Sparse Autoencoders Using Non-smooth Regularization

Sajjad Amini
Department of Electrical Engineering
and Electronics Research Institute (ERI),
Sharif University of Technology, Tehran, Iran
Email: s_amini@ee.sharif.edu

Shahrokh Ghaemmaghami
Department of Electrical Engineering
and Electronics Research Institute (ERI),
Sharif University of Technology, Tehran, Iran
Email: ghaemmag@sharif.edu

***Abstract*—Autoencoder, at the heart of a deep learning structure, plays an important role in extracting abstract representation of a set of input training patterns. Abstract representation contains informative features to demonstrate a large set of data patterns in an optimal way in certain applications. It is shown that through sparse regularization of outputs of the hidden units (codes) in an autoencoder, the quality of codes can be enhanced that leads to a higher learning performance in applications like classification. Almost all methods trying to achieve code sparsity in an autoencoder use a smooth approximation of ℓ_1 norm, as the best convex approximation of pseudo ℓ_0 norm. In this paper, we incorporate sparsity to autoencoder training optimization process using non-smooth convex ℓ_1 norm and propose an efficient algorithm to train the structure. The non-smooth ℓ_1 regularization have shown its efficiency in imposing sparsity in various applications including feature selection via lasso and sparse representation using basis pursuit. Our experimental results on three benchmark datasets show superiority of this term in training a sparse autoencoder over previously proposed ones. As a byproduct of the proposed method, it can also be used to apply different types of non-smooth regularizers to autoencoder training problem.**

I. Introduction

Being abstract is favorable for a representation [1]. The performance of a machine learning system depends heavily on the abstraction level of input pattern representation [2]. Visual perception system also utilizes abstract representations [3]. Many researchers put their attention on designing automatic systems for (abstract) representation learning. Toward this objective, the first step is to define an equivalent (necessarily measurable) for abstract (good) representation. In [2] several worthwhile equivalents have been reviewed, where a simple equivalent is: "*A good representation is one that is useful as input to a supervised predictor.*". This measure is widely used in various works to show the efficiency of their proposed representation learning systems.

Depth of the model selected to learn representation is an important factor. Based on complexity theory, deep architectures can be much more efficient than shallow architectures, in term of computational elements required to represent a function [4]. Consequently deep architectures are more favorable for representation learning but until recently, they suffer from getting stock in poor solutions when they are trained using gradient based methods with random internalization. *Deep learning* brings an interesting solution to this problem. The first idea was to add a pretraining step prior to training a structure using probabilistic models, known as restricted Boltzmann machine (RBF) [5]. Subsequent improvement was achieved when deterministic models named autoencoders were introduced for pretraining [6], [7].

Autoencoders consist of two parameterized deterministic functions namely encoding and decoding. Encoding function (or equivalently **encoder**) denoted by $f_e(\cdot;\theta_1)$ is a straightforward efficient mapping from input space representation to abstract representation (or **code**) where θ_1 represents the encoder parameters. The code corresponding to a sample input pattern $\mathbf{x}$ is easily computed as: $\mathbf{z} = f_e(\mathbf{x};\theta_1)$. A closely related function to the encoder is a decoder which constitutes a mapping from code space to input space to reconstruct input pattern and is denoted by $f_d(\cdot;\theta_2)$. The reconstructed pattern is computed as: $\hat{\mathbf{x}} = f_d(\mathbf{z};\theta_2)$. Training an autoencoder is the process of determining parameters θ_1 and θ_2 so as to minimize a distance measure between input pattern and its reconstructed version at the output of the decoder over a training set. The distance measure is usually squared reconstruction error or binary cross-entropy in the case of inputs with binary nature.

One widely used structure for encoder and decoder is affine mapping followed by optional non-linearity such as sigmoid or hyperbolic tangent. With this structure, autoencoder resembles a two layer neural network and training methods of neural networks could be utilized to train an autoencoder.

One major threat in training an autoencoder is getting stock in identity solution, a global minimizer which does not reveal the underlying structure of input training patterns. In order to solve this problem, constraint(s) should be added to autoencoder learning procedure. Two types of widely used constraints for the problem are: **1)Structural constraints:** These constraints are imposed on the structure of autoencoder. An example of this type of constraints is compressive autoencoder where code dimension is smaller than input dimension. This autoencoder tries to find a compressed representation of input pattern in code space. **2)Mathematical constraints:** Constraints of this type are imposed on the learning optimization process using regularizers. Sparsity as a mathematical constraint (regularizer) has been applied to autoencoders through various formulations that have come to

ISBN 978-90-827970-1-5

favorable results. In one of the earliest papers which utilizes autoencoders to build a deep architecture [4], the weights of encoding and decoding are tied to restriction of capacity. Ranzato *et al.* have shown that tying weights will introduce a form of sparsity to the problem [8]. Two other effective methods of direct imposition of sparsity on autoencoder have also been proposed. The first method penalizes hidden unit biases [8] while, in the second method, codes are directly penalized to be sparse [9], [10], [11]

Penalizing biases come with the deficiency that the weights could compensate penalized biases effect. Consequently, the method of penalizing code has been found more favorable than the first method. A most straightforward method to have a spare autoencoder is to penalize the pseudo ℓ_0 norm of code vectors but, due to the discrete nature of this pseudo norm, the resulting problem may become intractable in most of real world applications [12]. To overcome this problem ℓ_1 norm, the best convex approximation of pseudo ℓ_0 norm, is used to penalize the code vectors. Using ℓ_1 norm of the code vectors makes the cost function of autoencoder non-smooth which limits the use of efficient gradient based optimization methods. Consequently, few papers employ ℓ_1 norm as regularizer and try some closely related smooth approximation to this norm.

In [13], *Student-t* penalty is suggested as a smooth approximation of ℓ_1 norm. Another approach to sparse autoencoders is to penalize the average output of code elements over training patterns by using a divergence metric such as Kullback-Leibler [14]. One can easily show that denoising autoencoders also tries to sparsify the codes [15]. Other types of regularizers have also been proposed for autoencoders which result in enhanced performance [2], [16].

In this paper, we address sparse autoencoder based on penalizing ℓ_1 norm of the code vector and propose a solution to this non-smooth problem. The proposed method makes some changes to the problem formulation and uses a block coordinate descent strategy as a solution. This strategy breaks the problem variables into two blocks. The first variables block, which appears in non-smooth terms, is updated efficiently via proximal mapping. The second variables block, coming with the smooth term of cost function, is updated using gradient descent method. In the proposed method, detailed in section II, first a solution similar to the solution to the problem of simple autoencoder is attained, in which no sparsity is imposed on the code vector. Then, sparsity is gradually imposed on the code vectors and a sparse autoencoder is formed. This procedure demonstrates some interesting features that are discussed in section II and supported by experiments presented in section III. We then conclude the paper in section IV.

II. Proposed Method

Sparse autoencoder (SAE) is a mathematically constrained autoencoder. Ideally sparse autoencoder problem is formulated as follows:

$$\begin{aligned}\underset{\theta_1,\theta_2}{\arg\min}\frac{1}{N}\sum_{k=1}^{N}\|\mathbf{x^{(k)}}-\hat{\mathbf{x}}^{(\mathbf{k})}\|_2^2+\frac{\beta}{N}\sum_{k=1}^{N}\|\mathbf{z}^{(k)}\|_0\\+\lambda(\|\mathbf{W}^{(1)}\|_F^2+\|\mathbf{W}^{(2)}\|_F^2)\end{aligned}\tag{1}$$

where $\|\cdot\|_0$ stands for pseudo ℓ_0 norm, β and λ are hyper-parameters which control the code sparsity and the weight decay, respectively, N denotes the number of training patterns, $\mathbf{W}^{(i)}$ is weight matrix of i-th layer and $\mathbf{x}^{(k)}$, $\hat{\mathbf{x}}^{(k)}$ and $\mathbf{z}^{(k)}$ represent the k-th training pattern, its reconstructed version and code, receptively. When dimension of the code is sufficiently small, the problem could be handled but, as dimension of the code increases, the problem becomes intractable and relaxed and approximate problems should be considered [17], [8]. The best choice which shows impressive results in the sparse coding field is ℓ_1 norm [18], [19]. Using this approximation, we get into the following problem:

$$\begin{aligned}\underset{\theta_1,\theta_2}{\arg\min}\frac{1}{N}\sum_{k=1}^{N}\|\mathbf{x^{(k)}}-\hat{\mathbf{x}}^{(\mathbf{k})}\|_2^2+\frac{\beta}{N}\sum_{k=1}^{N}\|\mathbf{z}^{(k)}\|_1\\+\lambda(\|\mathbf{W}^{(1)}\|_F^2+\|\mathbf{W}^{(2)}\|_F^2)\end{aligned}\tag{2}$$

Problem (2), as a suitable approximation to problem (1), contains a non-smooth ℓ_1 norm term and, consequently, is unsolvable by the gradient method. Treating this non-smoothness is the main issue addressed in this paper. We propose a novel and efficient algorithm to directly solve problem (2) through a series of sub-problems. First, we rewrite the problem in matrix form:

$$\underset{\theta_1,\theta_2}{\arg\min}\frac{1}{N}\|\mathbf{X}-\hat{\mathbf{X}}\|_F^2+\frac{\beta}{N}\|\mathbf{Z}\|_1+\lambda(\|\mathbf{W}^{(1)}\|_F^2+\|\mathbf{W}^{(2)}\|_F^2)\tag{3}$$

where the matrices are defined as follows:

$$\mathbf{X}=[\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(N)}]\,,\hat{\mathbf{X}}=[\hat{\mathbf{x}}^{(1)},\ldots,\hat{\mathbf{x}}^{(N)}],\ \mathbf{Z}=[\mathbf{z}^{(1)},\ldots,\mathbf{z}^{(N)}]$$

and ℓ_1 matrix norm is $\|\mathbf{A}\|_1=\sum_{i=1}^{r}\sum_{j=1}^{c}|A_{ij}|$, where r and c stand for the row and column numbers of the matrix, respectively.

In order to deal with non-smooth terms in unconstrained problem (3), we change the problem to equivalent constrained optimization as follows:

$$\begin{aligned}\underset{\theta_1,\theta_2,\mathbf{Y}}{\arg\min}\ \ &\frac{1}{N}\|\mathbf{X}-\hat{\mathbf{X}}\|_F^2+\frac{\beta}{N}\|\mathbf{Y}\|_1\\&+\lambda(\|\mathbf{W}^{(1)}\|_F^2+\|\mathbf{W}^{(2)}\|_F^2)\ \ \text{subject to}\ \ \mathbf{Y}=\mathbf{Z}\end{aligned}\tag{4}$$

where we introduce a new matrix variable $\mathbf{Y}$ which links the cost function to the constraint. This problem is solved via penalty methods as follows [20]:

$$\begin{aligned}\mathrm{P}_{\text{L1-SAE}}:\underset{\theta_1,\theta_2,\mathbf{Y}}{\arg\min}\ \ &\frac{1}{N}\|\mathbf{X}-\hat{\mathbf{X}}\|_F^2+\frac{\beta}{N}\|\mathbf{Y}\|_1\\&+\lambda(\|\mathbf{W}^{(1)}\|_F^2+\|\mathbf{W}^{(2)}\|_F^2)+\frac{1}{2\alpha}\|\mathbf{Y}-\mathbf{Z}\|_F^2\end{aligned}\tag{5}$$

ISBN 978-90-827970-1-5 © EURASIP 2018

where $\alpha > 0$ is the penalty parameter. When $\alpha \to 0$, the solution of L1-SAE coincides with those of problem (4). Let we consider the solution to problem (5) as θ_1^α and θ_2^α and the solution to problem (2) as $\theta_1^\star$ and $\theta_2^\star$. Then we have $\lim_{\alpha\to 0}\theta_1^\alpha = \theta_1^\star$ and $\lim_{\alpha\to 0}\theta_2^\alpha = \theta_2^\star$.

Based on the above-mentioned equations, we first find solutions to the L1-SAE problem and then decrease α toward zero. To do this, we try to solve a series of sub-problems (5) for decreasing values of α. Our proposed method to solve these sub-problems is based on block coordinate descent strategy. We divide variables into two sets, and then solve each sub-problem in two steps. First, we keep fixed the variables belonging to set 2 and run the minimization procedure based on set 1 variables. In the next step, we just do the vice versa.

In the problem of L1-SAE, we choose $\mathbf{Y}$ as first block variable and θ_1 and θ_2 as the second block variable. We initialize variables in set 2 and perform the following two steps in each iteration:

Step 1: In this step we have the following optimization problem:

$$\mathbf{Y}(p+1) = \arg\min_{\mathbf{Y}} \ \frac{\beta}{N}\|\mathbf{Y}\|_1 + \frac{1}{2\alpha}\|\mathbf{Y} - \mathbf{Z}(p)\|_F^2 \qquad (6)$$

where p is the iteration index of the optimization problem. Problem (6) contains the non-smooth term of cost function, which is not suitable for optimization by the gradient methods. But, other optimization tools, like proximal mapping, can efficiently solve the problem and is employed in our proposed method. Before we continue, let us introduce the notion of proximal mapping [21].

Definition 1 (Proximal mapping [21])**.** *The proximal mapping of a proper and lower semi-continuous function* $g : \mathbb{R}^m \longrightarrow (-\inf, +\inf]$ *at* $\mathbf{x} \in \mathbb{R}^m$ *is defined as:*

$$\text{prox}_g(\mathbf{x}) \triangleq \arg\min_{\mathbf{u}\in\mathbb{R}^m}\{g(\mathbf{u}) + \frac{1}{2}\|\mathbf{x}-\mathbf{u}\|_2^2\} \qquad (7)$$

The above definition could easily be extended to use for the matrix functions. By comparing formulation (7) to problem (6), we see a clear resemblance and the solution to sub-problem (6) could simply be found as:

$$\mathbf{Y}(p+1) = \text{prox}_g(\mathbf{Z}(p)) \qquad (8)$$

where $g(\mathbf{A}) = \frac{\alpha\beta}{N}\|\mathbf{A}\|_1$. The proximal mapping of $g(\mathbf{X})$ has been formulated in the literature as [21]:

$$\text{prox}_g(\mathbf{A}) = \text{Soft-Threshold}_{\frac{\alpha\beta}{N}}(\mathbf{A}) \qquad (9)$$

where $\text{Soft-Threshold}_{\frac{\alpha\beta}{N}}(\cdot)$ is an element-wise function which performs the following operation on each element of the matrix:

$$\text{Soft-Threshold}_{\frac{\alpha\beta}{N}}(a_{ij}) = \begin{cases} a_{ij} - \frac{\alpha\beta}{N} & \text{if } a_{ij} > \frac{\alpha\beta}{N} \\ 0 & \text{if } |a_{ij}| < \frac{\alpha\beta}{N} \\ a_{ij} + \frac{\alpha\beta}{N} & \text{if } a_{ij} < -\frac{\alpha\beta}{N} \end{cases}$$

So the problem (6) is efficiently solve using the proximal mapping 8.

Step 2: In this step, the following optimization problem is considered:

$$\theta_1(p+1), \theta_2(p+1) = \arg\min_{\theta_1,\theta_2} \ \frac{1}{N}\|\mathbf{X} - \hat{\mathbf{X}}\|_F^2 + \frac{1}{2\alpha}\|\mathbf{Y}(p+1) - \mathbf{Z}\|_F^2 + \lambda(\|\mathbf{W}^{(1)}\|_F^2 + \|\mathbf{W}^{(2)}\|_F^2) \qquad (10)$$

As it can be seen, no non-smooth term is present in problem (10) and it can be solved based on the gradient methods [10] and natural gradient methods [22]. Error back-propagation could also be utilized to evaluate the gradient efficiently. In our proposed method, we use one iteration of the gradient descent for this step. Based on the proposed method, we can find an approximate solution to sub-problem (5) and use it to initialize a similar sub-problem (5) with decreased α. Following this procedure, we can find the solution to the problem (3) for small values of α. For simplicity, we choose a fixed decreasing factor (f) and scale α using this factor for subsequent sub-problems. The pseudo code of the proposed method is shown in algorithm 1.

Algorithm 1 L1-SAE

procedure TRAINING SPARSE AUTOENCODER
Input: Training pattern matrix ($\mathbf{X}$), Initial value for α (α_0), decreasing factor of α (f), Maximum number of reduction in α (T), sparsity hyper-parameter (β), weight decay hyper-parameter (λ), gradient descent step size (μ), Encoder and decoder functions
Initialization: $p = 0$, $\theta_1(p)$, $\theta_2(p)$, t = 1
Output: Autoencoder parameters
  while $t \le T$ **do**
    $\alpha = f^{t-1} \times \alpha_0$
    while termination is met **do**
      $\mu = \mu(p)$
      Compute $\mathbf{Y}(p+1)$ through (8)
      Compute $\theta_1(p+1)$ and $\theta_2(p+1)$ through (10)
      $p = p + 1$
    $t = t + 1$

Complexity of the proposed method for solving each sub-problem is affected by element-wise soft-thresholding operation in each step, which is highly parallelizable and could be implemented efficiently. Besides, rather than solving a series of sub-problems in full, we just need to use an approximate solution to the problem by reducing the number of iterative computations. Moreover, as our simulations show, the problem is insensitive to small values of reduction factors and $f \sim 0.6$ can also lead to acceptable results. This will limit the number of sub-problems needed to reach an approximate solution to problem (2).

III. EXPERIMENTAL RESULTS

In this section, we test performance of the proposed sparse autoencoder (L1SAE) over three benchmark image datasets as follows:

- MNIST handwritten digits [23]: This dataset contains 50000, 10000, and 10000 training, validation, and test

ISBN 978-90-827970-1-5 © EURASIP 2018

TABLE I: Selected hyper-parameters of different autoencoders

Parameters	β	ρ	λ_1	λ_2	σ
AE	-	-	0.00005	0.0001	-
KLDSAE	1	0.01	0.00005	0.0001	-
AL1SAE	3	-	0.00005	0.0001	-
DenAE	-	-	0.00005	0.0001	0.5
NCAE	3	0.05	0.0001	0.0001	-
L1SAE	0.5	-	0.00005	0.0001	-

TABLE II: Mean and standard deviation of network accuracy pretrained using different methods on MNIST, NORB and SVHN datasets.

	MNIST		NORB		SVHN	
Pretraining	Mean (%)	STD	Mean	STD	Mean	STD
AL1SAE	97.77	0.07	85.19	0.46	80.82	0.19
KLDSAE	97.93	0.06	86.46	0.33	81.15	0.11
DenAE	97.88	0.09	86.56	0.18	81.11	0.13
NCAE	97.98	0.04	86.68	0.19	81.22	0.12
L1SAE	98.05	0.06	86.85	0.24	81.36	0.15

samples of hand written digits, respectively. All the samples are 28×28 gray scale images, each represented by a vector of size 784.

- NORB object recognition dataset [24]: This dataset contains 24300 training and 24300 test samples. The samples are 2 channel 96×96 images of 50 toys from five generic categories: 1) four-legged animals, 2) human figure, 3) airplanes, 4) trucks, and 5) cars. Similar to the procedure used in [16], we take the inner 64×64, inner pixels of samples, and resize them to 32×32 pixels. Thus, each of the samples will contain $2 \times 32 \times 32 = 2048$ elements.
- SVHN dataset of street view house numbers [25]: This dataset contains 73257 training and 2603 test samples. All the samples are 32×32 gray scale images, each represented by a vector of size 1024.

The proposed method is compared with five other methods, including: 1) Simple autoencoder (AE) introduced in [4], 2) Sparse autoencoder based on smooth approximation of ℓ_1 norm by Student-t penalty (AL1SAE) introduced in [17], 3) Sparse autoencoder based on minimizing Kullback-Leibler divergence of average code elements from a fixed small value (KLDSAE) introduced in [14], 4) Denoising autoencoder introduced in [15], 5) Method introduced in [16] where non-negativity is added to the weight matrix of KLDSAE method (NCAE).

For unsupervised feature learning, we use our proposed method to train a sparse autoencoder, and then compare the learned features to those of other methods in terms of reconstruction error and degree of sparsity via Gini index [26]. All the methods were implemented in Matlab using stochastic gradient descent on 1000 mini-batches in each epoch (except NCAE that we use its code from https://github.com/ehosseiniasl/ Nonnegativity-Constrained-Autoencoder-NCAE.git). We use logistic functions for both encoder and decoder for all datasets. The sizes of codes for MNIST, NORB and SVHN datasets are 196, 256, and 256, respectively. Table I shows the best hyper-parameters selected through searching over a grid for all methods.

Figures 1(a) and 1(b) show the result of reconstruction error and sparsity over the test samples with corresponding standard deviation for different methods, respectively. As expected, simple autoencoder, which has no sparsity constraint (and the lowest sparsity), comes with the lowest reconstruction error (This method is added just as a reference for the lowest possible reconstruction error). Among other methods, the L1SAE comes with the lowest reconstruction error and highest sparsity simultaneously for MNIST and SVHN datasets. For NORB dataset, while the reconstruction error of the proposed method is not the best, it presents a significant improvement to sparsity over the existing autoencoders. This superiority originates from incremental complexity feature of the proposed method. In the starting epochs, a dense code is trained using the proposed method and receptive fields are not localized. Each time α decreases, the algorithm tries to prone the receptive fields of the previous sub-problem, so as to increase the codes sparsity. In other words, the algorithm gradually localizes the features and increases sparsity of the codes. This procedure is shown in Figure 1(c) for several receptive fields of autoencoder trained on MNIST dataset. Accordingly, in this procedure, complexity is gradually fed to the problem and we expect to enhance the efficiency (It is to be noted that NCAE results are not shown in figure 1, because its formulation to penalize weights is different from that in other methods, so may not lead to a fair comparison with different algorithms. However, we demonstrate its results obtained from the subsequent experiment).

In the next experiment, we use different methods to pretrain a deep supervised structure, and then fine tune the network parameters using the stochastic gradient descent. The network structure used for different datasets is as follows:

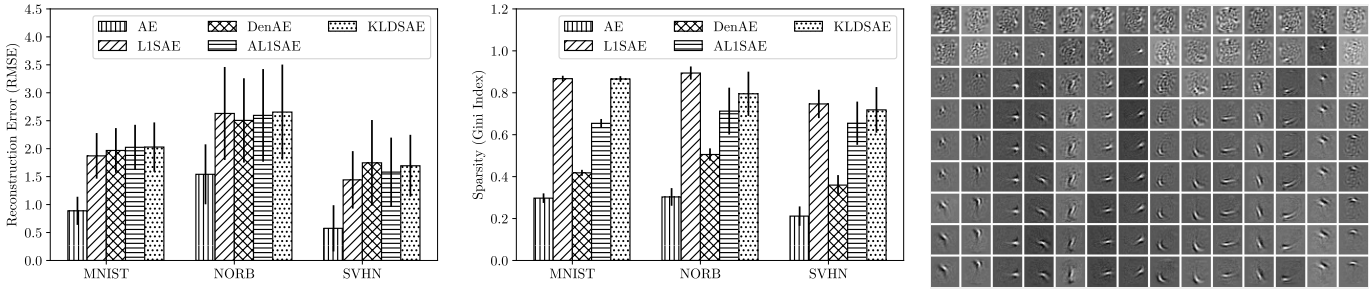
$$\text{MNIST dataset} : \text{FCL}(784) \Rightarrow \text{FCL}(196) \Rightarrow \text{FCL}(49) \Rightarrow \text{FCL}(10)$$
$$\text{NORB dataset} : \text{FCL}(2048) \Rightarrow \text{FCL}(256) \Rightarrow \text{FCL}(32) \Rightarrow \text{FCL}(5)$$
$$\text{SVHN dataset} : \text{FCL}(1024) \Rightarrow \text{FCL}(256) \Rightarrow \text{FCL}(64) \Rightarrow \text{FCL}(10)$$

As seen in table II, the proposed method presents higher accuracy in all datasets, because of the more efficient method employed for pretraining of network (the networks differ only in pretraining method).

IV. Conclusion

We have proposed a new method to train a sparse autoencoder based on ℓ_1 regularizer through a series of sub-problems obtained using penalty method. In each sub-problems, to handle non-smooth ℓ_1 term in the learning cost function, we adopt a block coordinate descent strategy which decomposes the problem variables into two sets, such that one holds all non-smooth terms and the other just deals with the smooth terms. The first variable set is updated using the proximal mapping, while the stochastic gradient descent is utilized to update the second variable set. Our simulations on three image datasets show that the proposed method presents a smaller

ISBN 978-90-827970-1-5 © EURASIP 2018



(a) Reconstruction error (b) Sparsity (c) Receptive Fields Evolution (MNIST dataset)

Fig. 1: Average and standard deviation of test samples reconstruction error and Gini index for different methods and datasets with some selected receptive fields evolution over MNIST dataset

reconstruction error and higher sparsity, as compared to those of several well-known autoencoders. The proposed method has also been evaluated in pretraining a deep structure which, after fine-tuning, outperforms the conventional methods.

REFERENCES

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[3] David H Hubel and Torsten N Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *The Journal of physiology*, vol. 148, no. 3, pp. 574–591, 1959.

[4] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle, "Greedy layer-wise training of deep networks," in *Advances in neural information processing systems*, 2007, pp. 153–160.

[5] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[6] Hervé Bourlard and Yves Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biological cybernetics*, vol. 59, no. 4, pp. 291–294, 1988.

[7] Richard S. Zemel, "Autoencoders, minimum description length and helmholtz free energy," in *Proceedings of the Neural Information Processing Systems*, 1994.

[8] Marc'Aurelio Ranzato, Christopher Poultney, Sumit Chopra, and Yann LeCun, "Efficient learning of sparse representations with an energy-based model," in *Proceedings of the 19th International Conference on Neural Information Processing Systems*. MIT Press, 2006, pp. 1137–1144.

[9] Y-lan Boureau, Yann L Cun, et al., "Sparse feature learning for deep belief networks," in *Advances in neural information processing systems*, 2008, pp. 1185–1192.

[10] Jiquan Ngiam, Adam Coates, Ahbik Lahiri, Bobby Prochnow, Quoc V Le, and Andrew Y Ng, "On optimization methods for deep learning," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 265–272.

[11] Will Y Zou, Andrew Y Ng, and Kai Yu, "Unsupervised learning of visual invariance with temporal coherence," in *NIPS 2011 workshop on deep learning and unsupervised feature learning*, 2011, vol. 3.

[12] Michael Elad, *Sparse and redundant representation*, Springer, 2010.

[13] Bruno A Olshausen and David J Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607, 1996.

[14] Andrew Ng, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, no. 2011, pp. 1–19, 2011.

[15] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.

[16] Ehsan Hosseini-Asl, Jacek M Zurada, and Olfa Nasraoui, "Deep learning of part-based representation of data using sparse autoencoders with nonnegativity constraints," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 12, pp. 2486–2498, 2016.

[17] Nan Jiang, Wenge Rong, Baolin Peng, Yifan Nie, and Zhang Xiong, "An empirical analysis of different sparse penalties for autoencoder in unsupervised feature learning," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2015, pp. 1–8.

[18] Scott Shaobing Chen, David L Donoho, and Michael A Saunders, "Atomic decomposition by basis pursuit," *SIAM review*, vol. 43, no. 1, pp. 129–159, 2001.

[19] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y Ng, "Efficient sparse coding algorithms," in *Advances in neural information processing systems*, 2007, pp. 801–808.

[20] Stephen Wright and Jorge Nocedal, "Numerical optimization," *Springer Science*, vol. 35, pp. 67–68, 1999.

[21] Neal Parikh, Stephen Boyd, et al., "Proximal algorithms," *Foundations and Trends® in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[22] Guillaume Desjardins, Karen Simonyan, Razvan Pascanu, et al., "Natural neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2071–2079.

[23] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[24] Yann LeCun, Fu Jie Huang, and Leon Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2004, vol. 2, pp. II–104.

[25] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS workshop on deep learning and unsupervised feature learning*, 2011.

[26] Mostafa Sadeghi and Massoud Babaie-Zadeh, "Iterative sparsification-projection: fast and robust sparse signal approximation," *IEEE Transactions on Signal Processing*, vol. 64, no. 21, pp. 5536–5548, 2016.

ISBN 978-90-827970-1-5 © EURASIP 2018