Arabian Journal for Science and Engineering (2020) 45:1367–1377
https://doi.org/10.1007/s13369-019-03942-3

RESEARCH ARTICLE - ELECTRICAL ENGINEERING

Deep Neural Networks with Extreme Learning Machine for Seismic Data Compression

Hilal H. Nuha[1] · Adil Balghonaim[1] · Bo Liu[1] · Mohamed Mohandes[1] · Mohamed Deriche[1] · Faramarz Fekri[2]

Received: 23 December 2018 / Accepted: 20 May 2019 / Published online: 23 May 2019

Abstract
Advances on seismic survey techniques require a large number of geophones. This leads to an exponential growth in the size of data and prohibitive demands on storage and network communication resources. Therefore, it is desirable to compress the seismic data to the minimum possible, without losing important information. In this paper, a stacked auto-encoder extreme learning machine (AE-ELM) for seismic data compression is proposed. First, a deep asymmetric auto-encoder is constructed, in which nonlinear activation functions are used in the encoder hidden layers and linear activation functions are utilized in the decoder layers. Second, the encoder hidden layers are connected in a cascade way, so that outputs of a hidden layer are considered as the inputs to the succeeding hidden layer. Third, the optimal weights of connections between the layers of the decoder are solved analytically. Lastly, the AE-ELMs are stacked to create the complete encoder/decoder. The extreme learning machine (ELM) is selected due to its analytical calculation of weights efficient training that is suitable for practical implementation. In this neural network, data compression is achieved by transforming the original data through the encoder layers where the size of outputs from the last encoder hidden layer is smaller than the original data size. The proposed method exhibits a comparable reconstruction quality on a real dataset but with a much shorter training duration than other deep neural networks methods. This neural network with more than 8000 hidden units achieved 1.28×10^{-3} of normalized mean-squared error for 10:1 of compression ratio with only 8.23 s of training time.

Keywords Deep neural networks · Auto-encoder · Extreme learning machine · Seismic data compression

1 Introduction

With recent advances in oil and gas exploration, modern large scale seismic surveys may produce hundreds of terabytes of seismic recording data to be stored and transmitted for processing, interpretation, and action taking daily [1]. Meanwhile, storage capacity and computational power have roughly grown up by factors of 100 and 300 every ten years, respectively. However, network and input/output (I/O) storage speeds have only increased by a factor of 40 and 20 every ten years, respectively [2, 3]. Therefore, storage and communication processes become the major bottleneck for seismic data processing when dealing with large data volume. With the advances of the computational power, using data compression to reduce the data volume is desirable to mitigate the bottleneck.

Various seismic data compression techniques have been proposed over the last twenty years. In practice, even with specifically designed seismic data compression techniques, lossless schemes achieve no more than 3:1 of compression ratio [4]. Therefore, a lossy scheme with minimized data distortion is preferable to achieve higher compression ratios. Transformation techniques for lossy seismic data compression have been extensively investigated in the literature. The Walsh–Hadamard transform (WHT) combined with an interpolation scheme was used for seismic data compression in [5]. In [6], the authors investigated the discrete cosine transform (DCT) for seismic data compression and compared it with optimal linear transforms like principal component analysis (PCA). The transformation techniques with fixed basis like WHT and DCT may offer high compression ratio but require high number of transform components to pre-

✉ Mohamed Mohandes
mohandes@kfupm.edu.sa

[1] Center of Energy and Geo-signal Processing (CeGP), King Fahd University of Petroleum and Minerals (KFUPM), Dhahran 31261, Saudi Arabia

[2] School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, USA

serve the signal energy. Therefore, more adaptive transform basis was proposed for seismic data compression. Duval and Nguyen [7] took advantage of multi-resolution transformations offered by wavelets to achieve an efficient seismic data compression. Wang and Wu [8] used a so-called local sine/cosine transform to perform compression. Geng et al. [9] developed a technique called the dreamlet transform to compress seismic data prior to migration process [10]. More recently, a distributed principal component analysis (DPCA) compression was proposed for smart seismic acquisition networks in [11, 12], wherein the overall statistics of the seismic traces collected by all geophones in the acquisition network are used for global principal component (PC) estimation.

Machine learning approach has become one of the most prominent compression methods since it is able to model the structure of the data from multiple observations [13, 14]. Therefore, data compression techniques using machine learning can be found in many recent studies. A near lossless biomedical data compression is presented using neural networks (NNs) in [15]. Tan et al. [16] applied a multilayer auto-encoder for mammogram image compression. The training was performed using image patches instead of whole images. In [17], Srivastava proposed a deep belief network (DBN) with multimodal inputs to learn representation from data and image for information retrieval tasks. The DBN is a neural network architecture with multiple hidden layers. It is trained by decomposing the network into pairs of layers. Each pair is trained sequentially as a restricted Boltzmann machine (RBM) using contrastive divergence (CD). The first RBM, constructed from the input layer and the first hidden layer, uses the training data as inputs. The second RBM, constructed from the first and second hidden layers, uses the outputs from the first RBM as inputs, and so on. This allows the DBN to learn different feature representations on each layer. The RBMs are then stacked and fine-tuned using supervised learning with target outputs via regular back-propagation. In [18], Yann Olivier proved that minimizing the code length of the data and minimizing reconstruction error of an auto-encoder are highly correlated. In [19], the authors used an auto-encoder for electrocardiogram (ECG) compression. For seismic data, Huang [20] showed that NNs can be used to find the principal components for compression. In [21], the author utilized auto-associative NNs to perform seismic data compression.

The generalization capability of neural networks relies heavily on the learned features of the dataset [22]. Therefore, learning the features to represent the principal structure of the data is crucial [23]. However, feature detections require domain knowledge and human perception to capture relevant attributes [24, 25]. Moreover, the relevant features must be extracted from the data and transformed to achieve a better performance [24, 26]. In [27], the author used a texture-based segmentation to extract the feature of images. An optimized fuzzy cellular automata (FCA) algorithm was proposed for edge detection in [28]. Hinton et al. [29] proved that the RBM and auto-encoders are efficient in feature detections and can be used to train multi-layer NNs. NNs with multiple hidden layers are referred to as the deep neural networks (DNN) [30]. A stacked auto-encoder (SAE) is a class of DNN created by stacking several auto-encoders for dimensionality reduction. The DNN outperforms traditional support vector machine (SVM) and multi-layer perceptron (MLP) for big data, but suffers from lengthy learning durations [24].

Alternatively, the extreme learning machine (ELM) proposed by Huang et al. [31] using single-layer feed-forward neural networks (SLFN) offers a fast learning speed and a good generalization performance. The auto-encoder ELM (AE-ELM) was introduced by Kasun et al. [24], which is able to reproduce the input data after nonlinear random transformation. Similar to the DNN, the multi-layer ELM (ML-ELM) proposed in [24], performs layer-wise unsupervised training using weights obtained by the AE-ELM. For classification problems, ML-ELM learns significantly faster than existing DNN while outperforming DBN [30] and performing in par with deep Boltzmann machine (DBM) [32] for the MNIST dataset [33].

In this paper, we introduce the stacked auto-encoder ELM (SAE-ELM) for seismic data compression. This auto-encoder architecture is asymmetric where the encoder and decoder hidden layers use different types of activation functions. The training data are fed to the encoder part to obtain output values of the hidden layers sequentially. The decoder weights are then calculated using AE-ELM using the hidden layer values from the encoder. The performance of the proposed model is then compared with two existing SAE learning techniques namely SAE with multiple back-propagation (SAE-MBP) [34] and stacked RBM with single back-propagation (SRBM-SBP) [29]. For all learning models, identical network configurations and the discrete cosine transform (DCT) are used for comparisons. The performances are evaluated by taking the normalized mean-squared error (NMSE) as the metric to measure the data distortion.

The remainder of this paper is organized as follows. Section 2 discusses the existing auto-encoder learning techniques for data compression and fundamental ideas of the ELM. Section 3 presents the proposed model and learning development. Our proposed method is then evaluated and compared with some existing algorithms in Sect. 4. Conclusions are presented in Sect. 5.

2 Stacked Auto-Encoder and Extreme Learning Machine

In this section, we briefly introduce the fundamental techniques, which will be used to develop the proposed scheme.

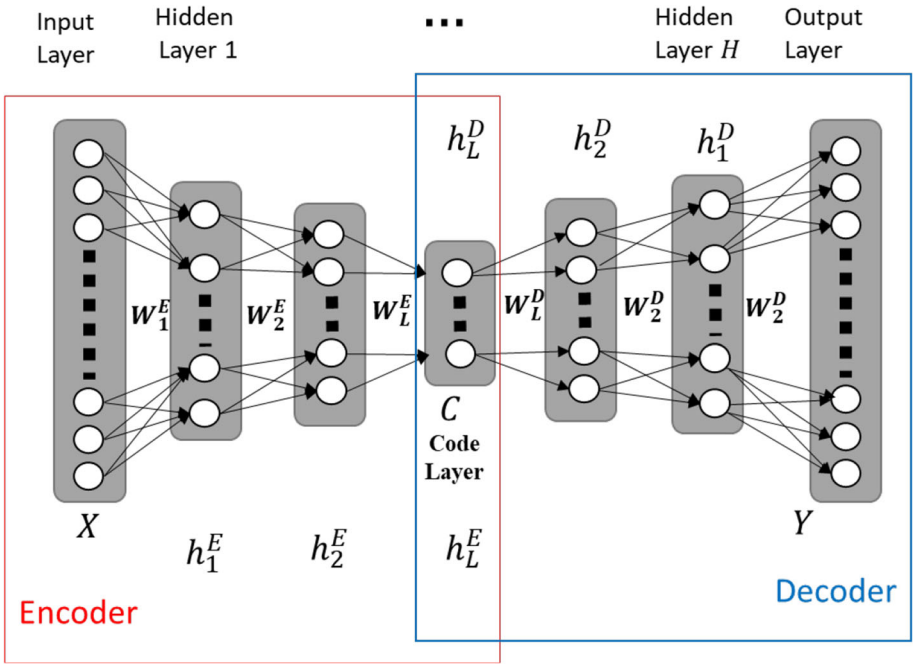

Fig. 1 Stacked auto-encoder

First, the rationale of data compression using SAE is described. Second, the basic idea of the ELM is presented. Finally, the extension of the ELM for auto-encoder is discussed.

2.1 Auto-Encoder for Data Compression

The SAE is a feed-forward neural network that can be used for dimensionality reduction with unsupervised learning. It consists of an input layer, a number of hidden layers, and an output layer as depicted in Fig. 1. The outputs of each layer are the inputs of the subsequent layer until reaching the output layer [14]. In the center of the hidden layers, a code or bottleneck layer is usually characterized by a layer with the smallest number of units. The encoder side performs the data compression with a specific compression ratio determined by the number of units in the code layer. The compressed data representation in the code layer is then reconstructed at the decoder side.

Given an SAE with L layers for encoding and L layers for decoding, and input signal $\boldsymbol{x}$ with n samples such that $\boldsymbol{x} = [x(1), x(2), \ldots, x(n)]^T$, the SAE attempts to reconstruct $\boldsymbol{x}$ at the output layer using two operations: compression and decompression, where the former is performed in the encoder side, and the latter is performed in the decoder side [14]. In the compression operation, each layer in the encoder sequentially transforms $\boldsymbol{x}$ to a compressed representation $\boldsymbol{z}$, at the code layer, such that: $\boldsymbol{z} = [z(1), z(2), \ldots, z(m)]^T$ where $m \ll n$. This operation yields a compression ratio of $m : n$. Each layer, say layer l, in the encoder produces an intermediate signal $\boldsymbol{z_l}$ by the following transformation:

$$z_l = f(\boldsymbol{W}_l z_{l-1} + \boldsymbol{b}_l), \tag{1}$$

where f denotes the activation function, $l = 1, \ldots, L$, $z_0 = \boldsymbol{x}$, z_l has m_l samples such that: $m = m_L, m_0 = n, z_L = \boldsymbol{z}$, $\boldsymbol{W}_l$ is a $m_l \times m_{l-1}$ weight matrix and $\boldsymbol{b}_l$ is a $m_l \times 1$ bias vector.

In the decoder part, the layers transform $\boldsymbol{z}$ to produce a reconstruction $\hat{\boldsymbol{x}}_l$ by the following transformation:

$$\hat{\boldsymbol{x}}_{\boldsymbol{l}} = f\left(\boldsymbol{W}_{l+1}^D \hat{\boldsymbol{x}}_{l+1} + \boldsymbol{b}_{l+1}^D\right), \tag{2}$$

where $l = [L-1, \ldots, 0]$, $\hat{\boldsymbol{x}}_L = \boldsymbol{z}$, $\hat{\boldsymbol{x}}_l$ has n_l samples such that: $m = n_L, n_l = n, \hat{\boldsymbol{x}}_0 = \hat{\boldsymbol{x}}$, $\boldsymbol{W}_l^D$ is a $n_l \times n_{l-1}$ weight matrix and $\boldsymbol{b}_l^D$ is a $n_l \times 1$ bias vector.

A single hidden layer of auto-encoder is not sufficient to model the features of data [29]. Therefore, an SAE with multiple hidden layers ($H \geq 2$) is typically used to perform seismic data compression. However, it is difficult to train SAEs with multiple hidden layers. The error gradients from the output layer gradually vanish while they are back-propagated layer by layer to the input layer, making the training extremely lengthy [16]. There are two major approaches to deal with this problem, namely SAE-MBP and SRBM-SBP. The SAE-MBP uses BP to train AE applied multiple times, i.e., once for each stacking and the final BP after stacking for fine-tuning [34]. BP is an extension of the delta rule to neural networks with multiple hidden layers. It utilizes the chain rule to iteratively compute gradients for

each layer by back-propagating the error from the output layer. Alternatively, the RBM is used to pre-train a DNN layer by layer in a supervised fashion. Given an input data, an RBM attempts to learn the probability distribution over the data using contrastive divergence (CD) by alternately sampling the value of the visible and hidden units of the RBM. The weights obtained by the RBM are then used as initial weights of a pair of layer in the DNN. The outputs of a layer become the input data for the next RBM to obtain initial weights for the next layer. The SRBM-SBP uses stacked RBM (SRBM) in which RBM is applied multiple times for each stacking and BP is applied only once at the end of the last stacking for fine tuning [29, 35].

2.2 Extreme Learning Machine

The ELM developed by Huang et al. [31] for SLFN shows that the hidden layer with M units with random weights can be used to transform the input data to M dimensional ELM random feature space. In the ELM, the network output is given as follows:

$$y(\boldsymbol{x}) = \sum_{i=1}^{M} \boldsymbol{\beta}_i z_i(\boldsymbol{x}) = \mathbf{z}(\mathbf{x})^{\mathbf{T}} \boldsymbol{\beta} \tag{3}$$

where $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_M]^T$ is the output weight matrix between the hidden and the output layers. $\mathbf{z}(\mathrm{x}) = [f_1(x), \ldots, f_M(x)]$ are the hidden layer outputs for the input x and $f_i(x)$ is the activation value of the i-th hidden unit. Given K training samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{K}$, the ELM is to solve the following linear equation:

$$\mathbf{Z}\boldsymbol{\beta} = \mathbf{y} \tag{4}$$

where $\boldsymbol{y} = [y_1, \ldots, y_K]^T$ denotes the target labels and $\mathbf{Z} = \left[\mathbf{z}^{\mathbf{T}}(x_1), \ldots, \mathbf{z}^{\mathbf{T}}(x_K)\right]^T$. The output weights $\boldsymbol{\beta}$ can be calculated by the following equation:

$$\boldsymbol{\beta} = \mathbf{Z}^{\dagger}\mathbf{y} \tag{5}$$

where $\mathbf{Z}^{\dagger}$ denotes the Moore–Penrose generalized inverse of matrix $\mathbf{Z}$.

A regularization term can be added to achieve a more robust solution with a better generalization performance [36] as follows:

$$\boldsymbol{\beta} = \left(\frac{\boldsymbol{I}}{C} + \boldsymbol{Z}^{\boldsymbol{T}}\boldsymbol{Z}\right)^{-1} \mathbf{Z}^{\boldsymbol{T}}\mathbf{y}. \tag{6}$$

where C is a regularization term.

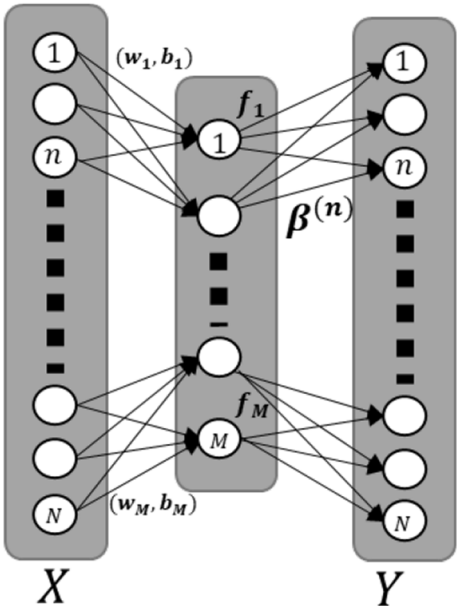

Fig. 2 An auto-encoder ELM

2.3 Auto-Encoder Using Extreme Learning Machine (AE-ELM)

Unsupervised learning has been extensively used to perform feature extractions [29, 37]. The AE-ELM was developed by Kasun et al. [24] as an extension to the regular ELM for feature representation via unsupervised learning. To construct an AE-ELM, the ELM is extended to perform unsupervised learning where input data is used as target output. Therefore, the number of units in the output layer is the same as the input layer.

Given M units in the hidden layer, the main goal of AE-ELM is to obtain meaningful features from the input data. By extending the ELM as a universal approximator [38], the network structure of the AE-ELM for feature representations is shown in Fig. 2. The AE-ELM has a capability of learning a meaningful feature representation [24]. A better performance can be achieved by choosing the random weights and random biases, which are orthogonal to each other.

In the AE-ELM, the hidden layer with orthogonal random weights and biases transforms the input data to an equal or different dimension spaces as shown by Johnson–Lindenstrauss lemma [39] as follows:

$$\begin{aligned} \mathbf{z} &= \boldsymbol{f}(\boldsymbol{wx} + \boldsymbol{b}) \\ \boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{W} &= I, b^T b = 1 \end{aligned} \tag{7}$$

where $\boldsymbol{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_M]$ denote the orthogonal random weights and $\boldsymbol{b} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_M]$ are the orthogonal random bias between the input units and hidden units.

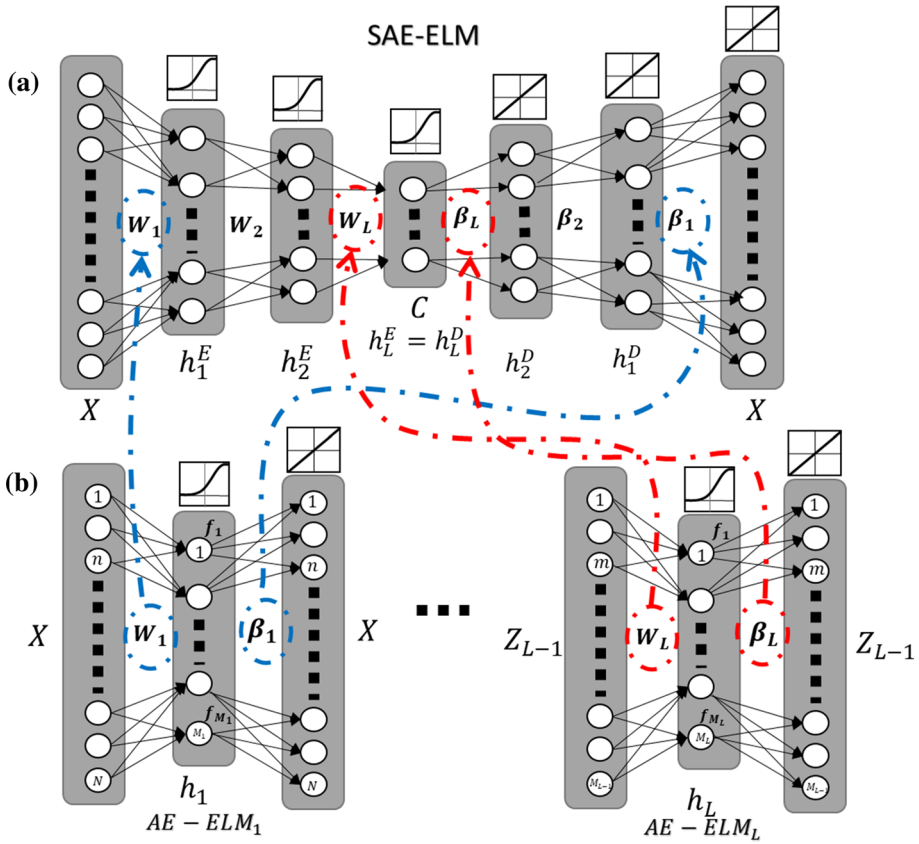

Fig. 3 SAE-ELM with **a** sequential trainings from AE-ELM$_1$ to AE-ELM$_L$, **b** stacking the AE-ELMs arranging weights $W_1 \ldots W_L$ and $\beta_L \ldots \beta_1$

The output weights $\boldsymbol{\beta}$ of the AE-ELM correspond to the transformation from the feature space to input data space. The output weights $\boldsymbol{\beta}$ are then obtained by:

$$\boldsymbol{\beta} = \left(\frac{\boldsymbol{I}}{C} + \boldsymbol{Z}^T\boldsymbol{Z}\right)^{-1}\mathbf{Z}^T\mathbf{X} \tag{8}$$

where $\boldsymbol{Z} = [z_1, \ldots, z_N]$ denote the hidden layer outputs of AE-ELM and $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$ are both input data and output data of the AE-ELM at the same time.

3 Proposed Model and Learning Method

To perform seismic data compression, we introduce an SAE model as depicted by Fig. 3a with asymmetric encoding–decoding operation. The asymmetric operation means that all layers in the encoder part of the model use nonlinear activation function, whereas all layers in the decoder counterpart use linear function without any biases. Given an input vector $\boldsymbol{x}$ and the reconstructed vector $\hat{\boldsymbol{x}}$, and SAE model with random encoder weights and biases $\{\boldsymbol{W}_1, \ldots, \boldsymbol{W}_L, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_L\}$, the main goal of the proposed learning method is to find the decoder weights $\{\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_L\}$ than minimize the discrepancies between $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$. The proposed asymmetric model allows us to obtain the analytical solution of the decoder weights using AE-ELM. The proposed learning scheme is expected to be faster than aforementioned SAE learning algorithms due to two factors. First, we only need to optimize the decoder weights, and secondly, each weight can be solved analytically without any iteration.

Given an input training data $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots \boldsymbol{x}_K\}$, the output of the l th encoder hidden layer $\boldsymbol{Z_l} = \{z_{l,1}, \ldots, z_{l,K}\}$ is given by (1). This operation is sequentially performed for each layer in the encoder until the code layer. The output of the code layer $\boldsymbol{Z} = \boldsymbol{Z_L}$ is then sequentially decoded by layers of the decoder part. The intermediate decoded vector $\hat{\boldsymbol{x}}_{\boldsymbol{l}}$ is the output vector of the l th hidden layer at the decoder part. The decoding operation is given as:

$$\hat{\boldsymbol{x}}_{\boldsymbol{l}} = \boldsymbol{\beta}_{l+1}\hat{\boldsymbol{x}}_{l+1}, \quad l = L - 1 \ldots 0 \tag{9}$$

where $\hat{\boldsymbol{x}}_{\boldsymbol{L}} = \boldsymbol{z}_{\boldsymbol{L}}$ is the output of the code layer and $\hat{\boldsymbol{x}}_{\boldsymbol{l}}$ denotes the output of the l th hidden layer at the decoder. The output of the final layer $\hat{\boldsymbol{X}} = \hat{\boldsymbol{X}}_0$ is the reconstructed data.

Suppose that the weights and biases in the encoder part satisfy the orthonormal condition in (7) and each output of the l th encoder hidden layer, $\boldsymbol{Z_l}$ is perfectly reconstructed by its intermediate decoded data $\hat{\boldsymbol{X}}_{\boldsymbol{l}}$. Then, each encoder–decoder parameter set $\{\boldsymbol{W_l}, \boldsymbol{b_l}, \boldsymbol{\beta}_l\}$ forms an AE-ELM with input data $\boldsymbol{Z_l}$ and output target $\hat{\boldsymbol{X}}_{\boldsymbol{l}} \approx \boldsymbol{Z_l}$ as depicted in Fig. 3b.

Therefore, given input data $\boldsymbol{X}$ and the output at the l th encoder hidden layer, $\boldsymbol{Z}_l, l = 1 \ldots L$, the weight $\boldsymbol{\beta_l}$ can be obtained as follows:

$$\boldsymbol{\beta}_l = \left(\frac{\boldsymbol{I}}{C} + \boldsymbol{Z}_l' \boldsymbol{Z}_l\right)^{-1} \boldsymbol{Z}_l' \hat{\mathbf{X}}_{l-1} \tag{10}$$

Since $\hat{\mathbf{X}}_{l-1}$ is the reconstructed data from $\boldsymbol{Z}_{l-1}$, then, Eq. (10) can be expressed as follows:

$$\boldsymbol{\beta}_l = \left(\frac{\boldsymbol{I}}{C} + \boldsymbol{Z}_l' \boldsymbol{Z}_l\right)^{-1} \boldsymbol{Z}_l' \boldsymbol{Z}_{l-1}. \tag{11}$$

Note that $\boldsymbol{Z_l}, l = 1 \ldots L$, must be calculated sequentially. However, once all $\boldsymbol{Z}_l$ are calculated, each $\boldsymbol{\beta}_l$ can be calculated independent from other $\boldsymbol{\beta}$.

The proposed scheme above is referred to as the stacked auto-encoder extreme learning machine (SAE-ELM) as illustrated in Fig. 3. The learning procedure is summarized as follows:

The SAE-ELM pseudo-algorithm

Input: training samples $\boldsymbol{X}$, encoder and decoder hidden layer number L, activation function $f(x)$, regularization parameter C.

1: Initialize an SAE with orthonormal random hidden unit parameters $(\boldsymbol{W}_i, \boldsymbol{b}_i), i = 1,2, \ldots L$.
2: Encoding: for $l = 1{:}L$
Compute the hidden layer output matrix $\boldsymbol{Z_l}$
$$\boldsymbol{z}_l = f(\boldsymbol{W_l z_{l-1}} + \boldsymbol{b}_l)$$
3: End for
4: Decoder weights: for $l = L{:}1$
Compute the decoder weights $\boldsymbol{\beta_l}$
$$\boldsymbol{\beta}_l = \left(\frac{\boldsymbol{I}}{C} + \boldsymbol{Z}_l' \boldsymbol{Z}_l\right)^{-1} \boldsymbol{Z}_l' \boldsymbol{Z}_{l-1}.$$
5: End for

4 Experimental Results

The East Texas, USA, dataset [40] consists of 594 seismic traces. The data are acquired with 33 vertical-component geophones with an interval of 220 ft. Each geophone records 18 traces originated by 18 dynamite explosions at a depth of 80–100 ft underground as depicted in Fig. 4. Each seismic trace has 1501 time samples acquired with 500 Hz of sampling frequency, which is approximately 3 s of recording length. Different geophones may record traces from the same shot with different magnitudes since the magnitude of the seismic wave is attenuated as it propagates to the geophones over distance. The magnitude of each trace is normalized to be between 0 and 1. Circular shift is used to align each trace such that the maximum value of the trace is located at the center of the trace. The maximum number of epochs for RBM is 20 with learning rate of 0.001 for code layer and 0.1 for other layers, whereas the BP uses maximum number of epochs of 500. The training is also terminated when the gradient is less than 10^{-5}. The regularization coefficient is fixed at $C = 10^8$ for all ELMs. In our model, the sigmoid activation functions are used for all hidden layers except the decoder side of the SAE-ELM since all AE-ELMs use linear activation function for outputs. All experiments are implemented using MATLAB with Intel i7 processor.

4.1 Representation Learning

To test the feature representation learning, we follow the setups in [24] by creating a mini dataset with 100 traces. Next, the mini dataset is used to train an AE-ELM (network configuration, input layer: 1501 units, code layer: 20 units, and output layer 1501 units, i.e., 1501-20-1501) and the contents of the output weights $\boldsymbol{\beta}$ are compared with BP and RBM with BP fine-tuning.

The weights learned by the BP and RBM are shown in Fig. 5a, b, respectively. Figure 5c displays the ELM output weights β instead of the input layer weights since the learning procedure is performed at the output layer weights and the input weights are regular random orthonormal transform basis. From top to down, all features are sorted based on their signal energy. It can be seen that the output weights β represent the features better than that of the RBM or BP based learning. The features learned using ELM show less noisy plots than that of the RBM or BP. The weights from our proposed method are more adapted to the dataset; therefore, it is expected to achieve a lower reconstruction error than that of other methods.

4.2 Optimal Architectures

In this experiment, we utilize 80% of traces for training and the rest 20% for testing. The normalized mean-squared error

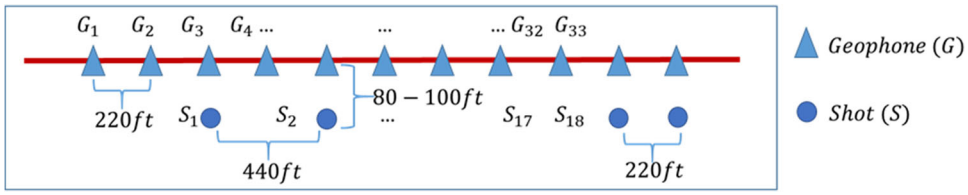

Fig. 4 Seismic acquisition layout

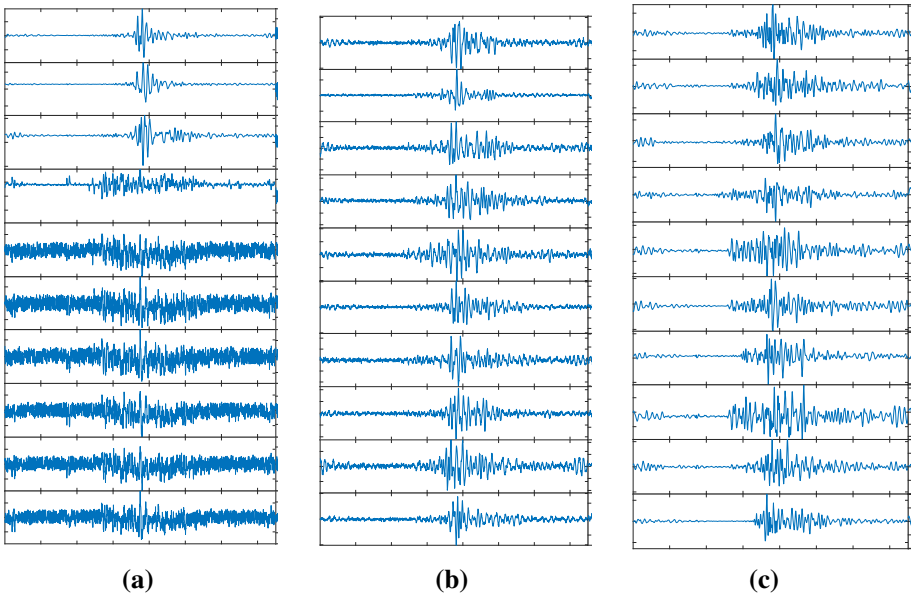

Fig. 5 Weights of **a** BP **b** RBM-BP **c**AE- ELM

(NMSE), as defined in the following equation, is used to measure the reconstructed data distortions.

$$\text{NMSE} = \frac{\sum_{i=1}^{K} \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|^2}{\sum_{i=1}^{K} \|\boldsymbol{x}_i\|^2}, \tag{12}$$

where K is the number of traces in the dataset. $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ denote the input vector and its corresponding reconstruction. In addition, normalized root mean-squared error (NRMSE) used by Rubin et al. [41] is used as a metric to measure the distortion level introduced by lossy compression.

$$\text{NRMSE} = \frac{\sqrt{\text{mean}\{(\mathbf{x} - \bar{\mathbf{x}})^2\}}}{\max(\mathbf{x}) - \min(\mathbf{x})} \tag{13}$$

where $\mathbf{x}$ is the data vector and $\bar{\mathbf{x}}$ is the mean vector.

Table 1 summarizes the experiment results to identify the optimum layer architecture for different compression ratios. The first column indicates the input and the encoder hidden layer sizes. For example, the layer size m_1-m_2-m_3-m indicates an auto-encoder with the architecture of m_1-m_2-m_3-m-m_3-m_2-m_1. The compression ratio is determined by the number of units at the code layer m. For example, SAE with architectures: 1501-1000-150 and 1501-150 uses the same number of units in the code layer $m = 150$, therefore the architectures lead to the same compression ratio $\frac{1501}{150} \approx 10:1$. On the other hand, architectures: 1501-1000-150 and 1501-1000-75 use different number of units in the code layer, i.e., $m = 150$ and $m = 75$, therefore the architectures yield 10:1 and 20:1 compression ratios, respectively.

From Table 1, one can notice that the training duration depends heavily on the number of units in the hidden layer. Obviously, architectures with many hidden layers and large number of units tend to require more time to be trained. However, for almost the same number of units, architectures of 1501-2500-1500-m-1500-2500-1501, on average, take longer training durations than that of the architectures of 1501-4000-m-4000-1501. It can also be noticed that increasing number of units at code layer does not increase the training duration significantly.

It can be noticed in Table 1, that the best results for each compression ratio are achieved by different architectures (as indicated by bold faces). The lowest NMSE for compression ratios 100:1 and 2:1 is achieved with a single hidden layer (with 1 5 and 7 50 units, respectively). However, for other compression ratios, the lowest NMSE is achieved by architectures with multiple hidden layers. The architectures with $m = 30$ and $m = 50$ corresponding to compression ratio of 50:1 and 30:1, respectively, exhibit the best performance when the first hidden layer is with $m_2 = 1000$ units, and do not show improvement despite the first hidden layer using $m_2 = 4000$ units. To the contrary, the best performance for

Table 1 Performance of SAE-ELM with different architectures and compression ratios

Architectures	Time (s)	NMSE (10^{-3})	NRMSE	CR
1501-15	0.0586	**10.01**	**0.0498**	100:1
1501-1000-15	1.303	10.91	0.0519	
1501-4000-15	6.555	10.43	0.0508	
1501-2500-1500-15	7.755	10.37	0.0506	
1501-25	0.104	8.27	0.0452	60:1
1501-1000-25	1.299	8.31	0.0453	
1501-4000-25	7.024	**7.89**	**0.0442**	
1501-2500-1500-25	7.909	7.998	0.0444	
1501-30	0.0987	7.602	0.0433	50:1
1501-1000-30	1.280	**7.11**	**0.0419**	
1501-4000-30	6.445	7.202	0.0422	
1501-2500-1500-30	7.994	7.534	0.0444	
1501-50	0.07	5.017	0.0352	30:1
1501-1000-50	1.308	**4.904**	**0.0348**	
1501-4000-50	6.147	5.020	0.0352	
1501-2500-1500-50	8.005	4.955	0.0350	
1501-75	0.0919	3.5	0.0295	20:1
1501-1000-75	1.3193	3.63	0.0300	
1501-4000-75	7.606	3.444	0.0292	
1501-2500-1500-75	8.049	**3.357**	**0.0288**	
1501-100	0.1133	2.515	0.0249	15:1
1501-1000-100	1.2756	2.599	0.0253	
1501-4000-100	7.1938	2.449	0.0246	
1501-2500-1500-100	8.205	**2.323**	**0.0239**	
1501-150	0.1427	1.512	0.0193	10:1
1501-1000-150	1.13	1.506	0.0193	
1501-4000-150	5.94	1.374	0.0184	
1501-2500-1500-150	8.235	**1.282**	**0.0178**	
1501-300	0.1901	0.495	0.0111	5:1
1501-1000-300	1.3567	0.337	0.0091	
1501-4000-300	6.889	**0.302**	**0.0086**	
1501-2500-1500-300	8.6607	0.393	0.0099	
1501-500	0.3954	0.175	0.0066	3:1
1501-1000-500	1.3579	0.13	0.0059	
1501-4000-500	6.92	**0.125**	**0.0056**	
1501-2500-1500-500	8.8095	0.201	0.0070	
1501-750	0.830	**0.072**	**0.0042**	2:1
1501-1000-750	1.9921	0.077	0.0044	
1501-4000-750	7.9148	0.082	0.0045	
1501-2500-1500-750	9.1623	0.143	0.0060	

architectures with $m = 25, 300$, and 500 is achieved when $m_2 = 4000$ units. It can be seen that for compression ratios of 20:1, 15:1, and 10:1, the architectures with five hidden layers (1501-2500-1500-m-1500-2500-1501) achieve lower NMSE than that of three hidden layers (1501-4000-m-4000-1501) despite having almost the same number of units in hidden layers (> 8000). As can be seen from Table 1, the results obtained using the NRMSE are all consistent with the ones using the NMSE measure. A sample of the original and the reconstructed seismic trace using SAE-ELM with 10:1 of compression ratio and 1.28×10^{-3} of NMSE is shown in Fig. 6.

4.3 Comparison with Other Methods

In this section, we compare the proposed method with the existing SAE learning methods, i.e., SRBM-SBP and SAE-MBP for data compression. The best results for each learning method are presented. In addition to that, we also use the discrete cosine transform (DCT) as a comparative method for seismic data compression with linear transformations. The DCT was reported to achieve the closest performance to the optimal linear transform [6].

Figure 7 shows the signal to noise ratio (SNR) in decibel (dB) from all techniques for different compression ratios. The SNR is related to NMSE as follows:

$$\text{SNR (dB)} = -\log_{10} \text{NMSE}. \tag{14}$$

It can be seen that, for compression ratios of 100:1 to 15:1, the SRBM-SBP achieves a higher SNR than other methods. However, the SRBM-SBP fails to improve the SNR for compression ratios less than 10:1. This indicates that SRBM-SBP performs poorly for low compression ratios compared to other methods. For the DCT, the SNR does not increase significantly for high compression ratios. However, for a very low compression ratio of 2:1, the DCT performs the best result. The SAE-MBP exhibits stagnant performance for all compression ratios where the SNR does not increase significantly despite more hidden units are added in the code layer. The proposed method exhibits comparable performance with other methods for different compression ratios but shows the best performances for compression ratios of 10:1 to 3:1. For example, for 10:1 and 5:1, the proposed method shows improvements over SRBM-SBP by 1 dB and 4 dB of SNR, respectively. Moreover, in our experiments, despite using different initial random weights, for the same configuration, the SAE-ELM achieves very similar performance, therefore, the proposed method is robust to the initial weights.

Table 2 exhibits the training durations of the architectures that achieve the best performances corresponding to Fig. 7. High compression ratios tend to require longer training times. However, for high compression ratio like 100:1,

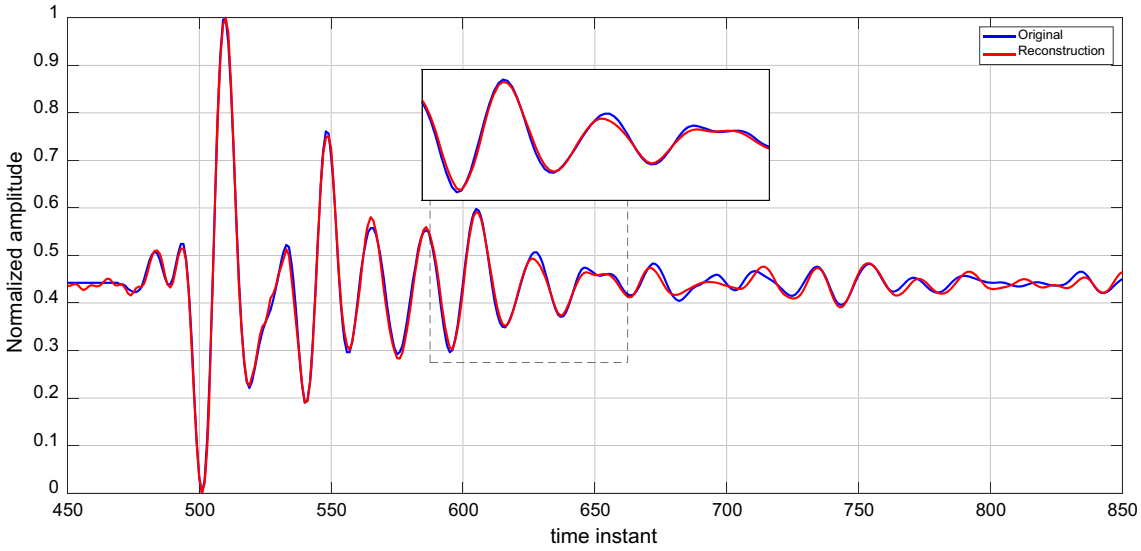

Fig. 6 Reconstruction sample with 10:1 compression ratio

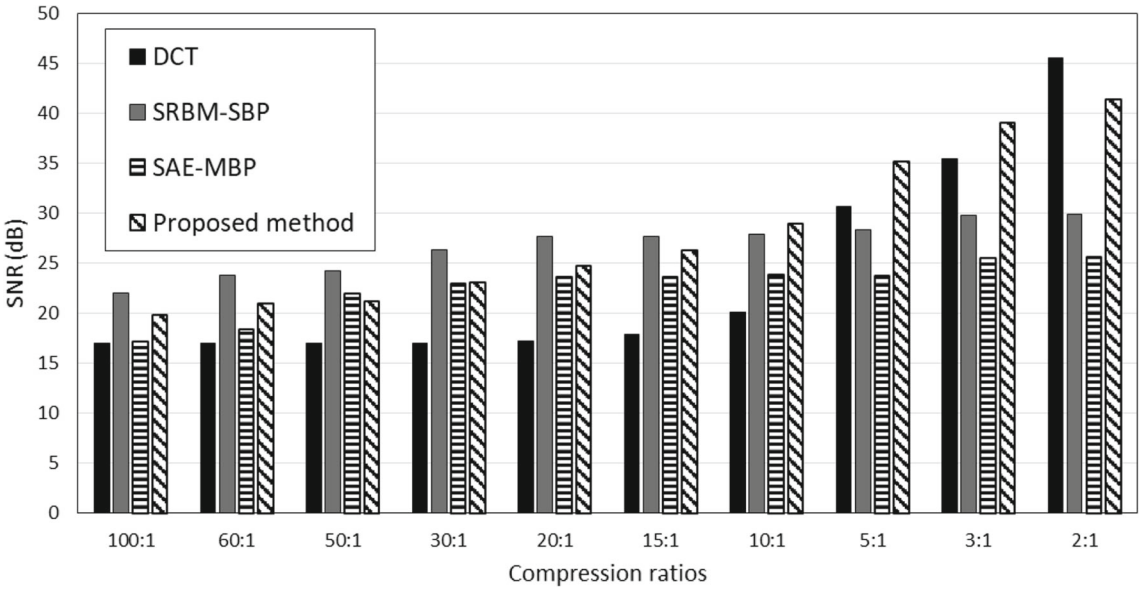

Fig. 7 SNR as a function of compression ratios for all methods

the SAE-MBP tends to terminate the training prematurely due to low gradient that result in short training durations. For each compression ratio, it can be seen that the proposed method achieves the lowest training duration. For example, for 10:1 of compression ratio, the proposed method completes the optimization in 8.2 s, whereas SRBM-SBP and SAE-MBP require 159.42 s and 8.235 s, respectively.

5 Conclusions

In this paper, we presented an asymmetric auto-encoder neural network architecture for seismic data compression namely the stacked auto-encoder extreme learning machine (SAE-ELM). First, an SAE was constructed with the following configuration: the encoder part of the network used nonlinear activation functions, whereas the decoder part used linear activation functions. Second, all encoder hidden layer output values were sequentially calculated. Third, the decoder weights were calculated using AE-ELM given the hidden layer output values. Data compression is achieved by transforming the original data through the encoder layers so that the data size is sequentially decreased. We compared our proposed method with other state of the art SAE models using a real seismic dataset. Our proposed method achieved comparable performances to the other methods in term of

Table 2 Time performance comparison of the SAE-ELM using East Texas testing data for 10:1 compression ratio

Compression ratios	Time durations (s)		
	SRBM-SBP	SAE-MBP	Proposed method
100:1	18.48	8.14	0.0586
60:1	41.71	15.18	7.02
50:1	55.01	42.66	1.280
30:1	57.98	56.69	1.308
20:1	69.91	69.75	8.049
15:1	96.94	99.68	8.205
10:1	159.42	138.2	8.235
5:1	233.41	231.3	6.889
3:1	361.22	411.1	6.92
2:1	558.34	550.3	0.830

reconstruction quality. It achieved 1.28×10^{-3} of NMSE for 10:1 of compression ratio with a significantly shorter training time. For the same compression ratios, the SAE-ELM also achieved better reconstruction quality than that of the linear transform like the discrete cosine transform (DCT) on average.

Acknowledgements This work is supported by the Center for Energy and Geo Processing (CeGP) at King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia, under Project GTEC1801.

References

1. Zhang, Y. Large-scale seismic data compression: application to full waveform inversion and extended image volume, Doctoral dissertation, University of British Columbia, April, 2018
2. Roberts, L.: Beyond Moore's law: internet growth trends. Computer **33**(1), 117–119 (2000)
3. Kuhn, M.; Kunkel, J.; Ludwig, T.: Data compression for climate data. Supercomput. Front. Innov. **3**(1), 75–94 (2016)
4. Rosten, T.; Marthinussen, V. A.; Ramstad,T. A.; Perkis, A.: Filter bank optimization for high-dimensional compression of pre-stack seismic data. In: Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, March 1999, 1999
5. Wood, L.C.: Seismic data compression methods. Geophysics **39**, 499–525 (1974)
6. Spanias, A.S.; Jonsson, S.B.; Stearns, S.D.: Transform methods for seismic data compression. IEEE Trans. Geosci. Remote Sens. **29**, 407–416 (1991)
7. Duval, L. C.; Nguyen, T. Q.: Seismic data compression: a comparative study between GenLOT and wavelet compression. In: Proceedings of SPIE- the international society for optical engineering, October 1999, 1999
8. Wang, Y.; Wu, R.-S.: Seismic data compression by an adaptive local cosine/sine transform and its effects on migration. Geophys. Prospect. **48**, 1009–1031 (2000)
9. Geng, Y.; Wu R. S.; Gao, J.: Dreamlet transform applied to seismic data compression and its effects on migration. In: SEG Annual Meeting January, Houston, 2009
10. Wang, S.; Li, J.; Chiu, S.; Anno, P.: Seismic data compression and regularization via wave packets. In In SEG technical program expanded abstracts October, 2010
11. Liu, B.; Mohandes, M.; Nuha, H.; Mohamed, D.; Faramarz, F.: A distributed principal component analysis compression for smart seismic acquisition networks. IEEE Trans. Geosci. Remote Sens. **56**(6), 3020–3029 (2018)
12. Liu, B.; Nuha, H.; Mohandes, M.; Deriche, M.; Fekri, F.: Disributed principal component analysis for data compression of sequential seismic sensor arrays. In: SEG technical program expanded abstracts October 2016, 2016
13. Said, A.; Mohamed, A.; Elfouly, T.; Harras, K.; Wang, Z.: Multimodal deep learning approach for joint EEG-EMG data compression and classification. In: Wireless Communications and Networking Conference (WCNC) May, 2017
14. Al-Said, M.T.M.; Abdellatif, A.; Mohamed, A.; Elfouly, T.; Harras, K.; O'Connor, M.: A deep learning approach for vital signs compression and energy efficient delivery in mHealth systems. IEEE Access **6**, 33727–33739 (2018)
15. Sriraam, N.; Eswaran, C.: Performance evaluation of neural network and linear predictors for near-lossless compression of EEG signals. IEEE Trans. Inf. Technol. Biomed. **12**(1), 87–93 (2008)
16. Tan, C.; Eswaran, C.: Using autoencoders for mammogram compression. J. Med. Syst. **35**(1), 49–58 (2011)
17. Srivastava, N.; Salakhutdinov, R. R.: Multimodal learning with deep boltzmann machine. In: Advances in Neural Information Processing Systems December, 2012
18. Ollivier, Y.: Auto-encoders: reconstruction versus compression. *[Online]* https://hal.archives-ouvertes.fr/hal-01104268, 2014.
19. Del Testa, D.; Rossi, M.: Lightweight lossy compression of biometric patterns via denoising autoencoders. IEEE Signal Process. Lett. **22**(12), 2304–2308 (2015)
20. Huang, K.-Y.: Neural networks for seismic principal components analysis. IEEE Trans. Geosci. Remote Sensing **37**, 297–311 (1999)
21. Nuha, H.; Mohandes, M.; Liu, B.: Seismic data compression using auto-associative neural network and restricted Boltzmann machine. In: SEG Technical Program Expanded Abstracts October, Anaheim, 2018
22. Akbarizadeh, G.; Rangzan, K.; Kabolizadeh, M.: Effective supervised multiple-feature learning for fused radar and optical data classification. IET Radar Sonar Navig. **11**(5), 768–777 (2016)
23. Sharifzadeh, F.; Akbarizadeh, G.; Kavian, Y.: Ship classification in SAR images using a new hybrid CNN–MLP classifier. J. Indian Soc. Remote Sens. **47**, 1–12 (2018)
24. Kasun, L.L.C.; Zhou, H.; Huang, G.-B.; Vong, C.M.: Representational learning with extreme learning machine for big data. IEEE Intell. Syst. **28**(6), 31–34 (2013)
25. Akbarizadeh, G.: A new statistical-based kurtosis wavelet energy feature for texture recognition of SAR images. IEEE Trans. Geosci. Remote Sens. **50**(11), 4358–4368 (2012)
26. Raeisi, A.; Akbarizadeh, G.; Mahmoudi, A.: Combined method of an efficient cuckoo search algorithm and nonnegative matrix factorization of different zernike moment features for discrimination between oil spills and lookalikes in SAR Images. IEEE J. Sel. Topics Appl. Earth Obs. Remote Sens. **99**, 1–13 (2018)
27. Modava, M.; Akbarizadeh, G.; Soroosh, M.: Integration of spectral histogram and level set for coastline detection in SAR images. IEEE Trans. Aerosp. Electron. Syst. **55**(2), 810–819 (2019)
28. Farbod, M.; Akbarizadeh, G.; Kosarian, A.; Rangzan, K.: Optimized fuzzy cellular automata for synthetic aperture radar image edge detection. J. Electron. Imaging **27**(1), 13–30 (2018)
29. Hinton, G.E.; Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006)
30. Hinton, G.; Deng, L.; Yu, D.; Dahl, G.E.; Mohamed, A.-R.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen, P.; Sainath, T.N.; et al.: Deep neural networks for acoustic modeling in speech recognition:

the shared views of four research groups. IEEE Signal Process. Mag. **29**, 82–97 (2012)
31. Huang, G.-B.; Zhu, Q.-Y.; Siew, C.-K.: Extreme learning machine: theory and applications. Neurocomputing **70**, 489–501 (2006)
32. Salakhutdinov, R.; Larochelle, H.: Efficient learning of deep Boltzmann machines. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics May, 2010.
33. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
34. Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y.; Manzagol, P.: Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. J. Mach. Learn. Res. **11**, 3371–3408 (2010)
35. Tee, Y.W.; Hinton, G.E.: Rate-coded restricted Boltzmann machines for face recognition. Neural Inf. Process. Syst. **13**, 908–914 (2000)
36. Huang, G.; Zhou, H.; Ding, X.; Zhang, R.: Extreme learning machine for regression and multiclass classification. IEEE Trans. Syst. Man Cybern. Part B (Cybern.) **42**(2), 513–529 (2012)
37. Akbarizadeh, G.; Moghaddam, A.: Detection of lung nodules in CT scans based on unsupervised feature learning and fuzzy inference. J. Med. Imaging Health Inf. **6**(2), 477–483 (2016)
38. Huang, G.; Chen, L.; Siew, C.: Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans. Neural Netw. **17**(4), 879–892 (2006)
39. Johnson, W.; Lindenstrauss, J.: Extensions of Lipschitz mappings into a Hilbert space. Contemp. Math. **26**(1), 189–206 (1984)
40. Mousa, W.A.; Al-Shuhail, A.A.: Processing of seismic reflection data using MATLAB™. Synth. Lect. Signal Process. **5**, 1–97 (2011)
41. Rubin, M.J.; Wakin, M.B.; Camp, T.: Lossy compression for wireless seismic data acquisition. IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens. **9**, 236–252 (2016)