861.

[28] F. Weninger, S. Watanabe, Y. Tachioka, B. Schuller, Deep recurrent de-noising auto-encoder and blind de-reverberation for reverberated speech recognition, in: Proceedings of 2014 IEEE International Conference on Acoustics, Speech and Signal Processing, 2014, pp. 4623–4627.

[29] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, C. D. Manning, Semi-supervised recursive autoencoders for predicting sentiment distributions, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2011, pp. 151–161.

[30] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, C. Potts, Recursive deep models for semantic compositionality over a sentiment treebank, in: Proceedings of the conference on empirical methods in natural language processing, Vol. 1631, 2013, p. 1642.

[31] P. Li, Y. Liu, M. Sun, Recursive autoencoders for itg-based translation., in: Proceedings of EMNLP, 2013, pp. 567–577.

[32] Y. Bengio, O. Delalleau, On the expressive power of deep architectures, in: Proceedings of Algorithmic Learning Theory, 2011, pp. 18–36.

[33] R. H. Byrd, P. Lu, J. Nocedal, C. Zhu, A limited memory algorithm for bound constrained optimization, SIAM Journal on Scientific Computing 16 (5) (1995) 1190–1208.

[34] D. S. Yeung, J.-C. Li, W. W. Ng, P. P. Chan, Mlpnn training via a multiobjective optimization of training error and stochastic sensitivity, IEEE Transactions on Neural Networks and Learning Systems 27 (5) (2016) 978–992.

[35] M. Chen, Z. Xu, K. Weinberger, F. Sha, Marginalized denoising autoencoders for domain adaptation, in: Proceedings of the 29th International Conference on Machine Learning (ICML-12), ICML '12, Omnipress, 2012, pp. 767–774.

[36] M. Lichman, UCI machine learning repository (2013).

[37] H. He, E. Garcia, et al., Learning from imbalanced data, Knowledge and Data Engineering, IEEE Transactions on 21 (9) (2009) 1263–1284.

[38] N. V. Chawla, Data mining for imbalanced datasets: An overview, in: Data Mining and Knowledge Discovery Handbook, Springer, 2010, pp. 875–886.

[39] L. Breiman, Random forests, Machine learning 45 (1) (2001) 5–32.

[40] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[41] I. Jolliffe, Principal component analysis, Springer, 2002.

[42] B. Schölkopf, A. Smola, K.-R. Müller, Kernel principal component analysis, in: Artificial Neural Networks-ICANN'97, 1997, pp. 583–588.

[43] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, F. Herrera, A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 42 (4) (2012) 463–484.