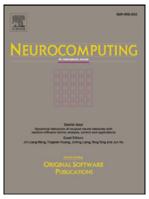
Contents lists available at ScienceDirect

Neurocomputing

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



Deep-FS: A feature selection algorithm for Deep Boltzmann Machines

Aboozar Taherkhani[a,*], Georgina Cosma[a], T. M McGinnity[a,b]

[a] *School of Science and Technology, Nottingham Trent University, Nottingham, UK*
[b] *Intelligent Systems Research Centre, Ulster University, Northern Ireland, Derry, UK*

A R T I C L E I N F O

Article history:
Received 5 March 2018
Revised 12 August 2018
Accepted 17 September 2018
Available online 22 September 2018

Communicated by Jiayu Zhou

Keywords:
Deep Boltzmann Machine
Deep learning
Deep Neural Networks
Feature selection
Restricted Boltzmann Machine
Generative models
Missing features

A B S T R A C T

A Deep Boltzmann Machine is a model of a Deep Neural Network formed from multiple layers of neurons with nonlinear activation functions. The structure of a Deep Boltzmann Machine enables it to learn very complex relationships between features and facilitates advanced performance in learning of high-level representation of features, compared to conventional Artificial Neural Networks. Feature selection at the input level of Deep Neural Networks has not been well studied, despite its importance in reducing the input features processed by the deep learning model, which facilitates understanding of the data. This paper proposes a novel algorithm, Deep Feature Selection (Deep-FS), which is capable of removing irrelevant features from large datasets in order to reduce the number of inputs which are modelled during the learning process. The proposed Deep-FS algorithm utilizes a Deep Boltzmann Machine, and uses knowledge which is acquired during training to remove features at the beginning of the learning process. Reducing inputs is important because it prevents the network from learning the associations between the irrelevant features which negatively impact on the acquired knowledge of the network about the overall distribution of the data. The Deep-FS method embeds feature selection in a Restricted Boltzmann Machine which is used for training a Deep Boltzmann Machine. The generative property of the Restricted Boltzmann Machine is used to reconstruct eliminated features and calculate reconstructed errors, in order to evaluate the impact of eliminating features. The performance of the proposed approach was evaluated with experiments conducted using the MNIST, MIR-Flickr, GISETTE, MADELON and PANCAN datasets. The results revealed that the proposed Deep-FS method enables improved feature selection without loss of accuracy on the MIR-Flickr dataset, where Deep-FS reduced the number of input features by removing 775 features without reduction in performance. With regards to the MNIST dataset, Deep-FS reduced the number of input features by more than 45%; it reduced the network error from 0.97% to 0.90%, and also reduced processing and classification time by more than 5.5%. Additionally, when compared to classical feature selection methods, Deep-FS returned higher accuracy. The experimental results on GISETTE, MADELON and PANCAN showed that Deep-FS reduced 81%, 57% and 77% of the number of input features, respectively. Moreover, the proposed feature selection method reduced the classifier training time by 82%, 70% and 85% on GISETTE, MADELON and PANCAN datasets, respectively. Experiments with various datasets, comprising a large number of features and samples, revealed that the proposed Deep-FS algorithm overcomes the main limitations of classical feature selection algorithms. More specifically, most classical methods require, as a prerequisite, a pre-specified number of features to retain, however in Deep-FS this number is identified automatically. Deep-FS performs the feature selection task faster than classical feature selection algorithms which makes it suitable for deep learning tasks. In addition, Deep-FS is suitable for finding features in large and big datasets which are normally stored in data batches for faster and more efficient processing.

1. Introduction

The successful performance of deep learning in various applications such as image recognition [1,2], speech recognition [3] and bioinformatics [4], has captured considerable attention in recent literature. Deep learning (DL) methods provide promising results on problems for which conventional machine learning methods

* Corresponding author.
E-mail addresses: aboozar.taherkhani@ntu.ac.uk (A. Taherkhani), georgina.cosma@ntu.ac.uk (G. Cosma), martin.mcginnity@ntu.ac.uk (T.M. McGinnity).

https://doi.org/10.1016/j.neucom.2018.09.040

have not made major progress, despite many attempts [1]. Conventional machine learning methods have limited ability to process raw data, and for this reason considerable effort is traditionally placed on feature engineering. Feature engineering represents data in a manner such that machine learning algorithms can identify patterns and classify the data. An important advantage of DL methods over conventional approaches (e.g. Artificial Neural Network, Support Vector Machine, Naïve Bayes), is that DL methods integrate the feature extraction and learning process into a single model, and thus feature engineering is dealt with as an integrated rather than a separate task.

Feature selection, aims to eliminate redundant and irrelevant features via different criteria. The most commonly used criteria measure the relevance of each feature to the desired output, and use this information to select the most important features [5,6]. Highly dependent features can be considered as redundant features and some of these can be eliminated during a feature selection process. Eliminating irrelevant and redundant features, results in a permanent reduction in the dimensionality of the data, and this can increase the processing speed and accuracy of the utilized machine learning methods.

Deep Neural Networks (DNNs), such as those proposed in [1,7,8] use feature extraction rather than feature selection methods for extracting underlying features from big data. For example Hinton et al. [1] proposed a DNN method which reduces data dimensionality through a non-linear combination of all input features in a number of layers of the neural network, and this approach inspired the development of new algorithms in the deep learning field [9,10,11]. DNNs can learn very complex relationships between variables through their high numbers of non-linear elements. However, if there exist a high number of irrelevant features in the input feature space, then the relationship between these irrelevant features may also be modelled. Modelling the irrelevant features acts as noise, and learning the associations between irrelevant features negatively impacts on the acquired knowledge of the network about the overall distribution of the data, as well as the computational time. Modelling the irrelevant features can also lead to overfitting a model [10], because the method learns irrelevant details from the training data and it becomes more biased to previously seen data [12]. A technique called Dropout [10] was proposed to increase the generalisation ability of neural networks which have a high number of neurons. However, a major limitation of the Dropout method is that it could retain all the input features and neurons that include redundant and irrelevant features.

The Deep Belief Networks (DBNs) proposed by Hinton and Salakhutdinov [1], and the Deep Boltzmann Machines (DBMs) proposed by Srivastava and Salakhutdinov et al. [7] are two types of DNNs which use densely connected Restricted Boltzmann Machines (RBMs). The high number of processing elements and connections, which arise because of the full connections between the visible and hidden units, increase the RBM's computational cost and training time. In addition, training several independent RBMs increases the training time [13–15]. When the scale of the network is increased, the required training time is also increased non-linearly. Reducing the number of input features can significantly reduce the size of the constructed weight matrix, and consequently it can reduce the computational cost of running deep learning methods, especially when a large network size is required for practical applications [16].

This paper proposes a novel algorithm, Deep-Feature Selection (Deep-FS), for embedding feature selection capabilities into DBMs, such that irrelevant features are removed from raw data to discover the underlying feature representations that are required for feature classification. DBMs primarily use an unsupervised learning method to initialize the learning parameters of a DNN. Then the initialized DNN is fine-tuned by a backpropagation method. Deep-FS combines the feature extraction property of DBM with a feature selection method which is based on the generative property of RBMs. RBMs are generative models and they can reconstruct missing input features. The proposed Deep-FS uses an RBM that is trained during the learning procedure of DBM to improve the efficiency of the method in dealing with high volumes of data. Deep-FS returns a reduced subset of features and improves the deep learning method's computational efficiency by reducing the size of the constructed network. DBMs are known to have good performance for feature extraction, and adding a feature selection ability to DBMs can lead to a new generation of deep learning models which have an improved ability to deal with highly dimensional data.

The remainder of the paper is structured as follows: Section 2 discusses the background to the work; Section 3 provides details of the proposed method; Section 4 describes the experimental results; and Section 5 provides a conclusion and future work.

2. Background

This section provides a background on Deep Boltzmann Machines and feature selection methods.

2.1. Deep Boltzmann Machine

Deep Neural Networks (DNNs) are mainly based on stochastic gradient descent and backpropagation training algorithms. Two main techniques are used for training DNNs. The first technique is based on a filtering strategy and the second one is based on unsupervised pre-training. The first filtering technique is used by Convolutional Neural Networks (CNNs) to locally filter inputs. Filtering is performed by convolving the input by weight matrices. In the second technique, information processing starts by using an unsupervised learning method. In this stage, unlabelled data can be used. Then, the DNN is fine-tuned by a supervised method using labelled data. Deep Belief Networks (DBN) [1] and DBMs [7,17] are examples which use this semi-supervised technique. Pre-training using an unsupervised method improves the generalisation of the trained network especially when the dataset contains a small amount of labelled data. This paper is focused on DBMs that include unsupervised learning during their first stage of training procedure.

DBMs have been used in different applications such as image-text recognition [7], facial expression recognition [18], 3D model recognition [19], and audio-visual person identification [20] and belong to a group of DNNs that uses a pre-training procedure. After the pre-training procedure, DBM is fine-tuned using labelled data [11,20]. A DBM is composed of a set of visible units corresponding to input data. Additionally, there are a network of symmetrically coupled stochastic binary units called hidden units. The binary hidden units are arranged in different layers and there are top-down and bottom up couplings between two adjacent layers. There are no direct connections between units in the same layer. A DBM represents the input data in several layers with increasingly complex representations. In DBM a learning procedure is executed to pre-train a number of layers in conjunction with each other. DBM can potentially be used to capture a complex internal representation of input data. The interaction of different hidden layers during the initial learning generates a deep network with a high ability to reconstruct ambiguous inputs through top-down feedback using different interacting layers. During the pre-training stage of a DBM, a learning procedure similar to RBMs is used. The RBM is discussed in Section 3.

Recently, a multimodal data processing method was designed based on DBMs [20]. In the multimodal data processing method, initially two networks of DBMs are trained separately on visual and auditory data. Then the output of the two DBMs are combined in a joint layer. The representation extracted from the joint layer is considered as input to a Restricted Boltzmann Machine (RBM). The RBM is trained on the joint layer representation. Then the entire network is fine-tuned by labelled data.

The fully interconnected network between the visible and hidden units increase the computational cost of a DBM, and finding a method to reduce the number of input feature through feature selection method can reduce the computational cost of a DBM.

2.2. Feature selection

The amount of data available to machine learning models is increasing exponentially. The number of features in datasets is also increasing, and many of the features are irrelevant or redundant, and thus not needed to improve the performance of a machine learning model [12]. Feature Selection (FS) can accelerate the learning process of a machine learning method because it allows the algorithm to learn using a smaller number of features. Additionally, it can improve the classification performance by preventing overfitting [12,21]. A feature selection method reduces the number of input features without significant decrease in the performance of a machine learning method [6,12,22].

Feature selection methods for classification can be divided into three main categories: filter, wrapper and embedded methods. These approaches are used to combine feature selection with a classification model. In filter methods, feature selection is performed as a prepossessing stage which is independent of the classification model. Each filter method ranks the features based on a criteria, and the highest ranked features are selected [6]. In feature ranking methods, the relevance of each feature to a class label is evaluated individually, and a weight is calculated for each feature. Then, features are ranked based on their weights and the top features with the highest weights are selected [12,21]. Maximum Relevance (MR) feature selection is a type of feature ranking algorithm, where Mutual information and kernel-based independence measures are usually employed to calculate the relevance score [23]. Feature ranking methods are simple and have low computational cost. Kira and Rendell's "Relief" algorithm [24], and the feature selection method proposed by Hall [25] are examples of feature selection methods that work based on the dependency of features to the class labels. Although the selected top features have the highest relevance to the class labels, they might be correlated with each other and have redundant information, and do not include all the useful information in the original feature sets.

The feature selection method proposed by Fleuret [26] considers the dependency between input features using conditional Mutual Information. The method considers the dependency of a new feature to the already selected features and eliminates that features that are similar to the previously selected features, because the main idea is that if a feature is dependent on previously selected features then it does not contain new information about the class, and it can be eliminated.

Peng et al. [27] proposed a feature selection method called Minimum Redundancy Maximum Relevance (mRMR) to solve the problem of feature redundancy. A subset of features which have high relevance to class labels and which are non-redundant are selected. The mRMR method has better performance compared to the method that only works based on the relevant features. In the mRMR filtering feature selection method, the dimension of the selected feature space is not set at the start of procedure.

In wrapper methods, the performance of a classifier is used to evaluate a subset of features. Different subsets of features are searched using different searching algorithms to find the optimal feature subset that gives the highest classification performance [6]. For an initial feature space dimensionality of N, a total of 2^N subsets can be evaluated, and that becomes an NP-hard problem. Sequential search, and evolutionary algorithms such as the Genetic Algorithm or Particle Swarm Optimization can be used to design a computationally feasible method for wrapper feature selection methods [6]. Wrappers have higher computational cost compared to filter feature selection methods.

With Embedded feature selection methods, the feature selection procedure is integrated into the training procedure of the classifier. It reduces the computational cost compared to wrapper methods where a high number of subsets should be retrained by the classifiers [6]. Guyon et al. [28] used Support Vector Machine (SVM) as a classifier to design an Embedded feature selection method. Features are evaluated during iterations of learning and the features that decrease the separation margin between classes are removed.

Feature selection algorithms coupled with feature extraction methods, can improve the performance of machine learning methods [12,21]. Feature selection algorithms can reduce the inputs to a classifier which in turn reduces computational cost and increases accuracy. However, classical feature selection algorithms are usually designed for small datasets and thus there is an emerging need to implement feature selection algorithms which can optimally search through large datasets with thousands of samples and a high number of features. This paper focuses on taking the idea of feature selection coupled with the feature extraction capability of deep learning to improve the performance of deep learning models.

2.3. Deep learning and feature selection

This section discusses a number of feature selection methods that are used with Deep Learning methods. Ruangkanokmas et al. [29] have used a classical filter-based feature selection approach for sentiment classification. A filter-based feature selection technique, called the chi-squared method, was proposed to select features and, thereafter the selected features were utilised to train a DBN. Feature selection and training the DBN were performed in two separate stages. Combining feature selection and DBN training, has the potential to improve the efficiency of a classifier.

Ibrahim et al. [30] have also used DBN, classical feature selection methods, and the unsupervised Active Learning method to process gene expression data. DBN was used to model high dimensional input data and to extract higher level representation of the input data. Then statistical classical feature selection methods, such as the *t*-test method, were used to select features from the higher level extracted features. Ibrahim et al's [30] method has used three cascaded independent modulus with separate computational costs. Computational costs can significantly increase when the number of training data is high. DBN is designed to work with a high number of training data samples and it requires feature selection functionality which is suitable for training large datasets, in order to prevent unnecessary computational cost.

Nezhad et al. [31,32] proposed a feature selection method which is based on a five-layer stacked Auto-Encoder deep network. Higher-level representative features were extracted from the hidden layer placed in the middle of the Auto-Encoder. Then, a classical feature selection method based on the feature ranking method and random forest was used to select features. After that, a supervised learning method was trained on the selected features and evaluated using the Mean Squared Error (MSE). The number of hidden neurons in the Auto-Encoder hidden layers, is adjusted to optimize the structure of the network based on the obtained results. This process is continued until reaching an acceptable result. This method has used different modules that increase the computational cost of the method.

Li et al. [33] proposed a deep feature selection (DFS) model using a deep structure of nonlinear neurons. They have used a multilayer perception (MLP) neural network to learn the nonlinearity of input data. Additionally, they use a sparse one-to-one linear layer before the MLP to select input features. The weight matrix of the first linear layer is a sparse matrix and the input features corresponding to nonzero weights are selected. Their proposed feature selection method is flexible such that the one-to-one linear layer corresponding to the feature selection can be added to other deep learning architectures. Despite the flexibility of the method, its accuracy is not perfect and experimental results have shown that the method did not outperform the random forest method.

Zhang and Wang [34] proposed a feature selection method for deep learning to classify scenes. They converted a feature selection method to a feature reconstruction problem. In a scene classification task they have selected the features that are more reconstructive than discriminative. However, removing discriminative features might reduce the classification accuracy in a typical classification task.

Deep learning methods, such as DBM, are usually composed of a high number of nonlinear processing elements that need a high number of training samples. On the other hand, the number of required observations (training samples) should grow exponentially with the number of input features [23,35] to train a DL method. Through feature selection, the number of input features and consequently the required number of training samples for training a DBM can be reduced. Therefore, feature selection is strongly required to help deep learning methods to be trained with less training data. In this paper, a feature selection method is proposed for DBM to improve its processing ability and reduce the computational cost of feature selection for DBM.

3. Principles of the proposed deep feature selection method

In this section, the mathematical properties of RBM are first described. Then, the RBM is used to design the proposed Deep-FS feature selection method. The principle of the proposed feature selection method which works based on RBM is presented in the second part of this section. Two versions of the proposed deep feature selection algorithm are presented. In the first version of the proposed feature selection algorithm, the RBM is not trained during feature selection. However, in the second version, the RBM is trained during the feature selection procedure.

3.1. Mathematical properties of Restricted Boltzmann Machines

A Boltzmann Machine (BM) is a parameterised probabilistic model. The parameters of a BM are trained to approximately model the important aspects of an unknown target distribution by using available samples drawn from the target distribution. The available samples that are used to train the parameters of the model are called training samples [36]. There are two types of units in a BM, these are the visible and hidden units (or neurons). The two sets of units are arranged in two layers. The first layer is constructed of visible units, and the hidden units (or neurons) are in the second layer. Each neuron or unit in the first layer corresponds to an input feature. For instance, if the input is an image then each visible unit corresponds to a pixel. In general, the visible units can accept different types of input features. The hidden units are used to model complex dependencies between visible units.

Restricted Boltzmann Machines (RBMs) are a special case of the general BM where there are no inter-connections between units in a single layer, i.e. each unit is fully connected to all units in another layer but does not have any connections with any units in its own layer (Fig. 1). RBMs have a close connection with statistical physics [37], and represent a type of energy-based model. During training,

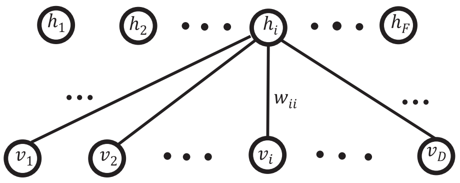


Fig. 1. Structure of a Restricted Boltzmann Machine. It is composed of two layers of neurons, namely the visible and hidden neurons. There are D visible neurons and F hidden neurons in this RBM.

the parameters of an RBM are adjusted to generate a model representing a probability distribution that is close to the actual probability distribution from which the data are drawn. RBMs have been successfully used for processing binary and real-value data [1,38–40].

Consider an RBM with D binary visible units. Let $\boldsymbol{V}$ be a vector that contains the states of the D binary visible units, i.e. $\boldsymbol{V} \in \{0,1\}^D$, and let $\boldsymbol{h}$ be a vector containing the states of the hidden units. Assume there are F hidden units in the RBM. The F dimensional hidden variables can be denoted by $\boldsymbol{h} \in \{0,1\}^F$. The joint configuration of $\boldsymbol{V}$ and $\boldsymbol{h}$ define the energy function (1).

$$E(\boldsymbol{V},\boldsymbol{h}) = -\sum_{i=1}^{D}\sum_{j=1}^{F} W_{ij} v_i h_j - \sum_{i=1}^{D} b_i v_i - \sum_{j=1}^{F} a_j h_j \tag{1}$$

where W_{ij} is a weight that connects the ith visible unit, v_i, and the jth hidden unit, h_j. b_i and a_j are biases that are related to the ith visible and the jth hidden units. The energy function is used to assign a joint distribution over the visible and hidden variables as shown in (2).

$$\boldsymbol{P}(\boldsymbol{V},\boldsymbol{h}) = \frac{1}{\boldsymbol{Z}} \exp\left(-\boldsymbol{E}(\boldsymbol{V},\boldsymbol{h})\right) \tag{2}$$

where Z is a normalizing term and it is called the partition function. Z is calculated by (3).

$$\boldsymbol{Z} = \sum_{\boldsymbol{V}}\sum_{\boldsymbol{h}} \exp(-\boldsymbol{E}(\boldsymbol{V},\boldsymbol{h})) \tag{3}$$

The sum is calculated over all possible pairs of ($\boldsymbol{V},\boldsymbol{h}$). Let $\boldsymbol{V}$ be a D dimensional vector and let $\boldsymbol{h}$ be an F dimensional binary vector. There are 2^{D+F} different pairs of ($\boldsymbol{V},\boldsymbol{h}$) when the visible units are binary. The conditional probabilities $P(\boldsymbol{h}|\boldsymbol{V})$ and $P(\boldsymbol{V}|\boldsymbol{h})$ can be calculated by (4) and (5).

$$P(\boldsymbol{h}|\boldsymbol{V}) = \prod_{j=1}^{F} p\left(h_j|\boldsymbol{V}\right) \tag{4}$$

$$P(\boldsymbol{V}|\boldsymbol{h}) = \prod_{i=1}^{D} p(v_i|\boldsymbol{h}) \tag{5}$$

Conditional Probabilities (4) and (5) can be written as follows:

$$\mathrm{p}\left(h_j = 1|\mathbf{V}\right) = \mathrm{g}\left(\sum_{i=1}^{D} W_{ij} v_i + a_j\right) \tag{6}$$

$$\mathrm{p}(v_i = 1|\mathbf{h}) = \mathrm{g}\left(\sum_{j=1}^{F} W_{ij} h_j + b_i\right) \tag{7}$$

where $g(x)$ is the logistic function, $1/(1+\exp(-x))$. The network can be trained to increase its assigned probability to an input by reducing the energy related to the input. The network parameters are updated by using the gradient-based method to reach the objective. Eq. (8) shows the derivative of the log probability of visible inputs on a set of observations, $\{\boldsymbol{V_n}\}_{n=1}^{N}$, with respect to a weight, W_{ij}.

$$\frac{1}{N}\sum_{n=1}^{N}\frac{\partial log(P(\boldsymbol{V_n}))}{\partial W_{ij}} = E_{data}\left[v_i h_j\right] - E_{model}\left[v_i h_j\right] \tag{8}$$

The first term, $E_{data}[v_i h_j]$, is the expectation of $v_i h_j$ with respect to data distribution. $E_{data}[v_i h_j]$, shows the frequency in which both visible unit, v_i, and hidden unit, h_i, have the binary values of one. $E_{model}[v_i h_j]$ is the same expectation with respect to the distribution defined by the model. The one-step contrastive diversion approximation is used to approximate $E_{model}[v_i h_j]$ [11,39]. It is calculated by running one iteration of the Gibbs sampler to reconstruct the visible unit using (6) and (7) to obtain (9).

$$\frac{1}{N}\sum_{n=1}^{N}\frac{\partial log(P(\boldsymbol{V_n}))}{\partial W_{ij}} \approx E_{data}\left[v_i h_j\right] - E_{model}\left[\tilde{v}_i h_j\right] \tag{9}$$

where $\tilde{v}_i$ is the reconstructed ith visible unit obtained through Gibbs sampling. Using the Gibbs sampler to approximate $E_{model}[v_i h_j]$ is called the contrastive approximation method. A similar procedure can be used to extract (10) and (11) for updating biases.

$$\frac{1}{N}\sum_{n=1}^{N}\frac{\partial log(P(\boldsymbol{V_n}))}{\partial a_j} \approx E_{data}\left[h_j\right] - E_{model}\left[\tilde{h}_j\right] \tag{10}$$

$$\frac{1}{N}\sum_{n=1}^{N}\frac{\partial log(P(\boldsymbol{V_n}))}{\partial b_i} \approx E_{data}[v_i] - E_{model}[\tilde{v}_i] \tag{11}$$

Gaussian-Bernoulli RBMs [1,7] are used for modelling real-valued vectors. In this case, the visible vector is $\boldsymbol{V} \in R^D$, and the hidden units are binary, $\boldsymbol{h} \in \{0,\ 1\}^F$. The RBM assigns the conditional distribution shown in (12) and (13).

$$\mathrm{p}\left(h_j = 1|\mathbf{V}\right) = \mathrm{g}\left(\sum_{i=1}^{D} W_{ij}\frac{v_i}{\sigma_i} + a_j\right) \tag{12}$$

$$p(v_i|\mathbf{h}) = \mathrm{N}\left(b_i + \sigma_i\sum_{j=1}^{F} W_{ij}h_j,\ \sigma_i^2\right) \tag{13}$$

A Gaussian distribution with mean $\mu = b_i + \sigma_i \sum_{j=1}^{F} W_{ij}$ and variance σ_i^2 is used for modelling a visible unit, i.e. $\mathrm{N}(\mu,\ \sigma_i^2)$. The derivative of the log probability on a set of observations, $\{\boldsymbol{V_n}\}_{n=1}^{N}$, is given by (14), which is similar to (8).

$$\frac{1}{N}\sum_{n=1}^{N}\frac{\partial log(P(\boldsymbol{V_n}))}{\partial W_{ij}} = E_{data}\left[\frac{v_i}{\sigma_i}h_j\right] - E_{model}\left[\frac{v_i}{\sigma_i}h_j\right] \tag{14}$$

Similar to the binary visible unit, the first expression on the right side of (14) can be calculated from the training data and the second expression can be approximated through a Gibbs sampler [7]. The result is obtained using (15).

$$\frac{1}{N}\sum_{n=1}^{N}\frac{\partial log(P(\boldsymbol{V_n}))}{\partial W_{ij}} \approx E_{data}\left[\frac{v_i}{\sigma_i}h_j\right] - E_{model}\left[\frac{v_i}{\sigma_i}h_j\right] \tag{15}$$

The input features, v_i, are usually normalized to have zero mean, $\mu = 0$, and a variance of one, $\sigma_i = 1$. This simplifies (15) and it becomes similar to (9). The following sections refer to (9) for both binary and real value input data.

3.2. Proposed RBM-based deep feature selection algorithm

An RBM is a generative probabilistic model and can generate the probability of the value of a visible unit given the states of hidden units. This property can be used to reconstruct missing visible units. The generative property of RBMs has been adopted to draw samples from the learned distribution to extract textures from images [41]. The generative property of RBMs has also been used to sample the missing parts of an input image in image denoising tasks [42]. The proposed Deep-FS algorithm adopts the generative property of RBM to define a method for feature selection.

Deep-FS aims to find a set of features with useful information. Therefore, features that do not hold useful information about the input data are removed by the generative property of the RBM. The final selected features have a lower number of features and they reduce the complexity of the network. Feature selection is performed via three steps 1) Initial training: of RBM on training data with all the features; 2) Feature elimination: Removing extra features by the initially trained RBM; and 3) Main Training: Training the DBM with the initially trained RBM on the training data which consists of the selected features. Each of these steps is described in the remainder of this section.

Step 1 - Initial training: The first step is performed via the learning method described in Section 3.1 by using (9). During the training procedure, training data is input into the RBM. Then the outputs of hidden neurons are calculated for all training data, and the first expression in (9) is calculated, i.e. $E_{data}[v_i h_j]$. In the next step the inputs are reconstructed by Gibbs sampling and the second expression in (9), $E_{model}[\tilde{v}_i h_j]$, is calculated accordingly. Finally, the RBM weights are adjusted by calculating the difference $E_{data}[v_i h_j] - E_{model}[\tilde{v}_i h_j]$ as shown in (9).

Step 2 - Feature elimination: In the second step, features are eliminated through a proposed feature selection algorithm which is describing in this step. The proposed algorithm can be tuned to eliminate a single feature or a group of features in each evaluation. The proposed algorithm starts with the set of all input features and evaluates the effect of each group of features by using the trained RBM. The learning aim in RBM is to minimize the network error or maximize the log-likelihood as shown in (9). The derivative of loss or error function of RBM can be obtained by multiplying (9) by the value of minus one [11]. During the learning process, the absolute value of the error is reduced. Consequently, the weight adjustments are stabilized. The learning procedure is stopped when the error reaches zero or a predefined number of learning epochs is reached. The weight adjustment can become zero when the error reaches zero. In particular, consider the error related to one of the input features. The error related to the ith visible unit is given in (16) by using (9).

$$-\frac{1}{N}\sum_{n=1}^{N}\frac{\partial log(P(V_n))}{\partial W_{ij}} \approx \left(E_{model}\left[\tilde{v}_i h_j\right] - E_{data}\left[v_i h_j\right]\right) \tag{16}$$

The reconstruction error is defined using (17):

$$e_i \propto \left(\left[\tilde{v}_i h_j\right] - \left[v_i h_j\right]\right)^2 = \left((\tilde{v}_i - v_i)h_j\right)^2$$

$$e_i = E_{model}\left[(\tilde{v}_i - v_i)^2\right] \tag{17}$$

where e_i is the reconstruction error related to input feature i. A lower e_i can cause a lower absolute weight adjustment.

The RBM learning rule is based on feature extraction and dimension reduction. In RBM's feature extraction method, different visible inputs are combined and hidden features are extracted. During this learning procedure, the value of weight adjustment is stabilized and consequently the absolute value of error shown in (16) is reduced. e_i described in (17) is used to define an elimination criterion for the proposed feature selection method.

The input features are investigated to find whether a feature v_i can be reconstructed by using other input features, i.e. to find whether the other features contain enough information to reconstruct the v_i. The ith visible unit, v_i, is eliminated and it is reconstructed by the trained RBM. To eliminate v_i, it is initialized to zero. Therefore, the visible unit v_i with the initial value of zero does not contribute to the output of the hidden variables (see (6) and (12)). The hidden features are generated only by the other visible inputs. Then the ith visible unit is reconstructed by the hidden units using (7) for binary or (13) for continuous or real value data. The ith reconstructed feature is called $\bar{v}_i$. Then (17) is used to calculate the reconstruction error, $\bar{e}_i = E_{model}[(\bar{v}_i - v_i)^2]$. Note that $\bar{v}_i$ and $\tilde{v}_i$ both are reconstructed visible units. However, $\bar{v}_i$ is generated when the initial value of v_i is set to zero, and $\tilde{v}_i$ is generated when v_i is set to its original value.

If the reconstruction error after elimination of ith visible unit, $\bar{e}_i$, is equal or less than the original reconstruction error of the visible unit, e_i, it becomes evident that the visible unit does not add any knowledge to the network or it reduces the general knowledge of the network. Therefore, removing the ith input feature reduces the complexity of the network and may also reduce the error of the network. Additionally, reducing the error can lead to the reduction of the absolute value of the learning error shown in (16). Consequently, the absolute value of the weight adjustment shown in (9) is reduced and the weights become stabilized. Thus, the feature selection method can cooperatively work with the RBM weight adjustment method to reduce the final error.

The proposed feature selection has been designed based on training aim of RBM. Suppose that a RBM is trained on all the input features. The weights are updated based on (9). The weight adjustment is dependent on $E_{data}[v_i h_j] - E_{model}[\tilde{v}_i h_j]$, a high value for the expression leads to high weight adjustment and a low value leads to low weight adjustment. If the weights are adjusted in such way that the RBM can regenerate the trained data, i.e. $v_i = \tilde{v}_i$, then $[v_i h_j] = [\tilde{v}_i h_j]$ leads to $E_{data}[v_i h_j] = E_{model}[\tilde{v}_i h_j]$. The equality leads to zero adjustment for the weights, i.e. $E_{data}[v_i h_j] - E_{model}[\tilde{v}_i h_j] = 0$ which implies that the network has trained perfectly. So an absolute value close to zero is desirable for the expression $E_{data}[v_i h_j] - E_{model}[\tilde{v}_i h_j]$, the zero value for the expression means that the network is trained successfully and it does not need more weight adjustments.

Based on the abovementioned desired aim for training an RBM, reducing the weight adjustments by making $E_{data}[v_i h_j] - E_{model}[\tilde{v}_i h_j]$ close the zero, the proposed feature selection is designed. The features are checked (individually or in groups) to determine whether removing the features makes the RBM close to the training aim. If the reconstructed visible unit is close to the initial value of the visible unit i.e. $v_i = \tilde{v}_i$, the learning has become close to its training aim and it does not need a high weight adjustment (because $E_{data}[v_i h_j] - E_{model}[\tilde{v}_i h_j]$ has a value close to zero). Therefore, if by removing a visible unit a similar situation occurs, i.e. the error of the reconstructed visible unit is reduced, which implies that the removed feature not only does not add new information to the network (to reconstruct the visible unit) but it also reduces the overall knowledge by increasing the reconstruction error. Therefore, it is better to remove the visible unit from the selected feature set to make RBM become close to its training aim. The proposed method, similar to the optimal brain surgeon approach [43], works based on the comparison of two errors. In the optimal brain surgeon approach the difference in the two errors is created by pruning of learning parameters, however in the proposed method the difference in errors is generated by eliminating the features. Then based on the derived reconstruction error the features are selected. Additionally, the proposed feature selection is designed to work with DBM.

The procedure of the proposed feature selection is summarized in the pseudocode described in Table 1. After training the RBM, it is used to calculate the reconstruction error e_i for ith input features by using (17) for all i. In the while loop in Table 1, the proposed method investigates a group of N_e visible features. In each iteration a group of N_e features are eliminated by setting the corresponding visible units to zero. Investigation of a group of input features when N_e>1 (see Fig. 2), reduces the number of required iterations and consequently it reduces the processing time. The reconstruction error $\bar{e}_k$ is calculated for the N_e eliminated features. $\bar{e}_k$ is the reconstruction error of kth visible unit in the N_e eliminated features in the while loop in Table 1. For the kth eliminated feature if $\bar{e}_k < e_k$, where e_k is the reconstruction error of same input feature before eliminating the features, then the kth eliminated feature is removed from input features permanently. Otherwise, the kth eliminated feature is considered as selected features. The while loop in Table 1 is continued until all the input features are investigated.

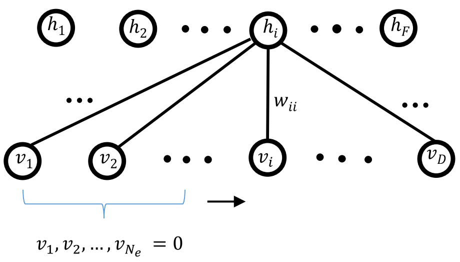


Fig. 2. In each iteration, a number of input features, N_e, are investigated, as a group of features which can potentially be eliminated. The investigated group of features are temporary removed by setting their values to zero, and then they are reconstructed by the trained RBM. Setting N_e to a high value reduces the number of RBM reconstruction iterations required to investigate the entire input features. N_e is set at the start of learning and a fixed N_e is used in the all iterations.

In Fig. 2 the investigated group of features are selected by adjacency. Reconstruction error $\bar{e}_i$ is utilized to evaluate the effect of removing groups of features which have not been selected based on adjacency; and groups of features which contain adjacent features. $\bar{e}_i$ is the reconstruction error of the ith input feature when N_e input features are eliminated by setting them to zero. As shown in Table 1, the $\bar{e}_k < e_k$ condition is used to decide whether to remove the kth feature, i.e. a visible unit with a lower reconstruction error is removed permanently. Therefore, visible units with lower $\bar{e}_i$ in the current iteration are more likely to be removed in the next feature selection iteration. Consequently, $\bar{e}_i$ can be used to find the next N_e features that should be tested for removal. The N_e visible units that have the lowest $\bar{e}_i$ in current iteration of feature selection are selected for elimination test in the next iteration. In this case, the N_e selected visible units usually are not adjacent units.

An alternative version of the algorithm is presented in Table 2, where the RBM is trained before feature selection on the all the input features (similar to the first version presented in Table 1). Additionally, it is trained during the feature selection procedure using the reduced number of features. In the alternative version, the RBM is first trained, then, e_i is calculated using (17) for all visible features (see Table 2). After that, a set of N_e features are selected as candidate features and they are temporarily eliminated from the input feature set by setting their values to zero. Then it is decided if the kth eliminated feature should be removed permanently by using $\bar{e}_k$. The elimination of N_e features are repeated to investigate all the input features. After removing of every Nth feature the RBM is trained with the reduced number of input features. Then the new trained RBM is used to continue the feature selection procedure (see Table 2).

Step 3 – Main training: After eliminating redundant features by using one of the two algorithms provided in Tables 1 and 2, the training of the RBM continues using the selected features, i.e. RBM with the remaining visible units is initialized by the previous corresponding weights and the learning is continued. The RBM and the selected features are used for training the DBM similar to the

Table 1
Pseudocode of the proposed feature selection method when the initially trained RBM is not trained during feature selection.

```
Train RBM on the training data.
Calculate the initial reconstruction error $e_i$ by the trained RBM using (17) for all $i$.
$N_v =' number\ of\ visible\ features'$
$i = 1$
While $i < N_v$:
  Select $N_e$ features for evaluation.
  $N_s = N_e$
  Set $v_k = 0\ for\ k \in \{N_e\ selected\ features\}$ (elimination of the $N_e$ features).
  Calculate the reconstruction error $\bar{e}_k$ for each eliminated feature using (17).
  for $k \in \{N_e\ eliminated\ features\}$:
    if $\bar{e}_k < e_k$ then:
        Remove the $k^{th}$ visible unit.
        $N_v = N_v - 1$
        $N_s = N_s - 1$
    else:
        Reset $v_k$ from 0 to its original value, and add it to selected features.
  $i = i + N_s$
```

Table 2
Pseudocode of the alternative version of the proposed feature selection method when the RBM is trained during the feature selection procedure.

```
Train RBM on the training data.
Calculate the initial reconstruction error $e_i$ by the trained RBM using (17) for all $i$.
$N_v =' number\ of\ visible\ features'$
$i = 1$
$N_r = 0$; #number of removed features
While $i < N_v$:
  Select $N_e$ features for evaluation.
  $N_s = N_e$
  Set $v_k = 0\ for\ k \in \{N_e\ selected\ features\}$ (elimination of the $N_e$ features).
  Calculate the reconstruction error $\bar{e}_k$ for each eliminated feature using (17).
  for $k \in \{N_e\ eliminated\ features\}$:
    if $\bar{e}_k < e_k$ then:
      Remove the $k^{th}$ visible unit.
      $N_v = N_v - 1$
      $N_s = N_s - 1$
      $N_r = N_r + 1$
    else:
        Reset $v_k$ from 0 to its original value, and add it to selected features.
  $i = i + N_s$
  If $N_r > N_{th}$:
    Train RBM with current reduced number of features.
    Calculate the initial reconstruction error $e_i$ by the trained RBM using (17).
    $N_r = 0$
```

method used in [17]. The learning method proposed in [17] has an unsupervised learning procedure where a DBM is initially trained. Then the learning parameters of the trained DBM are used to initialize a corresponding multilayer neural network. Finally, a standard back propagation method is used to train the multilayer neural network.

In the proposed methods, the knowledge acquired during the training process of the DBM (i.e. training of the RBM as a building block of the DBM) is used to perform the feature selection, i.e. the result of pre-training of DBM is used for feature selection. Therefore, the computation cost of the feature selection task is reduced, because feature selection is performed during the DBM's learning process.

The proposed feature selection method uses the generative ability of RBM to reconstruct eliminated features and then calculates reconstruction errors to determine whether to retain or remove features. Other deep learning algorithms that have the generative property can also be used by the proposed method. For instance, the Deep Belief Network (DBN) is a deep learning method that uses RBM during its training procedure; therefore, the proposed feature selection method can be used with DBN. The Auto-Encoder method is also a deep learning method that reconstructs its input at the network output. Therefore, the proposed method can be extended to an Auto-Encoder. The proposed feature selection method has the ability to work with data that are suitable for deep learning methods.

3.3. Normalized error

The normalized error shown in (18) is used to evaluate the performance of the proposed method.

$$E = \frac{\sum_{i=1}^{D} \bar{e}_i}{D} \tag{18}$$

where D is the number of visual input features, and $\bar{e}_i$ is the reconstruction error related to the ith visible unit calculated by (17).

4. Experimental results

This section first presents the experimental results of applying the proposed Deep-FS method on five benchmark datasets, namely

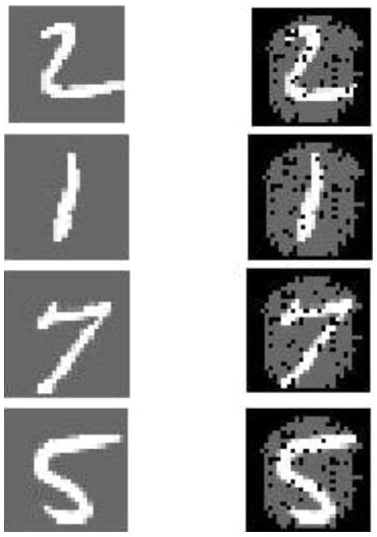


Fig. 3. Deep-FS removes those peripheral pixels that are shown by dark dots on the right column. The removed pixels do not have useful information about a digit.

MNIST [44], MIR-Flickr [7], GISETTE [45], MADELON [45], and PANCAN [46]. Datasets with a high number of features and training samples were selected in order to evaluate the accuracy and speed of the proposed feature selection method on high dimensional datasets. Thereafter, Deep-FS is compared with other feature selection methods using the MNIST dataset. Finally, Deep-FS's time complexity is analysed using MNIST and randomly generated data.

4.1. Experimental results on MNIST

This section provides a brief description of the MNIST image dataset, and provides an explanation and illustration of how features are selected and removed using the proposed Deep-FS method. Next, the effect of training an RBM during the feature selection procedure is discussed via an empirical comparison with the two proposed methods described in Tables 1 and 2. After that the effect of the application of the two selection methods for evaluating the features, i.e. selecting the $\boldsymbol{N_e}$ adjacent features (Fig. 1), and selecting the $\boldsymbol{N_e}$ features that have the lowest reconstruction error, $\bar{\boldsymbol{e}}_{\boldsymbol{i}}$, are investigated. Finally, the proposed feature selection method, Deep-FS, is compared with the original DBM [17].

4.1.1. MNIST image dataset

In the first set of the experiments, Deep-FS is applied on the MNIST image dataset [44]. MNIST contains 60,000 training samples and 10,000 test image samples of handwritten digits. In the MNIST dataset, each image is centred in a 28×28 pixel box. Each image in the MNIST dataset is obtained from an original image which is in a 20×20 pixel box through a transformation. The transformation is fully performed in such a way that the centre of mass of the pixels is preserved in the two images. This pre-processing is described by LeCun et al. [44]. The handwritten digits have different thickness, angular alignment, height and relative position in the image frame.

4.1.2. Illustration of the selected and removed features on the MNIST dataset

Fig. 3 illustrates some samples of reconstructed images after applying Deep-FS described in Table 2 on the MNIST dataset. The left column of Fig. 3 shows samples of the original images and the right-hand column shows the reconstructed images with the removed pixels filled by black pixels. Fig. 3 shows that the method has in practice removed pixels surrounding the digits. The peripheral pixels do not contain any useful information about the

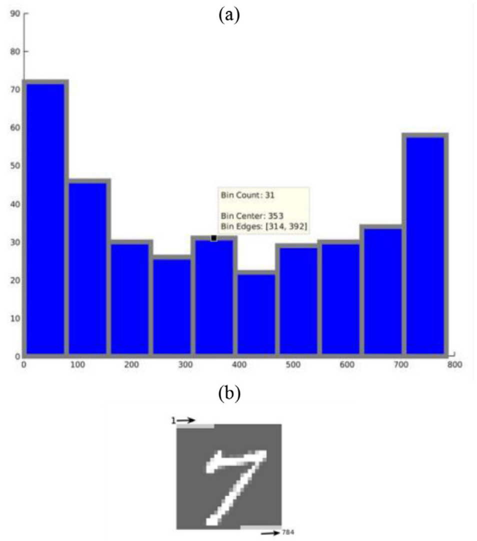


Fig. 4. (a) Histogram of removed pixels. (b) Order of indexing the pixels. The histogram shows that a high number of pixels which are located in the upper and lower sections (corresponding to the first and the last columns in the histograms) are removed. The removed pixels do not contain useful information about the digit.

digit, and therefore were removed. Some other pixels in the middle area of the images have been removed. The removed pixels can be reconstructed by the information from neighbouring pixels, and removing these pixels does not destroy the general appearance of the digits.

In Fig. 4(a), a histogram illustrating the removed pixels for digit 7 is shown. The pixels are indexed from 1 to 784 as shown in Fig. 4(b). The pixel located at the top right corner has the index of 1, the pixel next to it on its right side has the index of 2 and so forth. The first and last columns in the histogram, shown in Fig. 4(b), have higher values than others and correspond to the pixels that are at the top and bottom of the figure, respectively. As shown in Fig. 3, there is not much information in those pixels, and therefore these were removed appropriately by the proposed feature selection algorithm. Less pixels are removed from the area in the middle part of each image of the digits, as shown by shorter columns in the middle part of Fig. 4(b).

In this paper the learning method proposed by Salakhutdinov and Hinton [17] is used as the baseline learning method for comparison purposes. In this baseline learning method, a DBM is initially trained, and thereafter the trained DBM is used to initialize a multilayer neural network. Then, a standard back propagation method is used to train the multilayer neural network.

The input vector has $28 \times 28 = 784$ units. The DBM has two hidden layers. The first hidden layer has 500 hidden units and there are 1000 hidden neurons in the second hidden layer. A similar structure is used is used in the proposed method to compare the results. The difference between the Deep-FS and DBM [23] is in the first layer. Deep-FS selects a set of input pixels to reduce the number of input units. The maximum number of learning epochs are the same for all the methods. For example, the DNNs are fine-tuned for 100 learning epochs using backpropagation methods.

4.1.3. The effect of training the RBM during feature selection

Table 3 shows the experimental results when N_e adjacent features are used during the feature selection s stage. In this experiment, the results of training the first RBM during the feature selection stage(using the algorithm described in Table 1) are compared with those results when the RBM is not trained during feature selection (i.e. using the algorithm of Table 2). In the first column, Deep-FS$_1$, Deep-FS$_5$, and Deep-FS$_{10}$ are proposed meth-

Table 3
Experimental results on the MNIST data when adjacent features are used.

Method	N_e [a]	#Input features	Classification error during testing out of 10,000[b]	Processing time in seconds (s)
		RBM is not trained during feature selection (See Table 1)		
Deep-FS$_1$	*1*	*413*	101	55,872 s
Deep-FS$_5$	*5*	*482*	100	53,925 s
Deep-FS$_{10}$	*10*	*488*	97	53,425 s
		RBM is trained during feature selection (See Table 2)		
Deep-FS$_1$	1	406	106	56,087 s
Deep-FS$_5$	5	483	95	53,989 s
Deep-FS$_{10}$	*10*	*488*	96	53,287 s

[a] Number of eliminated features before a reconstruction.
[b] Classification–error during Training Out of 60,000 is zero for all the methods.

Table 4
Experimental results on the MNIST dataset when the features with the lowest reconstruction error, $\bar{e}_i$, are used.

Method	N_e [a]	#Input features	Classification error during testing out of 10,000 [b]	Processing time in seconds (s)
		RBM is not trained during feature selection (Algorithm shown in Table 1)		
Deep-FS$_1$	*1*	*415*	103	55,134 s
Deep-FS$_5$	*5*	*430*	94	52,805 s
Deep-FS$_{10}$	*10*	*430*	90	52,442 s
		RBM is trained during feature selection (Algorithm shown in Table 2)		
Deep-FS$_1$	*1*	*409*	101	55,571 s
Deep-FS$_5$	*5*	*416*	100	53,106 s
Deep-FS$_{10}$	*10*	*428*	94	52,716 s

[a] Number of eliminated features before a reconstruction.
[b] Classification–error during Training Out of 60,000 is zero for all the methods.

ods when N_e is set 1, 5, and 10, respectively. N_e is the number of pixels that are eliminated before feature reconstruction. In each iteration of the feature selection phase, N_e features are eliminated and thereafter reconstructed together by RBM as explained in Section 3.2 and Table 1. Deep-FS can evaluate features (pixels) one by one (N_e=1) or it can evaluate a group of features (pixels) as described in Section 3.2 ($N_e > 1$). In the later situation, instead of searching by a single feature, N_e=1, the search process searches by a group of features in groups of 5 and 10, i.e. N_e= 5 and N_e= 10, respectively. The third column, #Input Features, is the number of input pixels (features) which are selected during feature selection.

The fourth column, Classification Error During Testing, shows the classification error on the testing data (i.e. test images). Each test image is input into the network then the output of the ten neurons, each of which is corresponding to a class, on the output layer are investigated. The test input is assigned to a class that corresponds to the output neuron that has the maximum output value. The number of incorrect assignments are collected over 10,000 testing images and the results are reported in column Classification Error during Testing. The processing time column, is the total required time for the methods to perform training on 60,000 training images and also to test the trained network on the 10,000 testing images. Experiments were performed on an Intel E5-2640 v4 2.40 GHz processor with 64 GB RAM. Table 3 shows training RBM during feature selection slightly reduces the errors when $N_e > 1$ adjacent features are used.

4.1.4. The effect of evaluating the features with the lowest $\bar{e}_i$

Table 4 shows the results when the N_e features that have the lowest reconstruction error, $\bar{e}_i$, are used during the feature selection process. The N_e features with the lowest reconstruction errors in the current iteration of feature selection are evaluated in the next iteration of the feature selection procedure, and these features are usually not adjacent. Comparing Tables 3 and 4 reveals that using the features that have the lowest reconstruction error, $\bar{e}_i$, during the feature selection procedure (Table 4) improves the accuracy of the proposed feature selection method, compared to when adjacent features are used (Table 3). For instance, comparing Tables 3 and 4, the classification error for Deep-FS$_{10}$ is reduced from 97 (Table 3 when RBM is not trained) to 90 (Table 4 when RBM is not trained). The results show that the approach which eliminates N_e features with the lowest reconstruction error has higher accuracy compared to the method that eliminates N_e features which are adjacent.

Additionally, the results in Table 4 show that using the feature selection method, which does not train the RBM during the feature selection procedure (see Table 1), has higher accuracy compared to when RBM is trained during feature selection in addition to the initial training of RBM (see Table 2) when $N_e > 1$. For instance, Table 4 shows that Deep-FS$_{10}$ misclassified fewer image samples when RBM was not trained during the feature selection procedure, and the number of misclassified images was reduced from 94 to 90.

In conclusion, highest classification accuracy is achieved when a RBM is not trained during the feature selection procedure, and hence when Deep-FS$_{10}$ uses the initially trained RBM before feature selection and then performs the feature selection procedure by the initially trained RBM.

4.1.5. Comparing Deep-FS with the baseline DBM

The proposed Deep-FS method was compared against the DBM [17] method using the MNIST dataset. The comparison considered the effect of each approach on reducing the number of input features and misclassified cases, and reduction in processing time. The baseline DBM is the one which was originally introduced by Salakhutdinov and Hinton [17]. In this comparison, Deep-FS$_{10}$ is used because the experiments in Sections 4.1.3 and 4.1.4 revealed that removing features in groups of 10 is a better feature elimination strategy which provides higher classification accuracy. For Deep-FS$_{10}$, RBM is not trained during the feature selection procedure. Additionally, features are evaluated based on the lowest reconstruction error, $\bar{e}_i$, as its results are reported in Table 4.

Table 5
Experimental results on the MNIST dataset when the features with the lowest reconstruction error, $\bar{e}_i$, are used and the RBM is trained during feature selection.

Method	#Input features	Classification error during testing out of 10,000[b]	Processing time in seconds (s)
Baseline DBM [17]	784	97	55,505 s
Deep-FS$_{10}$[a]	*430*	90	52,442 s

[a] RBM is not trained during feature selection, and features with the lowest reconstruction error, $\bar{e}_i$, are used (see Table 4).
[b] Classification–error during Training Out of 60,000 is zero for all the methods.

The baseline DBM is trained on all 748 input pixels (see first row of Table 5). Table 5 shows that the proposed Deep-FS$_{10}$ method has reduced the number of input features. The proposed method selected 430 features out of 784 total input features (see second row of Table 5). The results show that the proposed method removes more than 45% of the input features. The proposed method reduced the number of misclassified cases from 97 to 90 for the baseline DBM and the proposed method respectively (see Table 5). The results show that the proposed method's capability in finding redundant features that do not add new information to the network and removing these redundant features without reducing the classification accuracy and it is much faster than the baseline DBM. Selecting appropriate features based on the lowest error difference helps the algorithm find appropriate features, and increases accuracy. The classification error of all the methods on the 60,000 training samples are zero.

The processing time of the experiment for the proposed method is reduced by about 5% when N_e is increased from 1 to 10 features. The processing time is reduced from 55,134 s for N_e=1 to 52,442 s for N_e=10 (see Table 4 when RBM is not trained during feature selection). Eliminating a number of input features, N_e>1, in each investigation reduces the required number of reconstruction procedures which consequently reduces processing time. The number of selected features is increased by about 4% when N_e is increased from 1 to 10 (see Table 4 when RBM is not trained during feature selection). A reconstruction error related to each eliminated feature is increased when the number of eliminated features, N_e, before the reconstruction is high. Eliminating a high number of input features during feature selection increases the reconstruction errors, $\bar{e}_k$. Consequently, a high number of features is kept in the selected feature set when $\bar{e}_k < e_k$ is used for removing the features (Table 1). Moreover, Table 5 shows that the processing time of Deep-FS$_{10}$ is lower than the baseline DBM [17]. The processing time of the proposed Deep-FS$_{10}$ is 52,442 s while the processing time of the baseline DBM [17] is 55,505 s, i.e. it has about 5.5% lower processing time. The low number of selected features, 430, for Deep-FS$_{10}$ compared to the initial 784 features that should be processed by the baseline DBM reduces the processing time of the proposed Deep-FS$_{10}$.

4.2. Experimental results on the MIR-Flickr dataset

In the second set of experiments, the performance of the proposed Deep-FS$_1$ method, that was used for MNIST data in Section 4.1, is tested using the MIR-Flickr dataset obtained from the Flickr.com social photography site [7,47]. The MIR-Flickr dataset contains one million samples. Each input sample consists of an image which may have user-defined image tags. Additionally, some of the input samples, image and user text tags, are labelled. Out of the one million input samples, only 25,000 images with user assigned tags are annotated with labels and the remaining 975,000 samples are unlabelled. Labelling large data is a very demanding task. The images and their corresponding user assigned tags are annotated by 38 labels including object and scene categories such as tree, bird, people, clouds, indoor, sky, and sunset. Each sample can belong to a number of classes.

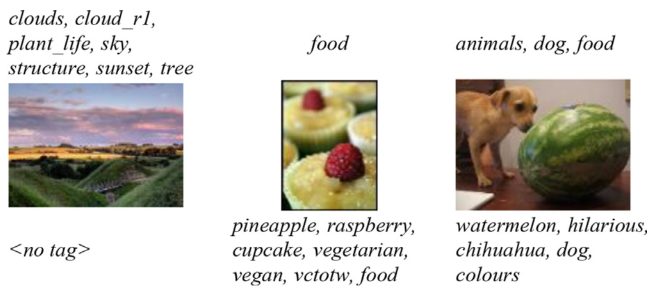

clouds, cloud_r1, plant_life, sky, structure, sunset, tree
food
animals, dog, food
<no tag>
pineapple, raspberry, cupcake, vegetarian, vegan, vctotw, food
watermelon, hilarious, chihuahua, dog, colours

Fig. 5. Samples from the MIR-Flickr dataset. The top words are the annotations and the words behind each image are tags written by users. The first image has no tags. Most of the images belong to various classes with different annotations as shown in the top row.

The unsupervised learning ability of RBMs, which are the building blocks of a DBM, enables DBMs to be trained using a huge number of unlabelled data, and RBMs and DBMs are known for their suitability in training unlabelled data. After initial training, a limited number of labelled data can be used for fine-tuning the model. Out of the 25,000 labelled samples in the MIR-Flickr dataset, 10,000 samples are used for training and another 5000 labelled samples are used as a validation set. The remaining 10,000 samples are used during the testing stage [7]. Each sample in the MIR-Flickr dataset has two sets of features, i.e. text and image features. First, the text features are described then the image features are introduced.

Many words which appear in the user defined tags are not informative and some of them are filtered. To organize the text input, a dictionary is generated. The dictionary contains the 2000 most frequent tags, which were extracted from a set of user tags found in one million samples. Then each text input is refined, and each text input contains only the text in the dictionary. Therefore, the text data of each sample is represented by the vocabularies of its user tags that are in the dictionary (i.e. the tags are restricted to the ones in the dictionary). Additionally, different feature extraction methods are used to extract real value features for each image. In the previous experiment on the MNIST dataset, (see Section 4.1) the inputs had binary values, however, here there are a number of real value features for each image. In total, 3857 features were extracted for each image [7]. The following features were extracted for each image [7]: 1. Concatenating Pyramid Histogram of Words (PHOW) (2000 features), 2. Gist (960 features), 3. Colour Layout Descriptor (192 features), 4. Colour Structure Descriptor (256 features), 5. Scalable Colour Descriptor (256 features), 6. Edge Histogram Descriptor (150 dimensions), and 7. Homogenous Texture Descriptor (150 features). The different features extract different aspects of an image. Three samples from MIR-Flickr dataset are shown in Fig. 5. The top row of Fig. 5 shows the annotations or labels and the bottom row shows the user tags.

Table 6
Classification results using features extracted from the first hidden layer of image pathway.

Layer	MAP	Prec@50	Image features
Baseline DBM [7]	0.476 ± 0.003	0.756 ± 0.005	3857
Proposed Deep-FS	0.478 ± 0.003	0.756 ± 0.008	3082

Table 7
Classification results using features extracted from joint hidden layer.

Layer	MAP	Prec@50	Image features
Based DBM [7]	0.622 ± 0.003	0.880 ± 0.005	3857
Proposed Deep-FS	0.622 ± 0.003	0.879± 0.005	3082

The DBM used in [7] is employed as the baseline learning method for comparison, for the MIR-Flickr image-text data in this set of experiments. There are 3857 Gaussian visible units with real number output values for image input, and there are two hidden layers for image pathway each of them composed of 1024 binary units. A Replicated Softmax model [7] with 2000 input units is used for text inputs. The text pathway is completed by two hidden layers each of which has 1024 binary units. A joint layer with 2048 binary hidden units is placed after the image and text pathways.

Each sample, image with corresponding user text tags, found in the MIR-Flickr dataset can belong to a number of classes. Mean Average Precision (MAP) and precision at top-50 predictions (Prec@50) are two standard methods commonly adopted to evaluate multi-label classification tasks [7]. MAP and Prec@50 are also used in this research to evaluate the classification performance of the proposed Deep-FS and the baseline DBM on the MIR-Flickr data.

Deep-FS uses the first 5000 batches of data with whole features to train the RBM. There are 128 samples in each batch. Then the proposed feature selection method uses the trained RBM to select features. The learning process is continued with the reminder of the training data by using the selected features. Extracted features from the first hidden layer of the image pathway are classified by the logistic regression method to show the effect of the feature selection method on the classification results. The results are shown in Table 6. Deep-FS returned a higher MAP than the baseline DBM method, achieving values of 0.478 and 0.476 respectively. There is no notable difference in Prec@50. Deep-FS selects 3082 out of 3857 image features. The proposed feature selection method removes 775 features to reduce the number of input features. The classification results on the hidden features extracted from joint hidden layer are shown in Table 7. Deep-FS removes features without affecting the classification performance of the testing data.

In Fig. 6 the errors of the Deep-FS method on the training and evaluation sets are compared to those of the baseline method, DBM [7], across various learning steps. In each learning step a new batch of data is trained. Eq. (17) is used to calculate the errors. Until step 5000 the two methods use the all the features so they reach the same error levels of 0.4228 and 0.2658 on training and evaluation sets respectively. In the next steps of the learning process, the errors of both methods are reduced, however, the error drops faster and reaches a final lower value when using the proposed method. For instance, the proposed method reaches the level of 0.2590 on the training set which is lower than that of the base method, i.e. 0.3075 (Fig. 6(a)) at the end of the learning steps. Similarly, the proposed method reaches the error level of 0.1745 compared to 0.2278 for the baseline method on evaluation set (Fig. 6(b)). The errors for the proposed Deep-FS method and the baseline DBM are 34% and 14% lower than the error at step 5000, i.e. 0.2658, respectively. The feature selection method removes the redundant and irrelevant features and consequently prevents over-

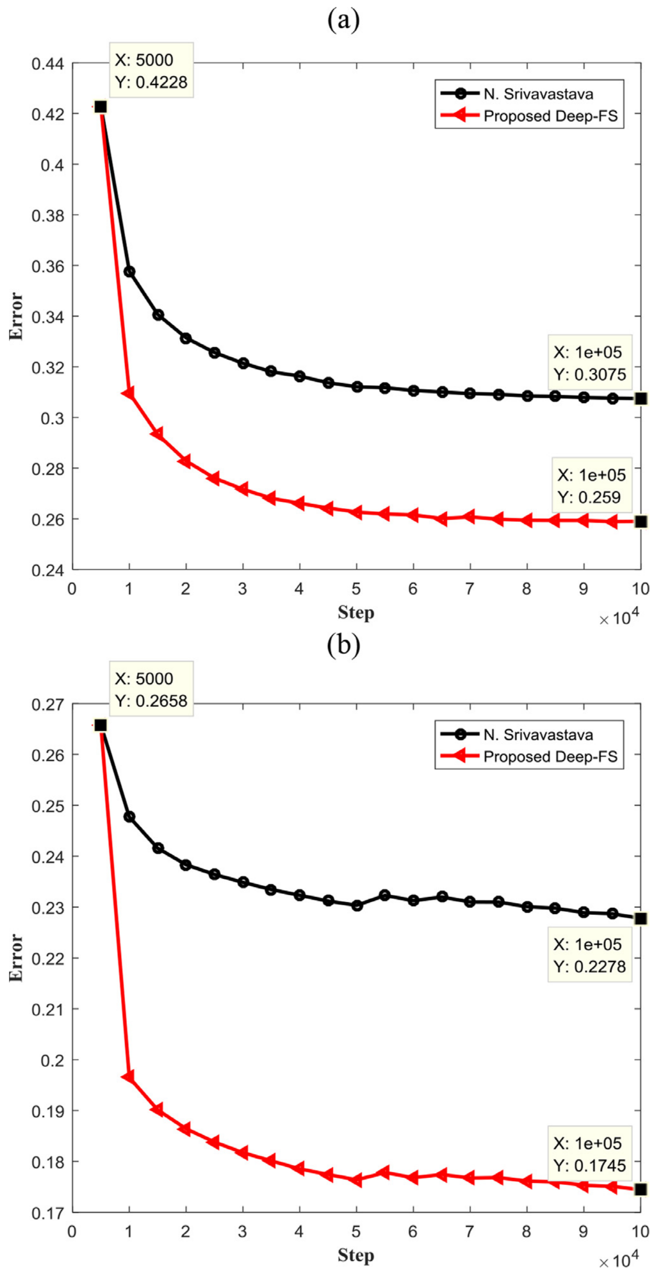


Fig. 6. Comparison of the error of Deep-FS against the error of the baseline DBM [7] on (a) training and (b) validation sets at different learning steps using the MIR-Flickr dataset. In each learning step a batch of training data is trained. The errors are reported after every 5000 steps. The proposed method significantly reduces the errors.

fitting the training data. The proposed method finds 775 irrelevant and redundant features. The removed features construct about 20% of the initial 3857 features.

4.3. Experimental results on the GISETTE dataset

The GISETTE dataset [45] is a benchmark dataset generated for binary classification tasks. GISETTE is a handwritten digit recognition dataset, which was part of the Advances in Neural Information Processing Systems (NIPS 2003) feature selection challenge. The GISETTE training data contains 6000 samples and 5000 features. The GISETTE learning task is a binary classification task to discriminate between two confusable handwritten digits of 4 and 9. In GISETTE, each digit has a dimension of 28 × 28. The pixels

Table 8
Performance comparison on the GISETTE dataset.

Method	# Features	10-fold classification accuracy (%)	Classifier training time in seconds (s)
No feature selection	5,000	92.9	262 s
Proposed Deep-FS	951	93.5	47 s

Table 9
Performance comparison on the MADELON dataset.

Method	# Features	10-fold classification accuracy (%)	Classifier training time in seconds (s)
No feature selection	500	79.4	6.34 s
Proposed Deep-FS	214	80.3	1.90 s

are normalised on the dataset to put their values in the [0,1] interval. The feature set contains the normalized values of pixels in addition to other features which have useful information for discriminating between digits 4 and 9. In the experiment on GISETTE, 6000 samples are used to train and test the proposed method. The Decision Tree classifier was adopted and k-fold cross validation with $k = 10$ was applied to evaluate the performance of the proposed feature selection method. The Decision Tree classifier was selected experimentally as it had achieved the highest classification accuracy compared to alternative conventional machine learning methods. Additionally, experiments showed that the Decision Tree classifier was trained in shorter time compared to most of the other methods. The number of splits in the Decision Tree was experimentally selected and set to 30. Table 8 shows that the proposed Deep-FS reduces the number of input features from 5000 to 951, i.e. reduction of 81% of input features. The accuracy of the Decision Tree classifier on the selected features is 93.5%. Its accuracy when using the entire set of features decreased to 92.9%. Using a smaller subset comprising the selected features reduced the training time of the classifier (see Table 8). The classifier needed about 262 s to train using all 5000 features, however, when the training was performed using the selected features (i.e. 951) features, it only needed 47 s to train. The proposed method reduced 82% of the classifier's training time (see Table 8).

4.4. Experimental results on the MADELON dataset

The MADELON dataset [45] is an artificial dataset, which was part of the Advances in Neural Information Processing Systems (NIPS 2003) feature selection challenge. This is a two-class classification problem with continuous input variables. The challenge is that the problem is multivariate and highly non-linear. In this experiment, 2000 training samples from the MADELON dataset are used, and each sample has 500 features. The performance of the proposed Deep-FS method on the MADELON dataset is reported in Table 9. Deep-FS reduces more than 57% of input features and achieves a higher classification accuracy, i.e. 80.3%. Additionally, it reduced the computation time of training the classifier, as reported in Table 9. It reduced 70% of the classifier training time. In summary, higher or very close accuracy is achieved using a much smaller set of features, but in less time (i.e. 4.44 fewer seconds) when Deep-FS is used.

4.5. Experimental results on the PANCAN dataset

PANCAN [46] was obtained from TCGA Pan-Cancer Clinical Data Resource (TCGA-CDR) [48]. The data contains 801 data samples from patients with 5 types of tumours: COAD, LUAD, PRAD, BRCA and KIRC. Each patient sample contains 20,531 features. A total of 264 features had the same value for all samples in the dataset and these were removed, resulting in a total number of 20,264 features. Table 10 shows thatDeep-FS has reduced the number of features from 20,264 to 4765, i.e. 76.49% reduction in the number of input features, and increased the accuracy from 97.1% to 98.5%. 10-fold cross validation was applied to achieve the results for each of the two situations reported in Table 10. Importantly, the results revealed significant reduction in the time needed by the classifier to train using the selected features. Training on the selected features needed 59.19 s compared to 400.39 s when all the features are used. Hence, training on the selected features was 341.20 s faster (i.e. it reduced 85% of the classifier's training time).

4.6. Comparison of Deep-FS with other feature selection approaches

In the following experiments the proposed Deep-FS is compared with other feature selection approaches. The comparison is performed in the following two steps. Step 1: Select features using the proposed and other existing feature selection algorithms; and Step 2: Train DBM using the selected subset of features.

Step 1: Initially, the proposed Deep-FS method and three other feature selection methods were separately applied to the MNIST dataset. Each feature selection method returned a selected subset of features, and then the selected features were used to train a DBM (results are presented in Table 11). Please note that most of the conventional existing feature selection algorithms are computationally very expensive and not suitable for large data. The Genetic Algorithm (GA) for feature selection described in [49], the Infinite Feature Selection (InfFS) [50], and the Laplacian Score (LS) for feature selection [51] methods were compared to Deep-FS. The GA-based mRMR freature selection algorithm [49] calculates the joint mutual information matrix between pairs of features and this makes the algorithm impractical for high dimensional datasets. InfFS [50] is a filter feature selection method that selects the most important features based on their ranks. All other features are considered to evaluate the score of a feature. InfFS maps the feature selection task to a graph and the feature selection is considered as a subset of features that make a path in the graph. A cost matrix is constructed to give pairwise associations between the features using variance and correlation of the features. The matrix is used to evaluate relevance and redundancy of a feature with respect to all the other features. Laplacian Score (LS) for feature selection [51] is another well-known method with the ability of finding features in unlabelled data [52,53]. The LS method uses a nearest neighbour graph to evaluate local geometric structures and selects features based on the constructed graph.

Step 2: The selected features identified by each feature selection method are used to train DBMs. The weights of each DBM are initialized randomly, then the DBM is trained on the selected fea-

Table 10
Performance comparison on the PANCAN dataset.

Method	# Features	10-fold classification accuracy (%)	Classifier training time in seconds (s)
No feature selection	20,264	97.1	400.39 s
Proposed Deep-FS	4765	98.5	59.19 s

Table 11
Experimental results with the MNIST dataset. Applying different feature selection methods and using the selected features to train a DBM. The number of selected features was set to 430. The training accuracy reached 100% for all the methods.

Feature Selection method	Number of misclassified images during testing (out of 10,000)	Processing time for FS in seconds (s)
GA [49] with DBM	184	30,784 s
InfFS [50] with DBM	156	151 s
Laplacian [51] with DBM	143	14,269 s
Proposed Deep-FS$_{10}$	90	133 s

Table 12
Improvement in the number of misclassified images and processing time of Deep-FS$_{10}$ vs other methods.

Feature selection method	Improvement in the number of misclassified images	Improvement in processing time in seconds (s)
Deep-FS$_{10}$ vs GA [49]	+94	+30,651 s (510.9 min)
Deep-FS$_{10}$ vs InfFS [50]	+66	+18 s
Deep-FS$_{10}$ vs Laplacian [51]	+53	+14,136 s (235.6 min)

tures. In the proposed Deep-FS$_{10}$, the weights which are trained during feature selection are used for initializing the DBM. The results are reported in Table 11.

Table 11 shows that the proposed Deep-FS$_{10}$ has achieved higher accuracy than the alternative approaches. Additionally, the proposed Deep-FS method can find the number of the selected features, i.e. 430, automatically. However, the other three feature selection methods require that the user specifies the number of selected features at the start of the feature selection procedure. The number of features (i.e. 430), obtained by the proposed method, is used by the other feature selection methods. The datasets are large and it is computationally very expensive to run the experiments using various numbers of features to experimentally determine the most suitable number of features to select. For this reason, there is a need for feature selection algorithms, such as Deep-FS, which can automatically identify the most relevant features in large data. The simulation results in Table 11 show that the proposed method has misclassified 90 images out of 10,000 (0.9% of the images) which is a lower error rate than the alternative methods. The training accuracy of the trained DBMs for all the methods is 100%. Table 11 also shows that the proposed feature selection method has the shortest processing time compared to the other methods. The GA [49] and Laplacian [51] methods need a much longer computation time to perform the feature selection task compared to Deep-FS. For instance, GA [49] took 30,784 s while the proposed method only took 133 s for the same feature selection task. The classical feature selection methods have high computational cost when applied to large datasets that have a high number of features and/or training samples. The improvement in performance and time when using the proposed Deep-FS$_{10}$ method instead of the other methods is provided in Table 12.

Most Classical feature selection methods have not been designed to work on datasets which contain a large number of features. Furthermore, classical feature selection methods have been designed to take as input a single matrix that contains all the training samples, and this is another reason which makes them unsuitable for large data. For instance, the unsupervised feature selection for multi-cluster data (MCFS) method proposed by Cai et.al. [54] was applied to the MNIST dataset, but computational problems were encountered. In particular, because MCFS constructs a square matrix with size $\boldsymbol{N} \times \boldsymbol{N}$, where $\boldsymbol{N}$ is the number of training samples, and there exist $N = 60{,}000$ image samples in MNIST, the square matrix was very big and MCFS could not converge when applied to the MNIST data. Other methods such as GA method can be applied to MNIST and other large datasets but have a high computation cost and computation time. However, the proposed Deep-FS overcomes the difficulties of working with datasets containing a high number of features and samples by dividing the training samples in a number of batches similar to what is performed in deep learning methods.

4.7. Time complexity analysis of the proposed method

In order to analyse the time complexity of the proposed method, two experiments are performed. The time complexity of the method is analysed in regard to the number of training samples and the number of input features.

In the first experiment, the computation times of the proposed Deep-FS method are obtained for different numbers of training samples. The MNIST dataset is used in the first experiment. The number of training samples is increased from 5000 to 50,000 in steps of 5000, and the running time of the proposed method for each number of training samples is calculated. Fig. 7 shows the computation time of the proposed Deep-FS against the number of training samples. A line with the equation of $T(n) = 0.0043n + 0.8140$ can fit to the data points. The equation shows that the time complexity of the proposed method in regard to the number of training samples is $O(n)$ in the big $O(.)$ notation.

In the second experiment for analysing the time complexity of the proposed method, the total number of input features is changed and the computation time of the proposed Deep-FS is calculated for the different number of input features. Uniformly distributed random datasets in [0,1] interval with different numbers of input features are generated to perform the second time com-

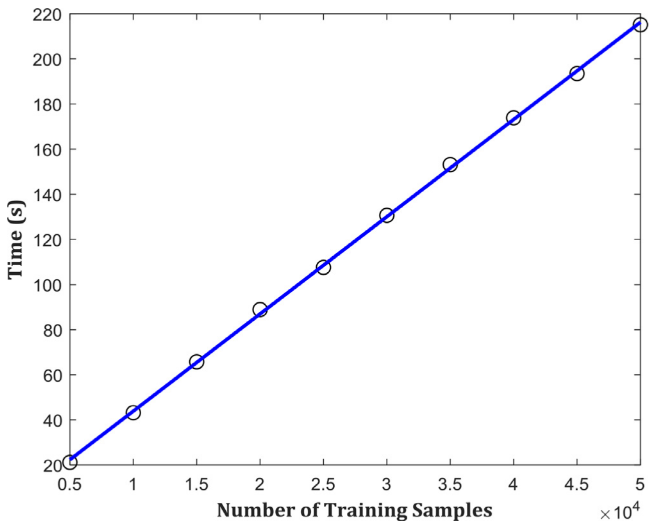


Fig. 7. Computation time of the proposed Deep-FS method when using various synthetic datasets each containing a different number of features. A line fits to the data points which are shown by 'O'.

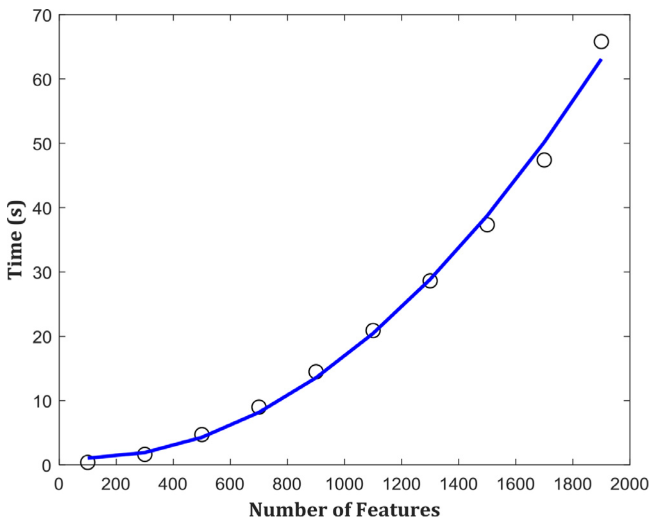


Fig. 8. Running time of the proposed Deep-FS method when using various synthetic datasets each containing a different number of features. A polynomial equation with the degree of 2 fits to the data points.

plexity analysis. There are 1000 training samples in the randomly generated dataset. The number of features are increased from 100 to 1900 with the interval of 200 features. Therefore, 10 random datasets, each with a different number of features. The running time of the proposed Deep-FS is calculated for each of the datasets and it is plotted in Fig. 8 with the sign 'o'. Then, a polynomial curve fits to the data points. The equation of the fitted curve is T(n) $=10^{-3}(0.0188n^2-3.2n+1171.8)$. The equation shows that the time complexity of the algorithm is $O(n^2)$ using the big $O(.)$ notation. The simulation results show that the time complexity of the proposed method in regard to the number of input features is higher that its complexity related to the number of training samples, i.e. $O(n^2)$ vs $O(n)$.

5. Conclusion

This paper proposes a novel feature selection algorithm, Deep-FS, which is embedded into the DBM classifier. DBM is considered as a non-linear feature extractor that can reduce the dimensionality of large or big data. The proposed Deep-FS feature selection method works in conjunction with the feature extraction method of DBM to reduce the number of input features and learning errors. Deep-FS uses a RBM that is trained during the training stage of a DBM to reduce computational cost. Deep-FS uses the generative property of RBMs which enables RBMs to reconstruct missing input features. A group of features is temporary eliminated from the input feature set to evaluate reconstruction error using a new criterion based on RBM. RBM treats the eliminated feature(s) as missing feature(s) and reconstructs these feature(s) by using the information from other input features. Then, the *reconstruction error* is used as a criterion for feature selection. The proposed feature selection method has two versions. In the first version of the proposed method, a RBM is initially trained, then it is used for feature selection. In the second version of the proposed method, the initially trained RBM is additionally trained on the reduced feature set during a feature selection procedure. Experiments revealed that the first version has a higher classification accuracy than the second version. Experiments also revealed that removing selected groups of features instead of single adjacent features improves performance and feature selection time.

Deep-FS was evaluated using the MNIST, MIR-Flickr, PANCAN, GISETTE and MADELON benchmark datasets. The results demonstrated that Deep-FS can reduce the number of inputs without affecting classification performance. Deep-FS reduced the number of misclassified samples on the MNIST data from 97 to 90. The proposed method automatically selected 430 features out of 784 features and it reduced the total processing time by 3063 s. When applied to the MIR-Flickr dataset it altered MAP from 0.476 to 0.478. The impact on classification accuracy is minor, which is a desirable result given that the number of inputs was reduced.

Moreover, Deep-FS has reduced the computation time. The proposed algorithm was effective in reducing the number of input features, i.e. it removed 15,499 features out of 20,264 features, and reduced classifier training time by 85% for the PANCAN dataset. Experiment results also revealed that the proposed feature selection method reduced the number of input features, improved cross validation accuracy, and reduced classifier training time on the GISETTE and MADELON datasets.

The proposed method was compared with three other feature selection methods namely the: GA [49], Infinite feature selection (InfFS) [50], and Laplacian Score for feature selection [51,55] using the MNIST dataset. The results showed that the proposed feature selection method reduced the number of misclassified images compared to the other methods. Additionally, it reduced the processing time of feature selection, for instance the proposed feature selection method performed automatic feature selection in 133 s while the GA method [49] performed the same feature selection task in 30,784 s.

Deep-FS can improve the processing ability of the deep learning method for multimodal data. Recently, Deep Neural Networks have shown their ability to process multimodal data with a large volume of data [1]. One common property of the multimodal data is their high dimensionality. Not all the input features might have useful information and irrelevant input features can introduce noise and degrade the performance. Reducing the number of input features and removing the irrelevant features can improve the ability of a deep learning model to process multimodal data. The feature selection method reduces computational cost by reducing the size of reconstructed matrix.

DBNs [1] belong to a group of DNNs that uses an unsupervised pre-training stage. During the first learning phase of DBNs, layer-wise unsupervised training is performed. Each layer learns a non-linear transformation from its input to capture the main information of its input. Each adjacent pair of layers is considered as an RBM [1,8]. An RBM is used to govern the unsupervised training and to extract features. The proposed feature selection method, which works based on RBM, can be applied to DNNs to improve its processing abilities. DBNs [1] have demonstrated good results in different applications such as speech recognition [11], audio [56], image and text classification [7]. To apply the proposed method to other deep learning methods, the proposed feature selection method can be initially used to select features which will be input into the deep learning classifier (or other classifier), and the classifier can be trained on the selected features.

Koller and Sahami's Markov Blanket filtering feature selection method [12,57] eliminates a feature if there is a Markov Blanket for the feature. For a target feature, a Markov Blanket is a minimal set of variables from a feature space on which all other variables are conditionally independent of the target feature. However, it is not straightforward to determine whether a set of features makes a Markov Blanket for the target feature, especially when the number of input features is high [12,57,58]. The proposed Deep-FS method defines a criterion for each feature and checks whether other features can reconstruct the target feature. In particular, with the proposed method, when the reconstruction error of a feature is reduced, the other features can contain a Markov Blanket for the target feature and that feature can be eliminated.

The proposed feature selection method will be very useful to researchers working with large and big data. Currently there are not many feature selection methods suitable for large data. The paper demonstrates that the proposed Deep-FS can be applied to unimodal and multimodal data for various tasks. In particular, Deep-FS has been applied to unimodal handwriting digit recognition datasets (MNIST, and GISETTE), a multi-modal dataset comprising images and text (MIR-Flickr), and a biomedical dataset (PANCAN).

Reducing the number of inputs and consequently the size of constructed weight matrix can be useful to manage limited hardware resources during hardware implementation of DNNs for complex tasks [1,59,60]. Deep-FS can be used to reduce the input size, and the trained network for a specific task can be implemented with less silicon area on hardware. Future work includes exploring the capability of the proposed Deep-FS in reducing the complexity of deep learning networks, through reducing the number of the input features in real world applications when the inputs are generated by sensors. Reduction of input features leads to the reduction of the number of sensors which can consequently reduce implementation costs. Deep-FS can offer a systematic way to find an optimized number of sensors. For example, Deep-FS can be applied to optimize the number and selected positions of sensors. Future work also includes applying the algorithm to large-scale data analytics tasks, such as human activity recognition which require use of deep learning algorithms.

Acknowledgements

The work was funded by The Leverhulme Trust Research Project Grant RPG-2016-252 entitled "Novel Approaches for Constructing Optimised Multimodal Data Spaces".

References

[1] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507, doi:10.1126/science.1127647.
[2] G.E. Hinton, S. Osindero, Y.W. Teh, A fast learning algorithm for deep belief nets, Neural Comput 18 (2006) 1527–1554, doi:10.1162/neco.2006.18.7.1527.
[3] G. Hinton, L. Deng, D. Yu, G.E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N. Sainath, B. Kingsbury, Deep neural networks for acoustic modeling in speech recognition, IEEE Signal Process. Mag (2012) 82–97, doi:10.1109/MSP.2012.2205597.
[4] H.Y. Xiong, B. Alipanahi, L.J. Lee, H. Bretschneider, D. Merico, R.K.C. Yuen, Y. Hua, S. Gueroussov, H.S. Najafabadi, T.R. Hughes, Q. Morris, Y. Barash, A.R. Krainer, N. Jojic, S.W. Scherer, B.J. Blencowe, B.J. Frey, The human splicing code reveals new insights into the genetic determinants of disease, Science 347 (6218) (2015). http://science.sciencemag.org/content/347/6218/1254806.abstract.
[5] E. Alpaydin, Introduction to Machine Learning, MIT press, 2014.
[6] G. Chandrashekar, F. Sahin, A survey on feature selection methods, Comput. Electr. Eng. 40 (2014) 16–28, doi:10.1016/j.compeleceng.2013.11.024.
[7] N. Srivastava, R. Salakhutdinov, Multimodal learning with deep Boltzmann machines, J. Mach. Learn. Res 15 (2014) 2222–2230. http://jmlr.org/papers/volume15/srivastava14b/srivastava14b.pdf.
[8] F. Liu, B. Liu, C. Sun, M. Liu, X. Wang, Deep belief network-based approaches for link prediction in signed social networks, Entropy 17 (2015) 2140–2169, doi:10.3390/e17042140.
[9] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, Adv. Neural Inf. Process. Syst. 19 (2007) 153.
[10] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (2014) 1929–1958, doi:10.1214/12-AOS1000.
[11] G.E. Dahl, D. Yu, L. Deng, A. Acero, Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition, IEEE Trans. Audio, Speech Lang. Process. 20 (2012) 30–42, doi:10.1109/TASL.2011.2134090.
[12] K. Javed, H.A. Babri, M. Saeed, Feature selection based on class-dependent densities for high-dimensional binary data, IEEE Trans. Knowl. Data Eng. 24 (2012) 465–477, doi:10.1109/TKDE.2010.263.
[13] J. Su, D.B. Thomas, P.Y.K. Cheung, Increasing network size and training throughput of FPGA restricted Boltzmann machines using dropout, in: Proceedings of the 24th IEEE International Symposium on Field-Programmable Custom Computing Machine, FCCM, 2016, 2016, pp. 48–51, doi:10.1109/FCCM.2016.23.
[14] N. Lopes, B. Ribeiro, Towards adaptive learning with improved convergence of deep belief networks on graphics processing units, Pattern Recognit 47 (2014) 114–127, doi:10.1016/j.patcog.2013.06.029.
[15] K. Ueyoshi, T. Marukame, T. Asai, M. Motomura, A. Schmid, Memory-error tolerance of scalable and highly parallel architecture for restricted Boltzmann machines in deep belief network, IEEE Int. Symp. Circuits Syst. (2016) 357–360, doi:10.1587/nolta.7.395.
[16] S.K. Kim, P.L. McMahon, K. Olukotun, A large-scale architecture for restricted Boltzmann machines, in: Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machine, FCCM, 2010, 2010, pp. 201–208, doi:10.1109/FCCM.2010.38.
[17] R. Salakhutdinov, G. Hinton, Deep Boltzmann machines, Aistats 1 (2009) 448–455, doi:10.1109/CVPRW.2009.5206577.
[18] S. He, S. Wang, W. Lan, H. Fu, Q. Ji, Facial expression recognition using deep Boltzmann machine from thermal infrared images, in: Proceedings of the 2013 Humaine Association Conference on Affecting Computing and Intelligent Interaction, 2013, pp. 239–244, doi:10.1109/ACII.2013.46.
[19] B. Leng, X. Zhang, M. Yao, Z. Xiong, A 3D model recognition mechanism based on deep Boltzmann machines, Neurocomputing 151 (2015) 593–602, doi:10.1016/j.neucom.2014.06.084.
[20] M.R. Alam, M. Bennamoun, R. Togneri, S. Member, F. Sohel, A Joint deep Boltzmann machine (jDBM) model for person identification using mobile phone data, IEEE Trans. Multimed. 19 (2017) 317–326.
[21] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, J. Mach. Learn. Res. 3 (2003) 1157–1182, doi:10.1016/j.aca.2011.07.027.
[22] J. Li, K. Cheng, S. Wang, F. Morstatter, R.P. Trevino, J. Tang, H. Liu, Feature selection: A data perspective, ACM Comput Surv (CSUR) 50 (6) (2017) 94. https://arxiv.org/abs/1601.07996.
[23] M. Yamada, J. Tang, J. Lugo-Martinez, E. Hodzic, R. Shrestha, A. Saha, H. Ouyang, D. Yin, H. Mamitsuka, C. Sahinalp, P. Radivojac, F. Menczer, Y. Chang, Ultra High-Dimensional nonlinear feature selection for big biological data, IEEE Trans Knowl Data Eng 14 (8) (2016) 1352–1365, doi:10.1109/TKDE.2018.2789451.
[24] K. Kira, L.A. Rendell, A practical approach to feature selection, in: Proceedings of the Ninth International Workshop on Machine Learning, 1992, pp. 249–256.
[25] M.A. Hall, Correlation-based feature selection of discrete and numeric class machine learning, in: Proceedings of international conference on machine learning (ICML), 2000, pp. 359–366.
[26] F. Fleuret, Fast binary feature selection with conditional mutual information, J. Mach. Learn. Res. 5 (2004) 1531–1555, doi:10.1007/s10182-011-0155-4.
[27] H. Peng, F. Long, C. Ding, Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy, IEEE Trans. Pattern Anal. Mach. Intell. 27 (2005) 1226–1238, doi:10.1109/TPAMI.2005.159.
[28] I. Guyon, J. Weston, S. Barnhill, V. Vapnik, Gene selection for cancer classification using support vector machines, Mach. Learn. 46 (2002) 389–422, doi:10.1108/03321640910919020.
[29] P. Ruangkanokmas, T. Achalakul, K. Akkarajitsakul, Deep belief networks with feature selection for sentiment classification, in: Proceedings of the 7th International Conference on Intelligent Systems, Modelling and Simulations, 2016, pp. 9–14, doi:10.1109/ISMS.2016.9.
[30] R. Ibrahim, N.A. Yousri, M.A. Ismail, N.M. El-Makky, Multi-level gene/MiRNA feature selection using deep belief nets and active learning, in: Proceedings of the 36th Annual International Conference on IEEE Engineering in Medicine and Biology Society, 2014, pp. 3957–3960, doi:10.1109/EMBC.2014.6944490.
[31] M.Z. Nezhad, D. Zhu, X. Li, K. Yang, P. Levy, SAFS: A deep feature selection approach for precision medicine, in: Proceedings of the IEEE International Conference on Bioinformatics and Biomedical, BIBM, 2016, 2017, pp. 501–506, doi:10.1109/BIBM.2016.7822569.
[32] M.Z. Nezhad, D. Zhu, N. Sadati, K. Yang, A Predictive Approach Using Deep Feature Learning for Electronic Medical Records: A Comparative Study, arXiv, 2018. http://arxiv.org/abs/1801.02961.

[33] Y. Li, C.-Y. Chen, W.W. Wasserman, Deep feature selection : theory and application to identify enhancers and promoters, J. Comput. Biol. 23 (5) (2015) 322–336, doi:10.13140/2.1.3673.6327.
[34] T. Zhang, Q. Wang, Deep learning based feature selection for remote sensing scene classification, IEEE Geosci. Remote Sens. Lett. 12 (2015) 1–5, doi:10.1109/LGRS.2015.2475299.
[35] Y. Saeys, I. Inza, P. Larranaga, A review of feature selection techniques in bioinformatics, Bioinformatics 23 (2007) 2507–2517, doi:10.1093/bioinformatics/btm344.
[36] A. Fischer, C. Igel, Training restricted Boltzmann machines: an introduction, Pattern Recognit 47 (2014) 25–39, doi:10.1016/j.patcog.2013.05.025.
[37] J. Chen, S. Cheng, H. Xie, L. Wang, T. Xiang, The equivalence of restricted boltzmann machines and tensor network states, Phys. Rev. B 97 (8) (2018) 085104, doi:10.1103/PhysRevB.97.085104.
[38] N. Srivastava, R. Salakhutdinov, Multimodal learning with deep Boltzmann machines, Adv. Neural Inf. Process. Syst. (2012) 2222–2230, doi:10.1109/CVPR.2013.49.
[39] G.E. Hinton, Training products of experts by minimizing contrastive divergence, Neural Comput 14 (2002) 1771–1800, doi:10.1162/089976602760128018.
[40] G.E. Hinton, Learning multiple layers of representation, Trends Cogn. Sci. 11 (2007) 428–434, doi:10.1016/j.tics.2007.09.004.
[41] N. Le Roux, N. Heess, J. Shotton, J. Winn, Learning a generative model of images by factoring appearance and shape, Neural Comput 23 (2011) 593–650, doi:10.1162/NECO_a_00086.
[42] Y. Tang, R. Salakhutdinov, G. Hinton, Robust Boltzmann machines for recognition and denoising, in: Proceedings of the IEEE Computer Society Conference Computer Vision and Pattern Recognition, 2012, pp. 2264–2271, doi:10.1109/CVPR.2012.6247936.
[43] B. Hassibi, D.G. Stork, G. Wolff, OptimalBrain Surgeon, Extensions and performance comparisons, in: J.D. Cowan, G. Tesauro, J. Alspector (Eds.), Advances in Neural Information Processing System 6, Morgan-Kaufmann, 1994, pp. 263–270 http://papers.nips.cc/paper/749-optimal-brain-surgeon-extensions-and-performance-comparisons.pdf.
[44] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE. 86 (1998) 2278–2323, doi:10.1109/5.726791.
[45] I. Guyon, S. Gunn, M. Nikravesh, L.A. Zadeh, Feature Extraction: Foundations and Applications, Springer, 2008.
[46] D.A. and fellow graduate Students, UCI Machine Learning Repository: gene expression cancer RNA-Seq Data Set, (n.d.). https://archive.ics.uci.edu/ml/datasets/gene+expression+cancer+RNA-Seq (accessed 23 May 2018).
[47] M.J. Huiskes, M.S. Lew, The MIR flickr retrieval evaluation, in: Proceeding of the 1st ACM International Conference on Multimedia Information Retrieval MIR, 2008, p. 39, doi:10.1145/1460096.1460104.
[48] Chang, et al., The Cancer Genome Atlas Pan-Cancer analysis project, Nat. Genet. 45 (2013) 1113–1120, doi:10.1038/ng.2764.
[49] O. Ludwig, U. Nunes, Novel maximum-margin training algorithms for supervised neural networks, IEEE Trans. Neural Networks. 21 (2010) 972–984, doi:10.1109/TNN.2010.2046423.
[50] G. Roffo, S. Melzi, M. Cristani, Infinite feature selection, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 4202–4210, doi:10.1109/ICCV.2015.478.
[51] X. He, D. Cai, P. Niyogi, Laplacian Score for Feature Selection, Adv. Neural Inf. Process. Syst. 18 (2005) 507–514. http://books.nips.cc/papers/files/nips18/NIPS2005_0149.pdf.
[52] Z. Zhao, H. Liu, Spectral feature selection for supervised and unsupervised learning, in: Proceedings of the 24th international conference on Machine learning, ACM, 2007, pp. 1151–1157, doi:10.1145/1273496.1273641.
[53] B. Wang, J. Zhu, E. Pierson, D. Ramazzotti, S. Batzoglou, Visualization and analysis of single-cell RNA-seq data by kernel- based similarity learning, Bioarxiv 1 (2016) 1–50, doi:10.1101/052225.
[54] D. Cai, C. Zhang, X. He, Unsupervised feature selection for multi-cluster data, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD, 2010, p. 333, doi:10.1145/1835804.1835848.
[55] S. Alelyani, J. Tang, H. Liu, Feature selection for clustering: a review, Data Clustering: Algorithms and Applications 29 (2013) 110–121.
[56] H. Lee, Y. Largman, P. Pham, a. Ng, Unsupervised feature learning for audio classification using convolutional deep belief networks, Adv. Neural Inf. Process. Syst. 22 (2009) 1096–1104, doi:10.1145/1553374.1553453.
[57] D. Koller, M. Sahami, Toward optimal feature selection, Int. Conf. Mach. Learn (1996) 284–292 doi:citeulike-article-id:393144.
[58] Y. Zeng, J. Luo, S. Lin, Classification using Markov blanket for feature selection, in: Proceedings of the IEEE International Conference on Granular Computing, 2009, pp. 743–747, doi:10.1109/GRC.2009.5255023.
[59] C. Wang, Q. Yu, L. Gong, X. Li, I.Y. Xie, DLAU: a scalable deep learning accelerator unit on FPGA, IEEE Trans. Comput. Des. Integr. Circuits Syst. 36 (2017) 513–517, doi:10.1109/TCAD.2016.2587683.
[60] D. Le Ly, P. Chow, D. Le Ly, P. Chow, D. Le Ly, P. Chow, High-performance reconfigurable hardware architecture for restricted boltzmann machines, IEEE Trans. Neural Networks. 21 (2010) 1780–1792, doi:10.1109/TNN.2010.2073481.

Aboozar Taherkhani received a B.Sc. degree in electrical and electronic engineering from Shahid Beheshti University, Tehran, Iran, an M.Sc. degree in biomedical engineering from the Amirkabir University of Technology, Tehran, and a Ph.D. degree from Ulster University, Londonderry, U.K., in 2017. He is currently a Research Fellow with the Computational Neuroscience and Cognitive Robotics Laboratory, Nottingham Trent University, Nottingham, U.K. His current research interests include artificial intelligence, deep neural networks, spiking neural network, and nonlinear signal processing.

Georgina Cosma received a Ph.D. degree in Computer Science from the University of Warwick, Coventry, UK, in 2008 and a First Class Honours BSc degree in Computer Science from Coventry University, Coventry, UK, in 2003. She is currently an Associate Professor at Nottingham Trent University, Nottingham, UK. Dr Cosma is a member of the IEEE Computer Society with Computational Intelligence, Big Data Community, and Brain Community memberships. She is recipient of The Leverhulme Trust project grant entitled “Novel Approaches for Constructing Optimised Multimodal Data Spaces.” Her research interests are in computational intelligence, nature-inspired feature selection, feature extraction, conventional machine learning and deep learning algorithms.

Martin McGinnity (SMIEEE, FIET) received a First Class (Hons.) degree in Physics in 1975, and a Ph.D. degree from the University of Durham, UK in 1979. He is currently a part-time Professor in Nottingham Trent University UK (NTU). Formerly he was Pro Vice Chancellor for Student Affairs and Head of the College of Science and Technology at NTU, Professor of Intelligent Systems Engineering and Director of the Intelligent Systems Research Centre in the Faculty of Computing and Engineering, University of Ulster. He is the author or co-author of over 300 research papers and has attracted over £25 million in research funding. His research interests are focused on computational intelligence, computational neuroscience, modelling of biological information processing and cognitive robotics.