Neurocomputing 358 (2019) 222–234

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Multi-label classification using a cascade of stacked autoencoder and extreme learning machines

Anwesha Law, Ashish Ghosh*

Machine Intelligence Unit, Indian Statistical Institute, 203 B. T. Road, Kolkata 700108, India

ARTICLE INFO

Article history:
Received 22 November 2018
Revised 26 April 2019
Accepted 17 May 2019
Available online 22 May 2019

Communicated by Dr Miche Y

Keywords:
Multi-label classification
Extreme learning machine
Stacked autoencoder
Multi-label extreme learning machine

ABSTRACT

This article introduces a cascade of neural networks for classification of multi-label data. Two types of networks, namely, stacked autoencoder (SAE) and extreme learning machine (ELM) have been incorporated in the proposed system. ELM is a compact and efficient single-label classifier which seems to lose its efficiency while dealing with multi-label data. This happens due to the complex nature of the multi-label data, which makes it difficult for the smaller networks to interpret it accurately. In our proposed work, we attempt to deal with few of the bottlenecks faced while handling multi-label data. Thus, we aim to enhance the performance of a stand-alone multi-label extreme learning machine (MLELM) by collaborating it with other networks. There are three basic phases in the proposed method: feature encoding, soft classification and class score approximation. In the first step, an SAE network is employed to generate a discriminating and reduced input representation of the multi-label data. This makes the data compact and more manageable for the successive stages. This data in turn is used by an MLELM in the next phase for the prediction of soft labels. In the final step, to improve the prediction capability of the previous network, a novel approach of approximating the class score is proposed using an additional MLELM. Comprehensive experimental evaluation of the proposed approach has been performed on seven datasets against eleven relevant algorithms, and overall it displays a promising performance.

1. Introduction

Traditional supervised machine learning algorithms [1,2] mostly deal with data which have a single corresponding label, i.e., each sample belongs to only one class. For example, while categorizing the sentiment of a person from a text snippet he/she has written, it might be labeled as either “happy”, “sad” or “neutral”. However, frequently we come across data that cannot be represented with a single label. This means that the documents we are analyzing may have multiple emotions involved; hence, it can belong to classes like “angry” and “sad” at the same time. This kind of data which can belong to multiple classes at once is known as multi-label data. In the past decade, researchers have developed various algorithms [3,4] to deal with multi-label data, mostly focusing on multi-label classification techniques. Among the different approaches to multi-label classification, the branch of *problem adaptation* techniques [4–6] have been significantly explored. This approach mainly focusses on modifying existing algorithms for binary/single-label classification in a way that can handle multi-label classification efficiently without changing the multi-label property of the data itself. Few well-known classification tools like neural networks [4,7–9], support vector machines [5], decision trees [6], etc. have been appropriately adapted by researchers to perform efficient multi-label classification.

Since multi-label data belongs to more than one classes at a time, the corresponding class boundaries invariably overlap thus making the decision space quite complex. To handle this problem efficiently, many researchers prefer using artificial neural networks (ANNs) [2,10–12] to learn the complex multi-label class boundaries. In literature, adaptations of multi-layer perceptron (MLP) [4], radial basis functions (RBF) [7], extreme learning machine (ELM) [8], deep neural networks (DNN) [13], etc. have been done by researchers to classify multi-label data efficiently. From these works, it is seen that more complex architectures become computationally bulky, whereas extremely simple networks may not be able to work as desired. Thus, while developing a classifier, efficiency and simplicity both need to be handled simultaneously. Keeping the above aspects in mind, our focus is directed towards one such simple yet effective model known as an extreme learning machine (ELM) [14]. ELMs are quite compact and perform efficiently when dealing with single-label data. Hence, adaptation of this single layer model has been made by various researchers to perform multi-label classification [15–17]. However, it is seen that the

* Corresponding author.
E-mail address: ash@isical.ac.in (A. Ghosh).

https://doi.org/10.1016/j.neucom.2019.05.051
0925-2312/

performance of a stand-alone multi-label ELM (MLELM) relatively deteriorates when it comes to multi-label data classification. It is not able to approximate the weights for data with multiple outputs as efficiently as it can do for single-label data. The complex multi-label data proves to be bulky for the simple one-pass ELM network. This motivated us to build a model that utilizes the strengths of ELMs and handles its drawbacks to improve the overall performance in the field of multi-label classification.

While dealing with multi-label data, researchers are faced with certain challenges which need to be handled effectively. In this article, a cascade of neural networks has been proposed which attempts to handle a few of the setbacks in the field of multi-label data classification. As mentioned previously, the focus of this model is directed towards ELMs. To strengthen the performance of ELMs, two specific issues faced due to the complexity of multi-label data has been handled using a couple of complementing networks. One challenge is to handle the dimensionality and representation of the input space for multi-label data. Simple ELMs have the requirement of a huge number of nodes in the hidden layer as compared to the input layer for efficient approximation of weights. This proves to be a bottleneck while using ELM for multi-label classification. Since, the input dimension is often quite large, it drastically increases the number of weights to be learnt by the network in one-pass, thus reducing the approximation capability of the network. To battle this problem, another simple yet efficient neural network, namely autoencoder (AE) [10], is used in the proposed model. Autoencoders are unsupervised networks which are capable of learning effective encoding of data. They produce a good representation of the actual data which makes the consecutive learning steps more functional. The other challenge faced is the effective mapping of the input to the complex output space. Due to this complexity, the one-pass learning scheme of ELMs falter while learning all the overlapping relations among classes. To give this mechanism an additional scope of learning we introduce a novel and slightly modified MLELM in coalition with the original MLELM to effectively approximate the class scores. Utilizing the simplistic nature of ELMs and effectiveness of AEs, we introduce a classifier system based on stacked AEs and ELMs to perform multi-label classification efficiently.

The proposed model is a cascade of stacked autoencoders and multi-label ELMs. It has three phases: feature extraction [18], soft classification [19], class score approximation. The first phase of the proposed model is feature extraction: this is performed using a stacked autoencoder. The large input dimension is thus transformed to a smaller, appropriate space which is used in the subsequent classification phase. In the classification phase of the model, a multi-label ELM is employed to predict the soft class labels for the data. The outputs generated in this phase is then fed to another MLELM in the final step. The last step is for class score approximation in which the second MLELM learns to map the soft class labels to its corresponding hard class labels. After the final class mapping is learned, a global threshold is calibrated to assign the hard class labels to the data. The three-step cascade of classifiers proposed in this article is a novel approach of handling couple of the challenges faced by multi-label classifiers. This approach of stacking networks has been used in accordance with the concept of stacked generalization [20]. However, the main contribution lies in the technique of using an extreme learning machine to learn the mapping of the class scores obtained from the previous classifier to the hard labels. As per the knowledge of the authors, this approach has not been attempted by researchers previously. The proposed model has been validated with seven multi-label datasets along with comparative analysis against eleven relevant algorithms.

In the next section, two preliminary topics namely stacked autoencoders and ELMs have been discussed. Few existing ELM-based multi-label classification techniques have been mentioned as related works in Section 3. In Section 4 the proposed method has been discussed in detail. Section 5 has a comparative analysis of the results obtained; and Section 6 is the concluding section of the article.

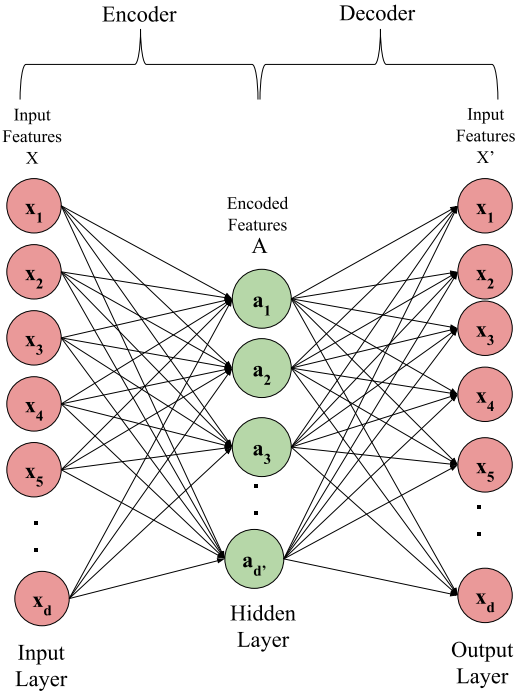

Fig. 1. Architecture of an autoencoder.

2. Preliminaries

Among the problem adaptation techniques which exist in the literature, few of the multi-label classifier models have used ELMs and autoencoders. Since the proposed method is based on MLELM and stacked autoencoder (SAE), a brief description of the related models like SAE, ELM, MLELM and other ELM-based techniques are given in the following sub-sections.

2.1. Stacked autoencoders

In this era of deep learning, the use of autoencoders (AEs) [21,22] have increased in various domains. Among the different types of autoencoders in the field of machine learning like denoising AE [23], deep AE [10], we focus on stacked autoencoders (SAEs) [24,25]. A simple AE is a variation of a feedforward neural network which has three layers: input, hidden and output. For N samples, each with feature vector $\mathbf{X_i}$, for $i = 1, \ldots, N$ and $\mathbf{X_i} \in R^d$, both the input and the output layer of the autoencoder has d nodes. The autoencoder works in an unsupervised fashion, unlike a regular feed forward network.

An example of a simple autoencoder is shown in Fig. 1. The autoencoder takes $\mathbf{X} = \{x_1, x_2, x_3, \ldots, x_d\}$ as both input and output. The task of an autoencoder is to encode the d-dimensional data to d' dimension in the first part (i.e., encode) and then decode the features from d' to d dimension in the decoder part. When d' is smaller than d, the autoencoder compresses the data to a smaller dimension and then uncompresses it in the next layer. If the input to the autoencoder is $\mathbf{X}$, the encoder maps it to a set of hidden nodes $\mathbf{A} = \{a_1, a_2, \ldots, a_{d'}\}$ where output at a node a_j is computed as

$$a_j = \phi\left(\sum_{i=1}^{d} w_{ij}x_i + b_j\right), \tag{1}$$

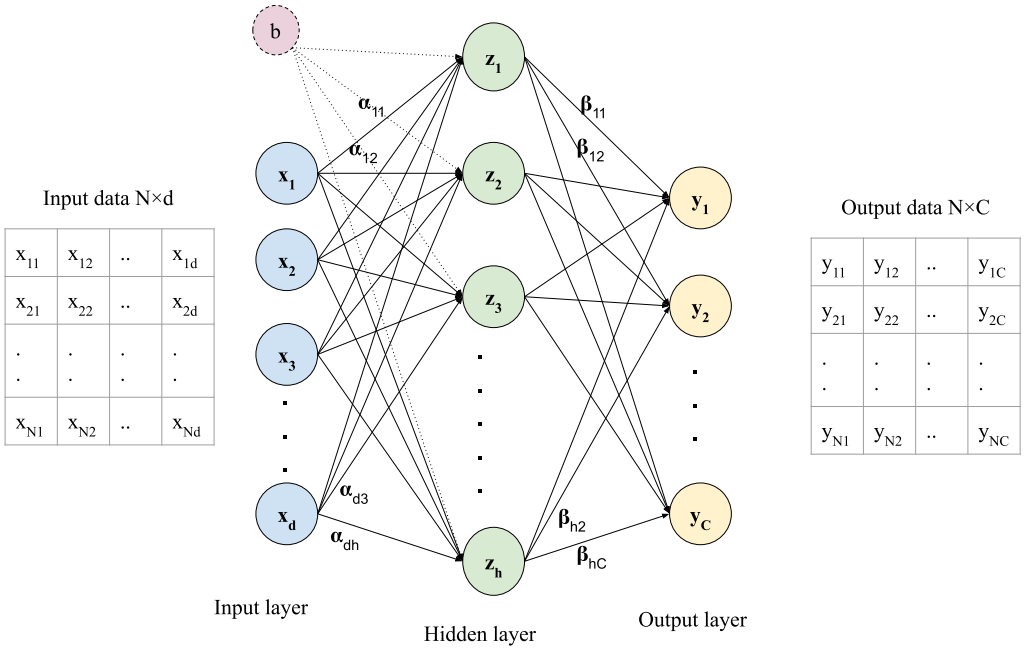

Fig. 2. Architecture of ELM.

where, ϕ is the transfer function for the encoder, w_{ij} is the connection weight between the x_i and a_j, and b_j is the bias. The function of the decoder is to map the encoded representation **A** back to an estimate **X′** of the original input **X**. Thus, the value at the kth output node x'_k is computed as follows.

$$x'_k = \psi\left(\sum_{j=1}^{d'} w_{jk} a_j + b'_k\right), \tag{2}$$

where, ψ is the transfer function for the decoder, w_{jk} is the connection weight between a_j and x'_k, and b'_k is the bias for the kth node of the decoder. The autoencoder is trained iteratively and the weights are updated through backpropagation like a multi-layer perceptron.

In the scenario of a stacked autoencoder, multiple autoencoders are sequentially placed one after the other. The encoded data from one AE is passed on as input to the next AE and so on, to further extract prominent features from the data. Autoencoders are used for various purposes, such as encoding of features, weight initialization of other networks, etc. In the proposed model we have used SAE in the form of a feature extractor before performing classification using ELMs.

2.2. Extreme learning machine

Extreme learning machine is a compact and efficient single layer feed-forward neural network which has been quite popular in the past decade. This unique network performs classification task efficiently and fast. It has been found that the learning speed of ELM can be a lot more than traditional feed-forward network while obtaining better generalization performance [26].

The architecture of the ELM has 3 layers: input, hidden and output. For a single-label sample **X**, the class is denoted as Y, where $\mathbf{X} \in R^d$ and $\mathbf{Y} \in R^C$. The input layer of an ELM has d nodes, hidden layer has h nodes and the output layer has C nodes. The network has input to hidden layer connection weights α, biases b and hidden to output layer weights β. An illustration of the ELM network is given in Fig. 2.

The uniqueness of this network is that the input layer weights and biases are randomly initialized and unlike most other ANNs, they are not updated any further. Learning from the data occurs in the hidden layer weights alone. An activation function ϕ is used at the hidden nodes. The output at any hidden node z_j is computed as

$$z_j = \phi\left(\sum_{i=1}^{d} \alpha_{ij} x_i + b_j\right), \tag{3}$$

where, $j = 1, \ldots, h$, α_{ij} is the connection weight between x_i and hidden node z_j, and b_j is the bias. Similarly at any output node y_k the final outcome is

$$y_k = \sum_{j=1}^{h} z_j \beta_{jk}, \tag{4}$$

where, β_{jk} is the weight from hidden node z_j to output node y_k. The above model is compactly represented as

$$\mathbf{Y} = \mathbf{Z}\beta, \tag{5}$$

where, $\mathbf{Y} = \{y_1, y_2, \ldots, y_C\}$, $\mathbf{Z} = \{z_1, z_2, \ldots, z_h\}$ and β is the $h \times C$ weight matrix between the hidden layer and the output layer. The outputs from the hidden layer **Z** and the output layer **Y** are already known. Thus, the hidden to output layer weight matrix β is approximated as:

$$\beta = \mathbf{Z}^{\dagger}\mathbf{Y}, \tag{6}$$

where, $\mathbf{Z}\dagger$ is the Moore–Penrose inverse of **Z** [27,28]. Once the weight matrix β is obtained, the ELM model has completed its training phase and is ready for testing. Class prediction for unknown samples are then performed like in any other feed-forward network.

Among the many applications of ELM, one is being used as an auto-encoder (ELM-AE) [29,30]. In general, ELM-AE is a sparse autoencoder and the input data is expanded in the hidden layer due to the presence of large number of hidden nodes in ELM. In the proposed method we require feature reduction, hence have opted for a stacked autoencoder instead of an ELM-AE. Kasun et al. [29] built an ELM-AE as a part of a deep model called multi-layer ELM specifically for image classification. In the stacked ELM model developed by Zhou et al. [30], the authors have divided a single

large ELM network into multiple stacked serially connected smaller ELMs. They have utilized ELM-AE in each iteration of the stacked-ELM algorithm to improve performance.

3. Related literature

Researchers have used the ELM network in different ways to classify multi-label data. The most simple form is referred to as multi-label ELM (MLELM) [15,31], which is an adaptation of the simple ELM network. These models do not require any architectural change; but the output provided at the last layer is multi-label in nature. Venkatesan and Er [15] used the MLELM with a bipolar representation of labels to improve the learning of the simple ELM; and Sun et al. [31] described a sample-wise threshold function using ELM called ELM-ML.

Few other works exist in literature, where the ELM has been used in various forms for multi-label classification. L_{21}-norm Minimization ELM [16] is an ELM based algorithm which combines the smallest training error of ELM with L_{21}-norm minimization of the hidden to output layer weight matrix. Canonical Correlation Analysis based ELM (CCA-ELM) was introduced by Kongsorot and Horata [8]. Correlations among the input features and the set of labels are computed using CCA, then the input space and label space are mapped to the new space. An ELM is used to classify and finally map the original input space. Zhang et al. [17] proposed a multi layer ELM-RBF for multi-label learning (ML-ELM-RBF). It is built from radial basis function for multi-label learning (ML-RBF) and weight uncertainty ELM-AE (WuELM-AE). This model stacks WuELM-AE to form a deep network, and then it performs clustering analysis on sample features of each possible class to compose the last hidden layer.

Along with the ELM-based techniques specifically tailored for multi-label classification, other ELM-based single-label models can also be employed as multi-label classifiers. Since, ELMs and neural networks in general do not require much architectural modification to switch from single-label to multi-label classification, they can be adapted to solve the multi-label classification problem. In [32], a fast pruned ELM was proposed that can automatically generate the number of hidden nodes. The relevance of the initial large number of hidden nodes is measured and the irrelevant ones are pruned. Rong et al. [33] developed an online sequential fuzzy ELM (OS-Fuzzy-ELM) which trains the ELM in an online batch-wise mode. The initial training is done with a chunk of data and the later chunks are used to update the parameters. In [34], an aircraft recognition system had been built which extracts three moment invariant features from the input aircraft images and feeds them as inputs to three separate modules of neural networks. These modules consist of ELMs which in turn perform classification, and the final outcome from the modules are combined. Niu et al. [35] proposed a self-adjusting ELM (SA-ELM) which learns the input to hidden layer weights using an ameliorated teaching learning based optimization instead of using the random weights. Apart from these, many other methods also exist in literature which can be adapted for multi-label classification.

However, unlike the above mentioned ELM-based techniques, the proposed method aims to improve the performance of ELMs while handling multi-label data specifically. In general, multi-label data tends to have a large number of features, which in turn requires the MLELM to use a larger number of hidden nodes. Not only does this increase the cost, it also degrades the performance of the network. Hence, it is not able to approximate the weights for data with multiple outputs as efficiently as it can do for single-label data. Also, the decision space of multi-label data is inherently quite complex. A simple MLELM is unable to map the input to the output space efficiently and learn the separating decision boundaries. The complex multi-label data proves to be bulky for the simple one-pass ELM network. Thus, the proposed method aims to handle these shortcomings with the use of a cascade of networks, where an individual network is employed to handle separate issues.

Apart from ELM based techniques for multi-label classification, there are numerous other methods that exist in literature [36–38]. One of the most popular approaches is binary relevance (BR) [36]. This method incorporates multiple binary classifiers, one for each label, and their outputs are combined to generate the predicted label-set. Classifier chains (CC) [37] also trains multiple classifiers, one for each label, choosing the order of training randomly. The first classifier is trained using only the original input attributes. The first output label is then added as new input attribute, and the new input space is used to train the second classifier, and so on. This forms a chain of classifiers which is quite popular, and has been extended to ensemble of classifier chains (ECC) [37], quick classifier chains (CCq) etc. Among ensemble techniques, Tsoumakas and Vlahavas [38] developed random k label-sets (RAkEL) method that generates random subsets of labels, training a multiclass classifier for each subset. Various efficient methods are being developed by researchers and the literature is continuously expanding in the field of multi-label classification.

4. Proposed method

In this article, a cascade of stacked autoencoders and extreme learning machines has been used for the classification of multi-label data (MLSAEELM). In the first phase, a stacked autoencoder network is used to reduce the dimension of the data. Stacked autoencoders (SAEs) are widely used for deep neural networks since they are capable of efficiently extracting underlying features from any given data. Although the proposed classifier is not too deep in nature, a reduced set of well encoded features obtained from the SAE are beneficial to retain the underlying property of the data. Once the relevant features have been extracted, these are passed on to a multi-label ELM for soft label prediction. This MLELM handles the multi-label data in a way similar to the simple ELM. To improve the classification performance of the MLELM, another MLELM is concatenated in the final phase of the model. The task of this network is to learn the mapping of the soft labels to hard labels. The final class scores predicted by the approximation MLELM are then used to assign hard class labels to the data.

4.1. Representation of multi-label data

By definition, each ith multi-label data instance $\mathbf{X_i} = \{x_{i1}, x_{i2}, \ldots, x_{id}\}$, $i = 1, \ldots, N$ is associated with a vector of multiple outputs $\mathbf{Y_i} = \{y_{i1}, y_{i2}, \ldots, y_{iC}\}$, where C is the number of classes. Among these C classes, the multi-label data instance can belong to more than one class at a time. Each element of the vector $\mathbf{Y_i}$ is a binary value, indicating if the corresponding label is relevant to the sample or not. Several labels can be active at once, unlike in the case of single-label data where only one label is active at a time. Each distinct combination of labels is known as a label-set. This representation of multi-label data is used while discussing the architecture of the proposed model.

4.2. Architecture of the network

The architecture of the proposed model includes a stacked autoencoder followed by a multi-label extreme learning machine for classification. Fig. 3 shows the overview of the model.

4.2.1. Stacked autoencoder

The stacked autoencoder part of the network contains individual autoencoders stacked sequentially. The first autoencoder AE_1

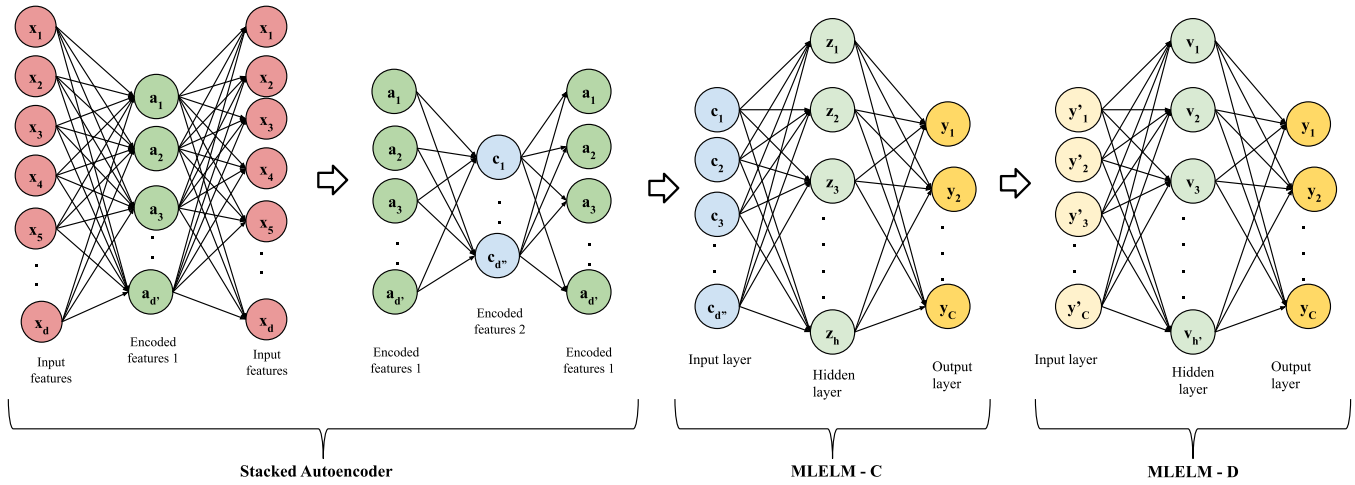

Fig. 3. Architecture of MLSAEELM.

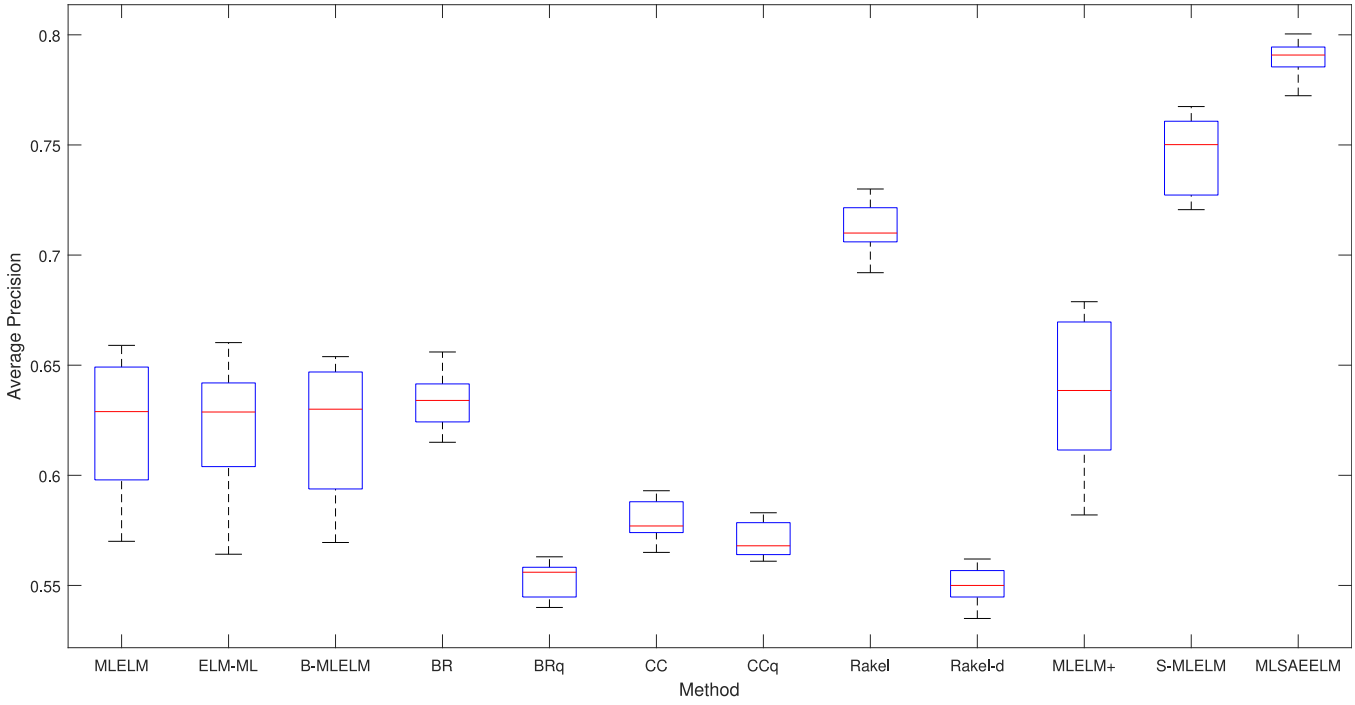

Fig. 4. Average Precision of all algorithms over k-fold cross-validation for Emotions dataset.

takes the original input $\mathbf{X_i} = \{x_1, x_2, \ldots, x_d\}$, trains itself iteratively and encodes the data to a smaller number of features (say, $\mathbf{a_i} = \{a_1, a_2, \ldots, a_{d'}\}$). These encoded features are then passed on to train the next autoencoder AE_2 and the features are further encoded to $\mathbf{c}$ which has d'' number of features and so on. The final set of encoded features from the autoencoder AE_n is used as input in the next phase. In Fig. 3, only two layers have been shown in the stacked autoencoder.

4.2.2. Multi-label ELM for soft classification (MLELM-C)

In this phase, a simple MLELM network (referred here as MLELM-C) is used, which is basically an ELM network performing multi-label classification. The encoded features obtained from the previous stacked autoencoder phase is taken as input and the class labels for the multi-label data are provided as output. Thus, the MLELM-C maps d'' number of input nodes to C output nodes with h hidden nodes. The hidden layer output vector $\mathbf{Z} = \{z_1, z_2, .., z_h\}$ is computed using Eq. (3). The MLELM-C model can thus be compactly represented as

$$\mathbf{Y} = \mathbf{Z}\beta, \tag{7}$$

where, $\mathbf{Y} = \{y_1, y_2, \ldots, y_C\}$ is the actual output vector and β is the $h \times C$ weight matrix between the hidden layer to the output layer. β is then determined using Eq. (6).

A large number of hidden nodes are required in comparison to the number of input nodes for the MLELM-C to be able to learn efficiently from the data. Since we have reduced the number of features of the input data, the input layer is not as large as a stand-alone MLELM. Thus, the hidden layer does not need to be extremely large and the weight matrix can be approximated comfortably. Once the weight matrix β has been approximated, the soft label scores at the kth output node can be computed as

$$y'_k = \sum_{j=1}^{h} z_j \beta_{jk}, \tag{8}$$

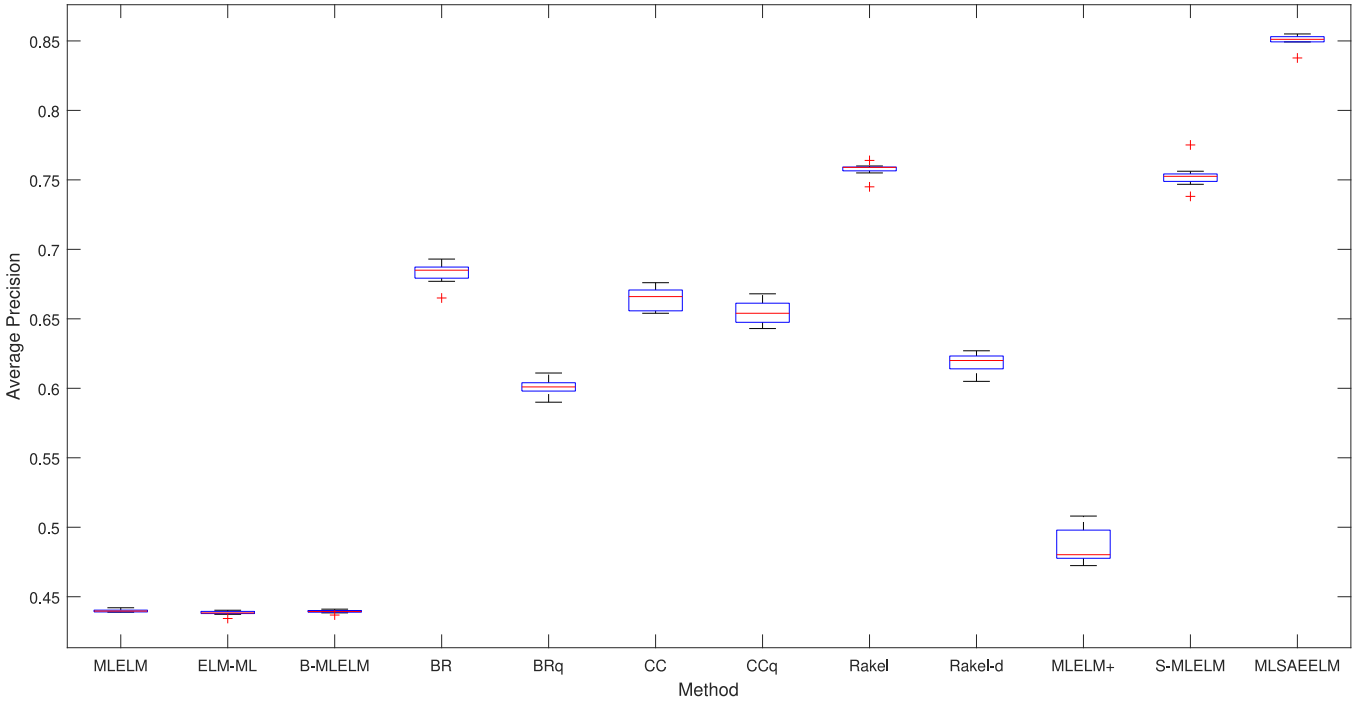

Fig. 5. Average Precision of all algorithms over k-fold cross-validation for Scene dataset.

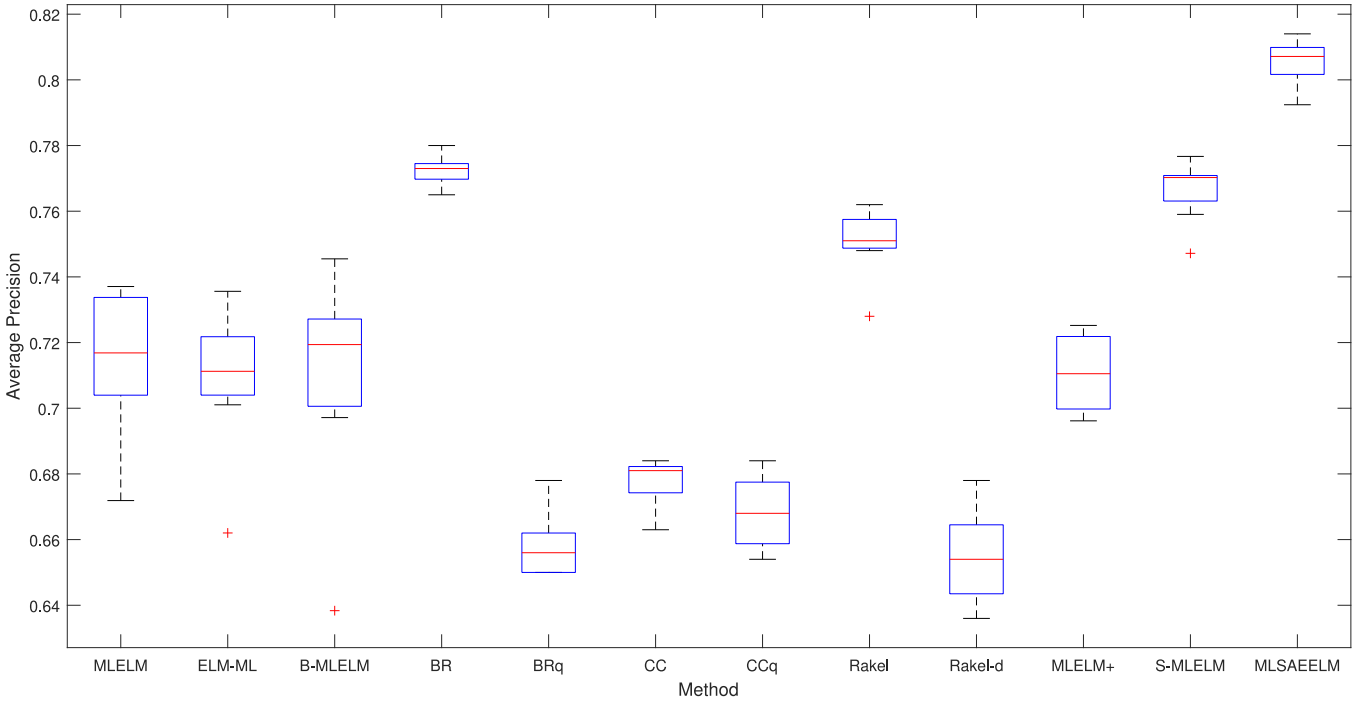

Fig. 6. Average Precision of all algorithms over k-fold cross-validation for Flags dataset.

where, β_{jk} is the weight from hidden node z_j to output node y_k. Each node in the output layer now contains the soft classification score for that particular class. These scores are usually converted to hard labels in a regular MLELM using a threshold. In our proposed method, we use these soft classification scores in the next phase.

4.2.3. Multi-label ELM for score mapping (MLELM-D)

The output generated from the MLELM-C is a set of scores, one for each class. For single-label data, the class label corresponding to the highest value is assigned to the sample. While labeling multi-label data, several class labels may be assigned to one sample depending on the obtained score. The general method of determining the hard multi-labels is by setting a threshold. If the predicted value is higher than the threshold, the class is relevant, hence label is 1, else label is 0 for the irrelevant class. This threshold selection is very crucial for multi-label data. A very low threshold can assign more labels than required and a very high threshold might end up under assigning labels to the data instances, both eventually leading to misclassification. Therefore, it is important that the model can predict correct hard labels from the scores

Table 1
Comparative results for Emotions dataset.

	H Loss ↓	R Loss ↓	One Err ↓	H Score ↑	S Acc ↑	Avg Pre ↑	Macro F1 ↑	Micro F1 ↑	Acc ↑
MLELM	0.4138	0.4039	0.5678	0.5862	0.3960	0.6037	1.3014	1.3009	0.3024
B-MLELM	0.4034	0.3943	0.5882	0.5966	0.4272	0.5996	1.3306	1.3230	0.3335
ELM-ML	0.4774	0.4683	0.6186	0.5226	0.3757	0.5475	1.2518	1.2500	0.2822
MLELM+	0.3655	0.3601	0.5042	0.6345	0.4541	0.6385	1.3265	1.3188	0.3545
S-MLELM	0.2792	0.2608	0.3833	0.7208	0.5267	0.7225	1.3972	1.3952	0.4360
BR	0.2640	0.3270	0.4470	0.7360	0.1720	0.6190	0.5570	0.5650	0.4220
BRq	0.2830	0.3040	0.4820	0.7170	0.1550	0.5400	0.5770	0.5830	0.4460
CC	0.2610	0.2940	0.4200	0.7390	0.2210	0.5790	0.5730	0.5850	0.4670
CCq	0.2780	0.2970	0.4470	0.7220	0.2140	0.5680	0.5670	0.5740	0.4590
Rakel	0.3450	0.2230	0.3170	0.6550	0.0930	0.7100	0.6100	0.6120	0.4710
Rakel-d	0.2610	0.3060	0.4590	0.7390	0.1650	0.5560	0.5660	0.5710	0.4330
MLSAEELM	**0.1852**	**0.1580**	**0.2735**	**0.8148**	**0.5695**	**0.8065**	**1.4489**	**1.4463**	**0.4957**

obtained from the classifier. To improve the prediction ability of the model, we propose to use a modified MLELM in the third phase that learns the mapping of the scores predicted by MLELM-C to the final class labels. This MLELM-D takes the predicted output values $\mathbf{Y}'$ from the MLELM-C as input and the original target labels $\mathbf{Y}$ as output. The input weights and biases are randomly initialized and the output at the hidden layer V is computed using Eq. (3). The hidden weights $\boldsymbol{\alpha}$ are similarly learned from $\mathbf{V}$ and the original output labels $\mathbf{Y}$ as

$$\alpha = \mathbf{V}^{\dagger}\mathbf{Y}. \tag{9}$$

This MLELM-D helps to map the soft scores predicted by the previous network to the hard labels instead of using a specific threshold like in traditional ELMs. The final predictions made by this MLELM-D network is then converted to hard class labels using a calibrated threshold.

4.3. Training phase

In this phase, the training data is passed through three networks sequentially. Each of these networks have a specific task and is trained iteratively.

4.3.1. Feature extraction

At the beginning of the training phase, the input data is fed to the stacked autoencoder to perform feature extraction. The reduced number of features and the number of layers in the autoencoder is predetermined. The SAE trains iteratively till it learns the input thoroughly. Once the SAE has learnt from the training data, the encoded features are obtained from the network. The extracted features of the training data are then passed on to the second network.

4.3.2. Soft class prediction

The multi-label ELM allocated for soft class prediction (MLELM-C) uses the reduced features from the previous network as input, and the original multi-label output in the output layer. The number of nodes in the hidden layer is determined depending on the number of input nodes. This network works in batch mode, where it takes all the input instances together and trains itself in one pass. Once the MLELM-C has learned the hidden layer weights, the outputs need to be predicted. The MLELM-C is again fed with the features encoded training data $\mathbf{c}$, only to generate class scores. The predicted score vector $\mathbf{Y}'$ at the output layer are calculated as

$$\mathbf{Y}' = \beta\phi(\gamma\mathbf{c} + \mathbf{B}), \tag{10}$$

where, $\boldsymbol{\gamma}$ is the input-hidden weight matrix, $\mathbf{B}$ is the bias vector, ϕ is the activation function and β is the hidden-output weight matrix computed by the MLELM-C. The predicted outputs $\mathbf{Y}'$ that is obtained from the MLELM-C are used to train the third network MLELM-D.

4.3.3. Class score approximation

Another MLELM (named MLELM-D) is used to improve the predictions of MLELM-C by mapping the class scores to actual class labels. The predicted values computed from MLELM-C are provided as input to the MLELM-D and the original class labels are used as output. This MLELM-D network also learns the hidden layer weights in one pass.

4.4. Testing phase

Once the complete multi-label stacked encoder and ELM model (MLSAEELM) are trained, it is ready for testing. In this phase, the test data is fed individually to the first SAE network. The SAE generates a set of encoded features for the test data in an unsupervised manner. The reduced input features are then passed on to the second phase. In the soft classification step, the trained MLELM-C computes the individual class scores for each of the test patterns. These intermediate predicted values are then given as input to the final network MLELM-D which then maps the soft class scores to actual class labels. This third network outputs the final predicted values which are then used to determine the hard class labels for the test samples.

5. Results and discussion

To evaluate the effectiveness of the proposed MLSAEELM technique, it has been tested on seven multi-label datasets using ten performance metrics. These outcomes have been compared with eleven other relevant techniques to judge the overall performance of the proposed model.

5.1. Experimental setup

Seven well-known multi-label datasets have been used to evaluate the performance of the proposed model. These datasets are emotions (music) [39], scene (image) [40], flags (image) [41], slashdot (text), yeast (biology) [5], delicious (web text) [42] and EUR-Lex (text) [43]. The datasets are openly available at http://mulan.sourceforge.net/datasets-mlc.html and http://waikato.github.io/meka/datasets/. Experiments have been done using MATLAB 2017a on a Windows OS with Intel Core i7 processor and 16 GB RAM.

Ten performance measuring indices [3] have been computed with k-fold cross-validation on the above datasets. These metrics are namely, Hamming loss (*H loss*), ranking loss (*R Loss*), one error (*One Err*), Hamming score (*H Score*), subset accuracy (*S Acc*), average precision (*Avg Pre*), macro-F1, micro-F1, overall accuracy (*Acc*) and average recall. Among these *H loss, H score, S Acc, Acc* and average recall are example-based metrics, *R Loss, One Err* and *Avg Pre* are ranking-based metrics, and macro F1, micro F1 are label-based metrics.

Table 2
Comparative results for Scene dataset.

	H Loss ↓	R Loss ↓	One Err ↓	H Score ↑	S Acc ↑	Avg Pre ↑	Macro F1 ↑	Micro F1 ↑	Acc ↑
MLELM	0.1819	0.4699	0.7713	0.8181	0.0014	0.4430	1.6362	1.6362	0.0010
B-MLELM	0.1778	0.4676	0.7672	0.8222	0.0010	0.4471	1.6450	1.6451	0.0007
ELM-ML	0.2763	0.4722	0.7905	0.7237	0.1916	0.4339	1.5441	1.6020	0.1815
MLELM+	0.1753	0.3895	0.7469	0.8247	0.0145	0.4966	1.6517	1.6517	0.0135
S-MLELM	0.1459	0.1340	0.3410	0.8541	0.5773	0.7892	1.6518	1.6533	0.5442
BR	0.1350	0.2320	0.3890	0.8650	0.4230	0.6870	0.6350	0.6240	0.5340
BRq	0.1440	0.2070	0.3930	0.8560	0.4020	0.6010	0.6230	0.6370	0.5410
CC	0.1440	0.2200	0.3810	0.8560	0.5310	0.6540	0.6160	0.6020	0.5840
CCq	0.1430	0.2110	0.3650	0.8570	0.5290	0.6590	0.6230	0.6120	0.5930
Rakel	0.2210	0.1510	0.3160	0.7790	0.2290	0.7450	0.6050	0.5860	0.5140
Rakel-d	0.1440	0.2260	0.4080	0.8560	0.4470	0.6050	0.6050	0.5960	0.5320
MLSAEELM	**0.0918**	**0.0760**	**0.2225**	**0.9082**	**0.6126**	**0.8677**	**1.6704**	**1.6710**	**0.6019**

Table 3
Comparative results for Flags dataset.

	H Loss ↓	R Loss ↓	One Err ↓	H Score ↑	S Acc ↑	Avg Pre ↑	Macro F1 ↑	Micro F1 ↑	Acc ↑
MLELM	0.4029	0.4162	0.4615	0.5971	0.5242	0.6598	1.1704	1.1203	0.4045
B-MLELM	0.3985	0.3754	0.3158	0.6015	0.5889	0.7227	1.1314	1.0959	0.4504
ELM-ML	0.5201	0.5256	0.6410	0.4799	0.4476	0.5877	1.0459	0.9813	0.3226
MLELM+	0.3666	0.3537	0.3493	0.6334	0.5846	0.7250	1.1732	1.1338	0.4734
S-MLELM	0.3679	0.3013	0.2051	0.6321	0.5807	0.7671	1.1502	1.1274	0.4474
BR	**0.2530**	0.2690	0.3090	**0.7470**	0.1750	0.7720	0.6690	0.7470	0.6060
BRq	0.2540	0.2720	0.2110	0.7460	0.1650	0.6710	0.7080	0.7660	**0.6270**
CC	0.2720	0.2940	0.2320	0.7280	0.2580	0.6800	0.6570	0.7220	0.5860
CCq	0.2650	0.2960	0.2110	0.7350	0.1600	0.6680	0.6800	0.7500	0.6120
Rakel	0.2830	0.2670	0.2680	0.7170	0.1490	0.7490	0.6960	0.7490	0.6070
Rakel-d	0.2750	0.3140	0.2320	0.7250	0.2110	0.6780	0.6480	0.7210	0.5860
MLSAEELM	0.2612	**0.1895**	**0.1429**	0.7388	**0.6758**	**0.8420**	**1.2442**	**1.2027**	0.5503

Table 4
Comparative results for Slashdot dataset.

	H Loss ↓	R Loss ↓	One Err ↓	H Score ↑	S Acc ↑	Avg Pre ↑	Macro F1 ↑	Micro F1 ↑	Acc ↑
MLELM	0.1803	0.2995	0.7530	0.8197	0.2222	0.3855	1.6882	1.8771	0.1571
B-MLELM	0.1750	0.3233	0.7701	0.8250	0.2235	0.3713	1.7361	1.8783	0.1605
ELM-ML	0.1347	0.2308	0.6640	0.8653	0.2837	0.4770	1.7442	1.8833	0.2182
MLELM+	0.2092	0.3269	0.7734	0.7908	0.1994	0.3658	1.7315	1.8729	0.1410
S-MLELM	0.0738	0.2165	0.5854	0.9262	0.3143	0.5302	1.7992	1.8905	0.2876
BR	0.0430	0.1440	0.5080	0.9570	0.2990	0.5950	0.2350	0.4600	0.3460
BRq	0.0450	0.3000	0.6170	0.9550	0.2880	0.3820	0.2490	0.4590	0.3480
CC	0.0560	0.3010	0.5390	0.9440	0.3630	0.4540	0.2490	0.4430	**0.4230**
CCq	0.0590	0.2970	0.5660	0.9410	0.3450	0.4290	0.2530	0.4170	0.3970
Rakel	0.0490	0.3980	0.8070	0.9510	0.1290	0.2000	0.1490	0.2540	0.1540
Rakel-d	0.0420	0.3030	0.6000	0.9580	0.3110	0.3920	0.2500	0.4730	0.3570
MLSAEELM	**0.0412**	**0.1078**	**0.4276**	**0.9588**	**0.3748**	**0.6768**	**1.8048**	**1.8940**	0.3593

Table 5
Comparative results for Yeast dataset.

	H Loss ↓	R Loss ↓	One Err ↓	H Score ↑	S Acc ↑	Avg Pre ↑	Macro F1 ↑	Micro F1 ↑	Acc ↑
MLELM	0.2177	0.1938	0.2541	0.7823	0.5855	0.7419	1.4664	1.4317	0.4751
B-MLELM	0.2178	0.1885	0.2464	0.7822	0.5709	0.7412	1.4743	1.4427	0.4595
ELM-ML	0.2228	0.1782	0.2583	0.7772	**0.6442**	0.7424	1.4790	1.4427	**0.5292**
MLELM+	0.2091	0.1875	0.2611	0.7909	0.5942	0.7429	1.4817	1.4465	0.4891
S-MLELM	0.2129	0.1838	0.2333	0.7871	0.5900	0.7480	1.4804	1.4459	0.4789
BR	0.2510	0.3150	0.4380	0.7490	0.0620	0.5820	0.3900	0.5680	0.4200
BRq	0.2610	0.2660	0.4940	0.7390	0.0620	0.5310	0.4190	0.6120	0.4690
CC	0.2660	0.3000	0.4820	0.7340	0.1440	0.5330	0.3970	0.5530	0.4250
CCq	0.2740	0.2830	0.5330	0.7260	0.1250	0.5320	0.4100	0.5890	0.4600
Rakel	0.3200	0.2860	0.4140	0.6800	0.0480	0.5510	0.4240	0.5690	0.4250
Rakel-d	0.2740	0.3120	0.4820	0.7260	0.0640	0.4950	0.3880	0.5450	0.3980
MLSAEELM	**0.1943**	**0.1648**	**0.2222**	**0.8057**	0.6104	**0.7649**	**1.4888**	**1.4533**	0.5021

The proposed model has been compared with eleven related models. The first method of comparison is a multi-label adaptation of the original ELM (MLELM) with a global threshold of 0.5. A bipolar MLELM [15] (B-MLELM) and ELM-ML [31] have been included for comparison. Among non-ELM methods, binary relevance (BR) [36], classifier chains (CC) [37] and random k label-sets (RAkEL) [38] along with one of their modifications (BRq, CCq and RAkEL-d) have been used for comparison. Apart from these existing techniques, partial structures of the proposed method have also been included for evaluation. The method referred to as S-MLELM includes only the first and second phase of the proposed model, i.e., an SAE followed by MLELM-C. Comparison with this model shows the utility of the MLELM-D network of the proposed work. Another model MLELM+ which contains the second and the third phase

Table 6
Comparative results for Delicious dataset.

	H Loss ↓	R Loss ↓	One Err ↓	H Score ↑	S Acc ↑	Avg Pre ↑	Macro F1 ↑	Micro F1 ↑	Acc ↑
MLELM	0.0207	0.2073	0.4915	0.9793	0.1552	0.2663	1.9609	1.9610	0.0998
B-MLELM	0.0204	0.2074	0.4845	0.9796	0.1532	0.2628	1.9613	1.9615	0.0985
ELM-ML	0.0234	0.1588	0.4519	0.9766	**0.2629**	0.2812	1.9597	1.9611	0.1840
MLELM+	0.1117	0.2765	0.5652	0.8883	0.1276	0.2210	1.9572	1.9573	0.1451
S-MLELM	0.0183	**0.1392**	0.3607	0.9817	0.1747	0.3476	1.9618	1.9614	0.1672
BR	0.0182	0.8824	0.3414	0.9818	0.0118	0.2681	0.0470	0.1933	0.1089
BRq	0.0215	0.8654	0.3541	0.9785	0.0109	0.2542	0.0460	0.1857	0.1310
CC	0.0187	0.1780	0.3059	0.9813	0.0255	0.1641	0.1233	0.2570	0.1467
CCq	0.0209	0.1654	0.3373	0.9791	0.0256	0.1746	0.1145	0.2511	0.1358
Rakel	0.0182	0.2002	0.3426	0.9818	0.0261	0.1881	0.1013	0.2482	0.1429
Rakel-d	0.0199	0.1823	0.3456	0.9801	0.0271	0.1785	0.1162	0.2261	0.1265
MLSAELM	**0.0147**	0.1499	**0.3075**	**0.9853**	0.1855	**0.3580**	**1.9691**	**1.9617**	**0.1881**

Table 7
Comparative results for EUR-Lex dataset.

	H Loss ↓	R Loss ↓	One Err ↓	H Score ↑	S Acc ↑	Avg Pre ↑	Macro F1 ↑	Micro F1 ↑	Acc ↑
MLELM	0.0232	0.0886	0.4007	0.9768	0.4267	0.6072	1.8431	1.9778	0.4135
B-MLELM	0.0186	0.0450	0.2027	0.9814	**0.5340**	0.7763	1.8688	1.9779	0.4703
ELM-ML	0.0109	0.0397	0.2412	0.9891	0.2108	0.7530	1.8588	**1.9782**	0.4582
MLELM+	0.0215	0.1112	0.4390	0.9785	0.4009	0.5617	1.8729	1.9777	0.3439
S-MLELM	0.0110	0.1319	0.1778	0.9890	0.3862	0.2211	1.8686	1.9780	0.5162
BR	0.0153	0.9478	0.2200	0.9847	0.2701	0.7631	0.1953	0.4417	0.4515
BRq	0.0202	0.8751	0.2118	0.9798	0.2618	0.7281	0.1879	0.4355	0.4987
CC	0.0050	0.7735	0.1937	0.9950	0.4796	0.7254	0.4908	0.7637	**0.5986**
CCq	0.0166	0.7543	0.2032	0.9834	0.4511	0.7203	0.5362	0.6982	0.5672
Rakel	0.0046	0.8107	0.1854	0.9954	0.4982	0.7747	0.5114	0.7810	0.5118
Rakel-d	0.0199	0.7968	0.1955	0.9801	0.5012	0.7683	0.5051	0.7554	0.5098
MLSAELM	**0.0031**	**0.0295**	**0.1677**	**0.9969**	0.4865	**0.7773**	**1.8785**	1.9780	0.5208

Table 8
Average recall for all datasets.

	Scene	Flags	Slashdot	Yeast	Emotions	Delicious	EUR-Lex
MLELM	0.0211	0.5406	0.4714	0.5568	0.4774	0.1246	0.5995
B-MLELM	0.0251	0.5921	0.4590	0.5462	0.5322	0.1253	0.5868
ELM-ML	0.1826	0.4547	0.5007	0.7203	**0.4972**	0.2388	0.0000
MLELM+	0.0166	0.5444	0.4357	0.5588	0.4552	0.2204	0.5086
S-MLELM	0.6112	**0.5682**	0.3603	0.5760	0.5625	0.1211	0.6154
BR	0.5426	0.1862	0.2891	0.0632	0.1654	0.1218	0.3562
BRq	0.3981	0.1655	0.2677	0.0671	0.1435	0.1209	0.3721
CC	0.5571	0.2351	0.3491	0.1231	0.2091	0.0925	0.4567
CCq	0.4872	0.1462	0.3167	0.1253	0.1879	0.0875	0.4435
Rakel	0.2765	0.1311	0.1472	0.0433	0.0879	0.0926	0.4873
Rakel-d	0.4367	0.1892	0.3418	0.0598	0.1376	0.1013	0.4967
MLSAELM	**0.6351**	0.6248	**0.3767**	**0.5707**	0.5598	**0.2242**	**0.6252**

of the proposed model (MLELM-C + MLELM-D) has been used for comparison to show the importance of using an SAE. The detailed analysis of the results are given in the following section.

5.2. Analysis of results

The proposed classification model (represented as MLSAEELM) has been compared with eleven other relevant multi-label classification methods (including some ELM based techniques) from literature. The 5-fold cross-validation results corresponding to these algorithms for seven datasets and ten performance measures are shown below. Tables 1–7 show the results for emotions, scene, flags, slashdot, yeast, delicious and EUR-lex datasets respectively for nine measures and Table 8 represents the average recall measure for all datasets.

Analysing the performance measures for all the methods in the above tables, it is seen that they vary in a similar manner for all the datasets across all the metrics. The algorithms MLELM and B-MLELM, both solo networks, are seen to perform quite close to each other for all the datasets, whereas their performance is quite low compared to other algorithms. Thus, it can be seen that using a stand-alone MLELM and its variations is unable to predict multi-label classes well. The single network is not sufficient for learning the intricacies that are present in the data.

Table 9
T-test statistics for all the algorithms against the proposed one.

Proposed v/s existing method	*T*-test value
MLELM	4.1996
B-MLELM	3.0272
ELM-ML	3.5019
MLELM+	4.7244
S-MLELM	1.6514
BR	4.6123
BRq	5.7343
CC	4.3426
CCq	4.7881
Rakel	2.4001
Rakel-d	4.5857

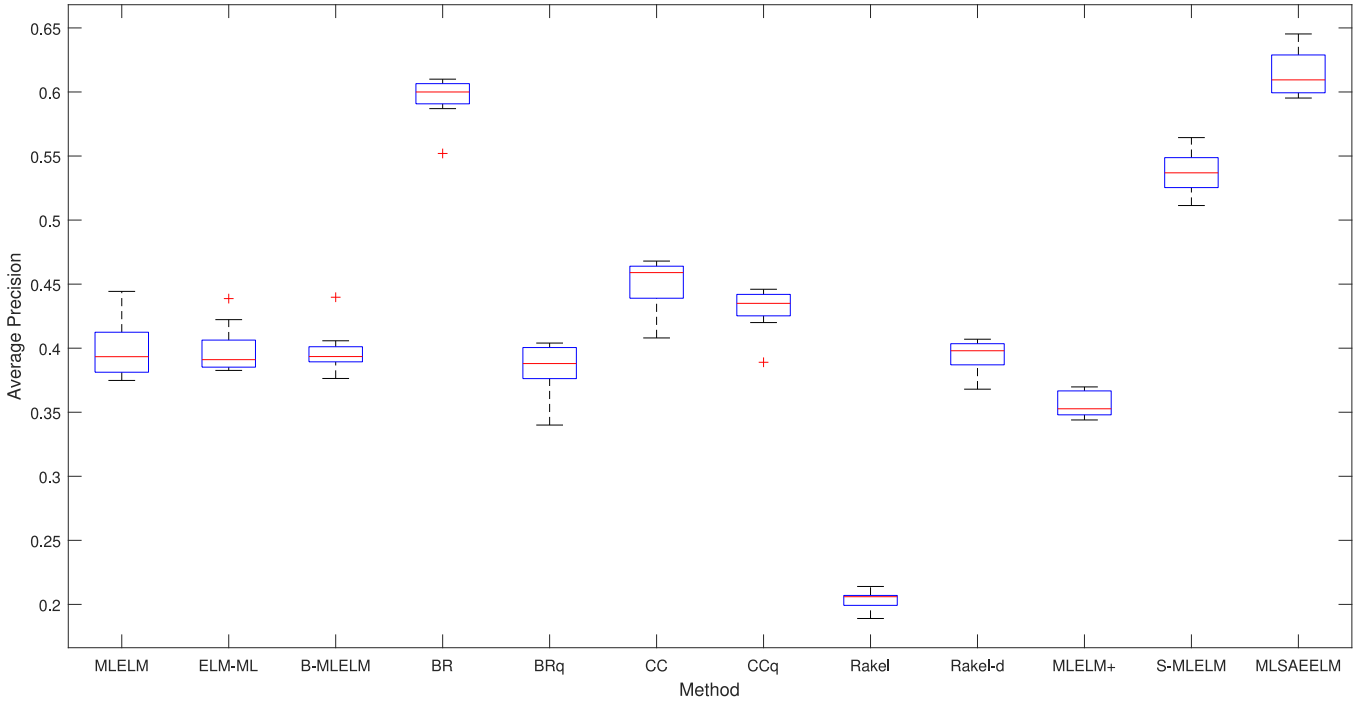

Fig. 7. Average Precision of all algorithms over k-fold cross-validation for Slashdot dataset.

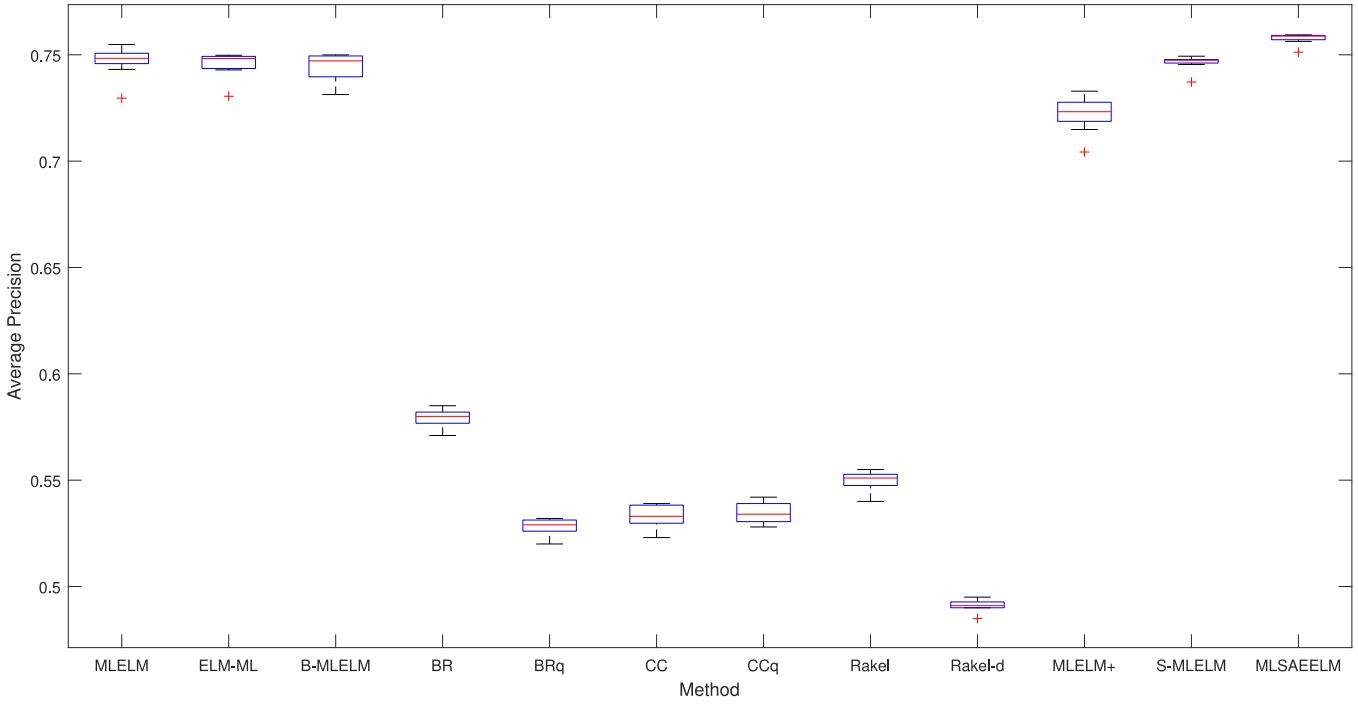

Fig. 8. Average Precision of all algorithms over k-fold cross-validation for Yeast dataset.

ELM-ML and MLELM+ both use two consecutive networks instead of one MLELM, and their performance is better than that of the single MLELMs in most of the cases. ELM-ML uses the second network for threshold approximation, whereas MLELM+ uses the second MLELM to map class score (MLELM-D). Among these two, MLELM+ seems to perform better than ELM-ML, which indicates the benefit of using the proposed MLELM-D network. Among MLELM, MLELM+, S-MLELM and MLSAEELM it is seen that the proposed model has significant improvement in performance by using SAE for feature encoding and MLELM-D for class score approximation. The proposed method is also seen to perform better than the other existing methods BR, BRq, CC, CCq, RAkEL and RAkEL-d. These results have been further confirmed by testing MLSAEELM against these methods by varying the amount of training data as shown in Figs. 4–10. These box plots show the average precision of the methods for all the datasets over k-fold cross-validation, ranging k from 2 to 10.

Among all the methods used for comparison, the proposed approach is seen to perform significantly better over the existing ones for all the seven datasets and all the performance metrics.

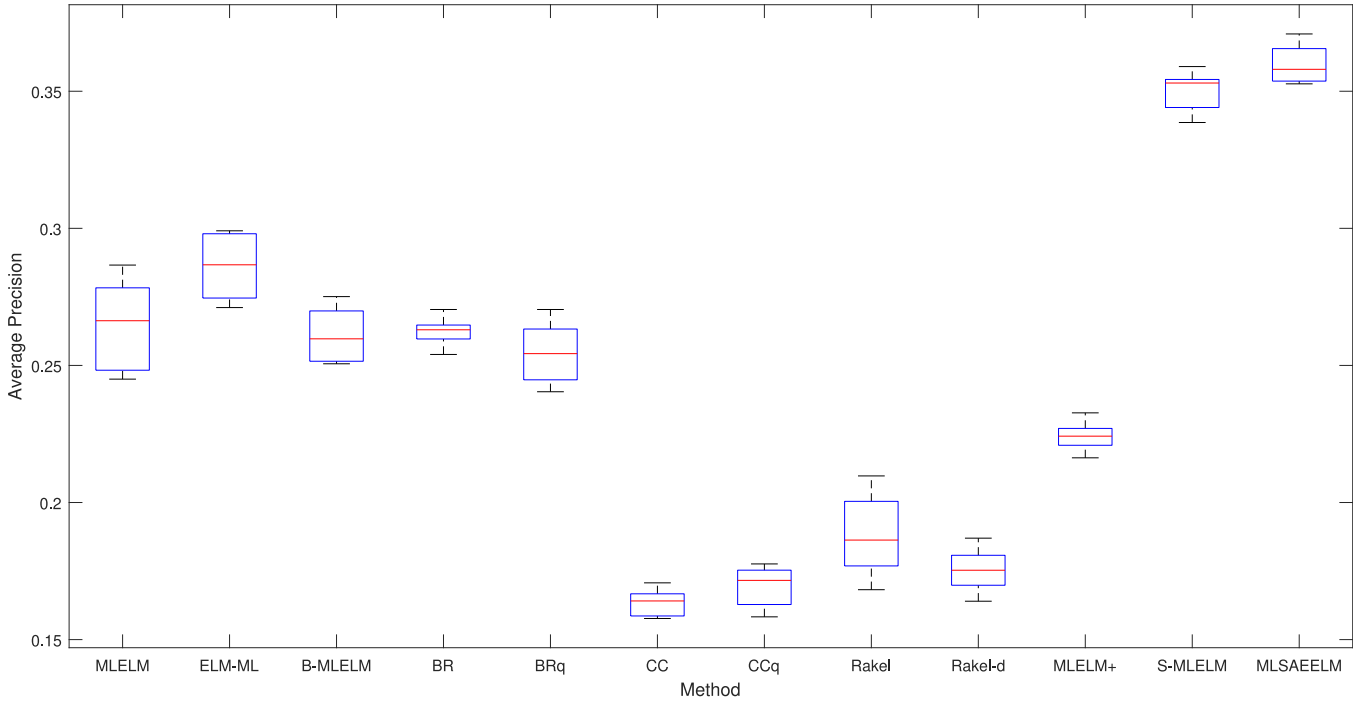

Fig. 9. Average Precision of all algorithms over k-fold cross-validation for Delicious dataset.

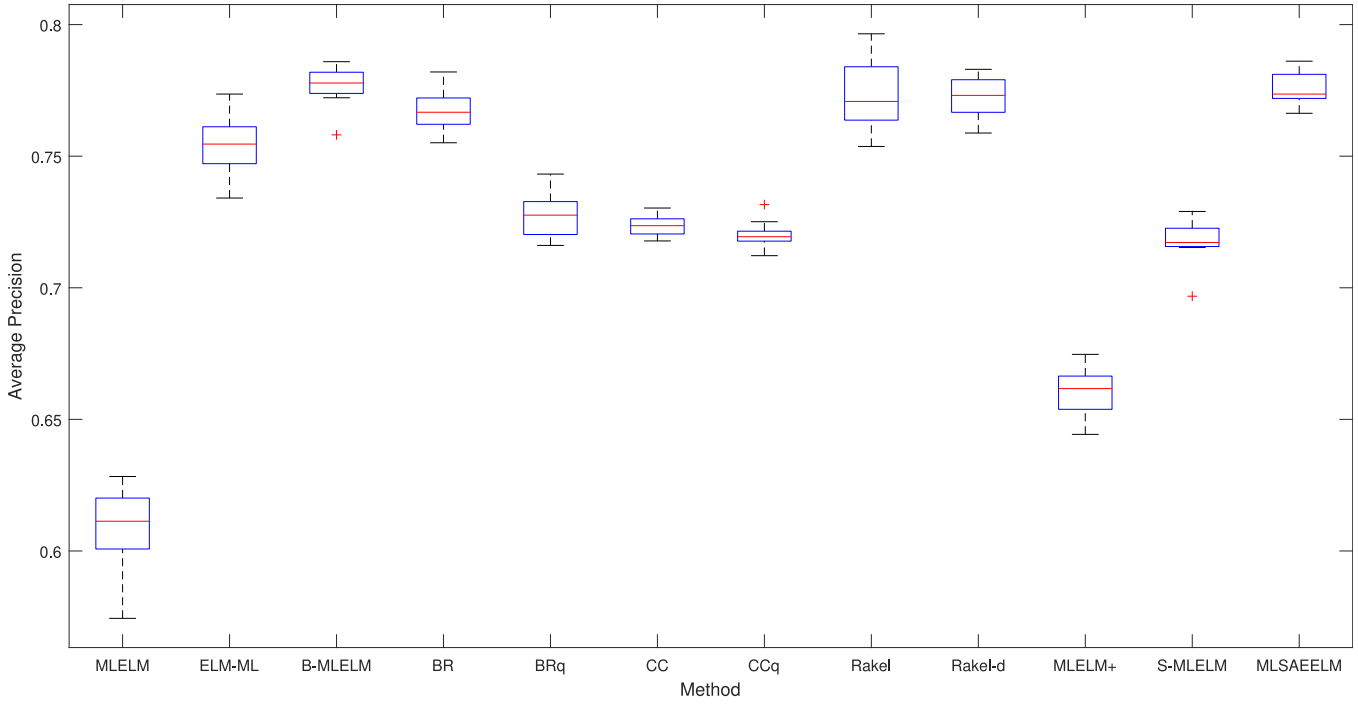

Fig. 10. Average Precision of all algorithms over k-fold cross-validation for EUR-Lex dataset.

T-test statistics on the proposed approach against all the compared methods for average precision metric is shown in Table 9. For $t_{0.90} = 1.440$ with degrees of freedom $= 6$, the proposed method MLSAEELM outperforms all the other models.

Computational complexity of the proposed algorithm has been computed experimentally and compared with the other existing methods and shown in Table 10. Both the training times and testing times have been recorded separately. It is seen that the training time for the proposed algorithm is higher compared to the other algorithms, especially for the larger datasets. This is mainly due to the stacked autoencoder network which is iteratively trained in the proposed algorithm, thus adding on some extra amount of training time. Due to this reason, S-MLELM also has a large training time. Since, the training phase is performed offline, a larger training time does not quite affect the actual testing speed of the algorithm. From the recorded testing times, it can be seen that the proposed method is quite fast and its speed is comparable to the rest of the methods. Moreover, its testing time for some datasets

Table 10
Run-time (in seconds) of all the algorithms for all datasets.

		Emotions	Scene	Flags	Slashdot	Yeast	EUR-Lex	Delicious
MLELM	*Train*	0.1749	1.1105	0.0124	2.1116	1.5371	161.9306	215.9715
	Test	0.0011	0.0088	0.0018	0.0629	0.0221	3.8547	0.1599
B-MLELM	*Train*	0.1778	1.1031	0.0194	2.1183	1.4245	181.0913	234.7683
	Test	0.0013	0.0079	0.0011	0.0672	0.0034	3.7205	0.1713
ELM-ML	*Train*	0.1328	4.2526	0.0183	66.7521	1.6929	936.4543	755.9038
	Test	0.0035	0.0416	0.0028	0.6456	0.0179	0.4021	0.3654
MLELM+	*Train*	0.0632	2.0992	0.0163	44.4069	0.7496	167.9387	405.1314
	Test	0.0037	0.0242	0.0027	0.2026	0.0131	0.2066	0.1894
S-MLELM	*Train*	2.3339	24.9166	1.2771	338.0252	7.4861	12856.6601	528.0802
	Test	0.0852	0.0637	0.0596	0.0763	0.0609	0.5123	0.0833
BR	*Train*	0.3346	0.6719	0.3063	0.6518	0.3785	47.5633	40.1136
	Test	0.0151	0.0417	0.0405	0.1096	0.1362	3.7268	30.9803
BRq	*Train*	0.3202	0.6551	0.2917	0.6265	0.3905	49.1082	40.1021
	Test	0.0145	0.0322	0.0336	0.1077	0.1159	3.0465	29.5411
CC	*Train*	0.3622	0.9818	0.3211	1.4761	0.7788	973.2005	40.2038
	Test	0.0180	0.0348	0.0154	0.1085	0.0752	4.1138	33.1692
CCq	*Train*	0.3396	0.9481	0.3202	1.4655	0.7791	864.5708	39.2140
	Test	0.0154	0.0301	0.0132	0.0998	0.0663	4.0213	32.9651
Rakel	*Train*	0.6254	4.4748	0.3752	1.3193	4.0614	6467.4840	347.1959
	Test	0.0211	0.1389	0.0185	0.1424	0.0872	3.9647	31.6542
Rakel-d	*Train*	0.5889	4.4734	0.3776	1.3330	4.0544	6542.3890	347.9101
	Test	0.0204	0.1134	0.0166	0.1326	0.0803	3.7465	30.2645
MLSAEELM	*Train*	2.5726	25.7914	1.2392	334.4661	8.2951	18353.8901	677.5598
	Test	0.0801	0.0592	0.0575	0.0876	0.0741	0.8718	0.1609

especially the larger ones are faster than most of the other compared algorithms.

Thus, from all the results obtained, it can be said that the application of a stacked encoder for feature extraction and additionally learning the soft class scores to hard labels mapping in the proposed method significantly improves the performance of the MLELM compared to other algorithms.

6. Conclusion

This article proposes a novel stacked autoencoder and extreme learning machine network for multi-label classification (MLSAEELM). ELM is a compact neural network which learns from the training data in one-pass. Applications of this network has been made in various domains, but its use is limited in the field of multi-label classification. The network is innovative and unique but it is not able to handle multi-label data efficiently on its own. To explore the possibilities in the area of multi-label classification, by utilizing the strengths and coping with the challenges faced by ELM, an MLSAEELM model has been proposed here. It uses stacked autoencoders for feature encoding, MLELM for soft classification and in a novel class score approximation method, which finally results in multi-label classification. Comparative analysis of the proposed algorithm on seven datasets with ten performance metrics against eleven other algorithms has shown encouraging performance improvement. In future, this cascade of networks can be adapted to a deeper framework to facilitate working with larger and more complex datasets.

Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work of Anwesha Law was supported by the Indian Statistical Institute, India. The authors would also like to thank the reviewers for their valuable insight which has helped in enhancing the article.

References

[1] C.M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer-Verlag, Berlin, Heidelberg, 2006.
[2] S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice-Hall of India, New Delhi, 2008.
[3] F. Herrera, F. Charte, A.J. Rivera, M.J. Del Jesus, Multilabel Classification: Problem Analysis, Metrics and Techniques, Springer, 2016.
[4] M.L. Zhang, Z.H. Zhou, Multilabel neural networks with applications to functional genomics and text categorization, IEEE Trans. Knowl. Data Eng. 18 (10) (2006) 1338–1351.
[5] A. Elisseeff, J. Weston, A kernel method for multi-labelled classification, in: Proceedings of Advances in Neural Information Processing Systems, 2002, pp. 681–687.
[6] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, H. Blockeel, Decision trees for hierarchical multi-label classification, Mach. Learn. 73 (2) (2008) 185.
[7] M.L. Zhang, ML-RBF: RBF neural networks for multi-label learning, Neural Process. Lett. 29 (2) (2009) 61–74.
[8] Y. Kongsorot, P. Horata, Multi-label classification with extreme learning machine, in: Proceedings of the 6th International Conference on Knowledge and Smart Technology (KST), IEEE, 2014, pp. 81–86.
[9] A. Law, K. Chakraborty, A. Ghosh, Functional link artificial neural network for multi-label classification, in: Proceedings of International Conference on Mining Intelligence and Knowledge Exploration, Springer, 2017, pp. 1–10.
[10] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
[11] A. Mondal, S. Ghosh, A. Ghosh, Partially camouflaged object tracking using modified probabilistic neural network and fuzzy energy based active contour, Int. J. Comput. Vis. 122 (1) (2017) 116–148.
[12] B.N. Subudhi, S. Ghosh, A. Ghosh, Application of Gibbs–Markov random field and Hopfield-type neural networks for detecting moving objects from video sequences captured by static camera, Soft Comput. 19 (10) (2015) 2769–2781.
[13] F. Wu, Z. Wang, Z. Zhang, Y. Yang, J. Luo, W. Zhu, Y. Zhuang, Weakly semi-supervised deep learning for multi-label image annotation, IEEE Trans. Big Data 1 (3) (2015) 109–122.
[14] G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: Proceedings of the 2004 IEEE International Joint Conference on Neural Networks, 2, IEEE, 2004, pp. 985–990.
[15] R. Venkatesan, M.J. Er, Multi-label classification method based on extreme learning machines, in: Proceedings of the 13th International Conference on Control Automation Robotics & Vision (ICARCV), IEEE, 2014, pp. 619–624.
[16] M. Jiang, Z. Pan, N. Li, Multi-label text categorization using L21-norm minimization extreme learning machine, Neurocomputing 261 (2017) 4–10.
[17] N. Zhang, S. Ding, J. Zhang, Multi layer ELM-RBF for multi-label learning, Appl. Soft Comput. 43 (2016) 535–545.
[18] I. Guyon, A. Elisseeff, An Introduction to Feature Extraction, Springer, Berlin, Heidelberg, pp. 1–25.
[19] E. Binaghi, P.A. Brivio, P. Ghezzi, A. Rampini, A fuzzy set-based accuracy assessment of soft classification, Pattern Recognit. Lett. 20 (9) (1999) 935–948.
[20] D.H. Wolpert, Stacked generalization, Neural Netw. 5 (2) (1992) 241–259.

[21] C.Y. Liou, W.C. Cheng, J.W. Liou, D.R. Liou, Autoencoder for words, Neurocomputing 139 (2014) 84–96.
[22] C.Y. Liou, J.C. Huang, W.C. Yang, Modeling word perception using the Elman network, Neurocomputing 71 (16–18) (2008) 3150–3157.
[23] 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, J. Mach. Learn. Res. 11 (2010) 3371–3408.
[24] J. Zabalza, J. Ren, J. Zheng, H. Zhao, C. Qing, Z. Yang, P. Du, S. Marshall, Novel segmented stacked autoencoder for effective dimensionality reduction and feature extraction in hyperspectral imaging, Neurocomputing 185 (2016) 1–10.
[25] D. Chakraborty, V. Narayanan, A. Ghosh, Integration of deep feature extraction and ensemble learning for outlier detection, Pattern Recognit. 89 (2019) 161–171.
[26] G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1–3) (2006) 489–501.
[27] C.R. Rao, S.K. Mitra, Generalized inverse of a matrix and its applications, in: Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Theory of Statistics, The Regents of the University of California, 1972, pp. 601–620.
[28] D. Serre, Matrices: Theory and Applications, Springer-Verlag New York, 2010.
[29] L.L.C. Kasun, H. Zhou, G.B. Huang, C.M. Vong, Representational learning with ELMs for big data, IEEE Intell. Syst. 28 (6) (2013) 31–34.
[30] H. Zhou, G.B. Huang, Z. Lin, H. Wang, Y.C. Soh, Stacked extreme learning machines, IEEE Trans. Cybern. 45 (9) (2015) 2013–2025.
[31] X. Sun, J. Wang, C. Jiang, J. Xu, J. Feng, S.S. Chen, F. He, ELM-ML: study on multi-label classification using extreme learning machine, in: Proceedings of ELM-2015, 2, Springer, 2016, pp. 107–116.
[32] H.J. Rong, Y.S. Ong, A.H. Tan, Z. Zhu, A fast pruned-extreme learning machine for classification problem, Neurocomputing 72 (1–3) (2008) 359–366.
[33] H.J. Rong, G.B. Huang, N. Sundararajan, P. Saratchandran, Online sequential fuzzy extreme learning machine for function approximation and classification problems, IEEE Trans. Syst. Man Cybern. Part B (Cybern.) 39 (4) (2009) 1067–1072.
[34] H.J. Rong, Y.X. Jia, G.S. Zhao, Aircraft recognition using modular extreme learning machine, Neurocomputing 128 (2014) 166–174.
[35] P. Niu, Y. Ma, M. Li, S. Yan, G. Li, A kind of parameters self-adjusting extreme learning machine, Neural Process. Lett. 44 (3) (2016) 813–830.
[36] T. Gonçalves, P. Quaresma, A preliminary approach to the multilabel classification problem of portuguese juridical documents, in: Portuguese Conference on Artificial Intelligence, Springer, 2003, pp. 435–444.
[37] J. Read, B. Pfahringer, G. Holmes, E. Frank, Classifier chains for multi-label classification, Mach. Learn. 85 (3) (2011) 333.
[38] G. Tsoumakas, I. Vlahavas, Random k-labelsets: an ensemble method for multilabel classification, in: Proceedings of European Conference on Machine Learning, Springer, 2007, pp. 406–417.
[39] K. Trohidis, G. Tsoumakas, G. Kalliris, I.P. Vlahavas, Multi-label classification of music into emotions., in: Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR), 8, 2008, pp. 325–330.
[40] M.R. Boutell, J. Luo, X. Shen, C.M. Brown, Learning multi-label scene classification, Pattern Recognit. 37 (9) (2004) 1757–1771.
[41] E.C. Gonçalves, A. Plastino, A.A. Freitas, A genetic algorithm for optimizing the label ordering in multi-label classifier chains, in: Proceedings of the 25th International Conference on Tools with Artificial Intelligence (ICTAI), IEEE, 2013, pp. 469–476.
[42] G. Tsoumakas, I. Katakis, I. Vlahavas, Effective and efficient multilabel classification in domains with large number of labels, in: Proceedings of ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD08), 21, 2008, pp. 53–59.
[43] E.L. Mencia, J. Fürnkranz, Efficient pairwise multilabel classification for large-scale problems in the legal domain, in: Proceedings of Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2008, pp. 50–65.

Anwesha Law is pursuing Ph.D in Computer Science as a Senior Research Fellow at the Machine Intelligence Unit, Indian Statistical Institute. She has completed B. Tech. (Computer Science & Engineering) in 2011 from Sikkim Manipal Institute of Technology, Sikkim, India and M. Tech. (Computer Science & Engineering) in 2013 from Jadavpur University, Kolkata, India. Her research interests include multi-label learning, artificial neural networks, pattern recognition and machine learning.

Ashish Ghosh is a Professor with the Machine Intelligence Unit, Indian Statistical Institute. He has published over 200 research papers in internationally reputed journals and refereed conferences, and has edited eight books. He received the Young Scientists Award from the Indian Science Congress Association in 1992 and the Indian National Science Academy in 1995. He is acting as a member of the editorial boards of various international journals. His research interests include pattern recognition, machine learning, data mining, image and video analysis, soft computing, neural networks, evolutionary computation, and bioinformatics.