Deep Learning of Constrained Autoencoders for Enhanced Understanding of Data

Babajide O. Ayinde, *Student Member, IEEE*, and Jacek M. Zurada, *Life Fellow, IEEE*

***Abstract*— Unsupervised feature extractors are known to perform an efficient and discriminative representation of data. Insight into the mappings they perform and human ability to understand them, however, remain very limited. This is especially prominent when multilayer deep learning architectures are used. This paper demonstrates how to remove these bottlenecks within the architecture of non-negativity constrained autoencoder. It is shown that using both L1 and L2 regularizations that induce non-negativity of weights, most of the weights in the network become constrained to be non-negative, thereby resulting into a more understandable structure with minute deterioration in classification accuracy. Also, this proposed approach extracts features that are more sparse and produces additional output layer sparsification. The method is analyzed for accuracy and feature interpretation on the MNIST data, the NORB normalized uniform object data, and the Reuters text categorization data set.**

***Index Terms*— Deep learning (DL), part-based representation, receptive field, sparse autoencoder (SAE), white-box model.**

I. Introduction

DEEP learning (DL) networks take the form of heuristic and rich architectures that develop unique intermediate data representation. The complexity of architectures is reflected by both the sizes of layers and, for a large number of data sets reported in the literature, also by the processing. In fact, the architectural complexity and the excessive number of weights and units are often built into the DL data representation by design and are deliberate [1]–[5]. Although deep architectures are capable of learning highly complex mappings, they are difficult to train, and it is usually hard to interpret what each layer has learned. Moreover, gradient-based optimization with random initialization used in training is susceptible to converging to local minima [6], [7].

In addition, it is generally believed that humans analyze complex interactions by breaking them into isolated and understandable hierarchical concepts. The emergence of part-based representation in human cognition can be conceptually tied to the non-negativity constraints [8]. One way to enable easier human understandability of concepts in neural networks is to constrain the network's weights to be non-negative. Note that such representation through non-negative weights of a multilayer network perceptron can implement any shattering of points provided suitable negative bias values are used [9].

Drawing inspiration from the idea of non-negative matrix factorization (NMF) and sparse coding [8], [10], the hidden structure of data can be unfolded by learning features that have capabilities to model the data in parts. Although NMF enforces the encoding of both the data and features to be non-negative thereby resulting in additive data representation, however, incorporating sparse coding within NMF for the purpose of encoding data is computationally expensive, while with AEs, this incorporation is learning-based and fast. In addition, the performance of a deep network can be enhanced using non-negativity constrained sparse autoencoder (NCSAE) with part-based data representation capability [11], [12].

It is remarked that weight regularization is a concept that has been employed both in the understandability and generalization context. It is used to suppress the magnitudes of the weights by reducing the sum of their squares. Enhancement in sparsity can also be achieved by penalizing the sum of absolute values of the weights rather than the sum of their squares [13]–[17]. In this paper, the work proposed in [11] is extended by modifying the cost function to extract more sparse features, encouraging the non-negativity of the network weights, and enhancing the understandability of the data. Other related model is the non-negative sparse autoencoder (NNSAE) trained with an online algorithm with tied weights and linear output activation function to mitigate the training hassle [18]. While Lemme *et al.* [18] uses a piecewise linear decay function to enforce non-negativity and focuses on shallow architecture, the proposed uses a composite norm with focus on deep architectures. Dropout is another recently introduced and widely used heuristic to sparsify AEs and prevent overfitting by randomly dropping units and their connections from the neural network during training [19], [20].

More recently, different paradigms of AEs that constrain the output of encoder to follow a chosen prior distribution have been proposed [21]–[23]. In variational autoencoding, the decoder is trained to reconstruct the input from samples that follow chosen prior using variational inference [21]. Realistic data points can be reconstructed in the original data space by feeding the decoder with samples from chosen prior distribution. On the other hand, adversarial AE matches the encoder's output distribution to an arbitrary prior distribution using adversarial training with discriminator and the

Manuscript received June 1, 2016; revised December 19, 2016 and June 12, 2017; accepted August 14, 2017. This work was supported by NSF under Grant 1641042. *(Corresponding author: Jacek M. Zurada.)*

B. O. Ayinde is with the Department of Electrical and Computer Engineering, University of Louisville, Louisville, KY 40292 USA.

J. M. Zurada is with the Department of Electrical and Computer Engineering, University of Louisville, Louisville, KY 40292 USA, and also with the Information Technology Institute, University of Social Science, 90-113 Łódz, Poland (e-mail: jacek.zurada@louisville.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2017.2747861

generator [22]. Upon adversarial training, encoder learns to map data distribution to the prior distribution.

The problem addressed here is threefold.

1) The interpretability of AE-based deep layer architecture fostered by enforcing high degree of weight's non-negativity in the network. This improves on NCSAEs that show negative weights despite imposing non-negativity constraints on the network's weights [11].
2) It is demonstrated how the proposed architecture can be utilized to extract meaningful representations that unearth the hidden structure of a high-dimensional data.
3) It is shown that the resulting non-negative AEs do not deteriorate their classification performance.

This paper considerably expands the scope of the AE model first introduced in [24] by: 1) introducing smoothing function for L_1 regularization for numerical stability; 2) illustrating the connection between the proposed regularization and weights' non-negativity; 3) drawing more insight into a variety of data set; 4) comparing the proposed with recent AE architectures; and 5) supporting the interpretability claim with new experiments on text categorization data. This paper is structured as follows. Section II introduces the network configuration and the notation for non-negative sparse feature extraction. Section III discusses the experimental designs and Section IV presents the results. Finally, conclusions are drawn in Section V.

II. Non-Negative Sparse Feature Extraction Using Constrained Autoencoders

As shown in [8], one way of representing data is by shattering it into various distinct pieces in a manner that additive merging of these pieces can reconstruct the original data. Mapping this intuition to AEs, the idea is to sparsely disintegrate data into parts in the encoding layer and subsequently additively process the parts to recombine the original data in the decoding layer. This disintegration can be achieved by imposing non-negativity constraint on the network's weights [11], [25], [26].

A. L_1/L_2-Non-Negativity Constrained Sparse Autoencoder (L_1/L_2-NCSAE)

In order to encourage higher degree of non-negativity in network's weights, a composite penalty term (1) is added to the objective function, resulting in the cost function expression for L_1/L_2-NCSAE

$$\begin{aligned} &J_{L_1/L_2\text{-NCSAE}}(\mathbf{W},\mathbf{b}) \\ &\quad = J_{AE} + \beta\sum_{r=1}^{n'} D_{\mathrm{KL}}\left(p \,\middle\|\, \frac{1}{m}\sum_{k=1}^{m} h_r(\mathbf{x}^{(k)})\right) \\ &\qquad + \sum_{l=1}^{2}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}} f_{L_1/L_2}\big(w_{ij}^{(l)}\big) \end{aligned} \tag{1}$$

where $\mathbf{W} = \{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}\}$ and $\mathbf{b} = \{\mathbf{b}_x, \mathbf{b}_h\}$ represent the weights and biases of encoding and decoding layers, respectively; s_l is the number of neurons in layer l; and $w_{ij}^{(l)}$ represents the connection between jth neuron in layer $l-1$ and ith neuron in layer l and for given input $\mathbf{x}$

$$J_{\mathrm{AE}} = \frac{1}{m}\sum_{k=1}^{m}\left\|\sigma(\mathbf{W}^{(2)}\sigma(\mathbf{W}^{(1)}\mathbf{x}^{(k)} + \mathbf{b}_x) + \mathbf{b}_h) - \mathbf{x}^{(k)}\right\|_2^2 \tag{2}$$

where m is the number of training examples, $||\,.\,||_2$ is the Euclidean norm, $D_{\mathrm{KL}}(.)$ is the Kullback–Leibler (KL) divergence for sparsity control [27] with p denoting the desired activation and the average activations of hidden units, n' is the number of hidden units, $h_j(\mathbf{x}^{(k)}) = \sigma(\mathbf{W}_j^{(1)}\mathbf{x}^{(k)} + b_{x,j})$ denotes the activation of hidden unit j due to input $\mathbf{x}^{(k)}$, and $\sigma(.)$ is the element-wise application of the logistic sigmoid, $\sigma(\mathbf{x}) = 1/(1+exp(-\mathbf{x}))$, β controls the sparsity penalty term, and

$$f_{L_1/L_2}(w_{ij}) = \begin{cases} \alpha_1\Gamma(w_{ij},\kappa) + \dfrac{\alpha_2}{2}||w_{ij}||^2 & w_{ij} < 0 \\ 0 & w_{ij} \geq 0 \end{cases} \tag{3}$$

where α_1 and α_2 are L_1 and L_2 non-negativity-constraint weight penalty factors, respectively. p, β, α_1, and α_2 are experimentally set to 0.05, 3, 0.0003, and 0.003, respectively, using 9000 randomly sampled images from the training set as a held-out validation set for hyperparameter tuning, and the network is retrained on the entire data set. The weights are updated as below using the error backpropagation

$$w_{ij}^{(l)} = w_{ij}^{(l)} - \xi\frac{\partial}{\partial w_{ij}^{(l)}} J_{L_1/L_2\text{-NCSAE}}(\mathbf{W},\mathbf{b}) \tag{4}$$

$$b_i^{(l)} = b_i^{(l)} - \xi\frac{\partial}{\partial b_i^{(l)}} J_{L_1/L_2\text{-NCSAE}}(\mathbf{W},\mathbf{b}) \tag{5}$$

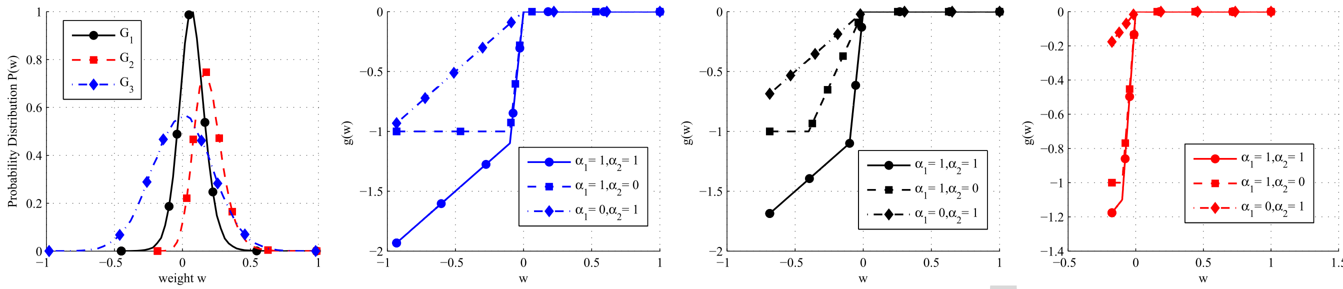
where $\xi > 0$ is the learning rate and the gradient of L_1/L_2-NCSAE loss function is computed as

$$\begin{aligned} &\frac{\partial}{\partial w_{ij}^{(l)}} J_{L_1/L_2\text{-NCSAE}}(\mathbf{W},\mathbf{b}) \\ &\quad = \frac{\partial}{\partial w_{ij}^{(l)}} J_{\mathrm{AE}}(\mathbf{W},\mathbf{b}) \\ &\qquad + \beta\frac{\partial}{\partial w_{ij}^{(l)}} D_{KL}\left(p \,\middle\|\, \frac{1}{m}\sum_{k=1}^{m} h_j(\mathbf{x}^{(k)})\right) + g\big(w_{ij}^{(l)}\big) \end{aligned} \tag{6}$$

where $g(w_{ij})$ is a composite function denoting the derivative of $f_{L_1/L_2}(w_{ij})$ (3) with respect to w_{ij} as

$$g(w_{ij}) = \begin{cases} \alpha_1\nabla_{\mathbf{w}}\|w_{ij}\| + \alpha_2 w_{ij} & w_{ij} < 0 \\ 0 & w_{ij} \geq 0. \end{cases} \tag{7}$$

Although the penalty function in (1) is an extension of NCSAE (obtained by setting α_1 to zero), a close scrutiny of the weight distribution of both the encoding and decoding layer in NCSAE reveals that many weights are still not non-negative despite imposing non-negativity constraints. The reason for this is that the original L_2 norm used in NCSAE penalizes the negative weights with bigger magnitudes stronger than those with smaller magnitudes. This forces a good number of the weights to take on small negative values. This paper uses additional L_1 to even out this occurrence, that is, the



(a) (b) (c) (d)

Fig. 1. (a) Symmetric (G_3) and skewed (G_1 and G_2) weight distributions. Decay function with three values of α_1 and α_2 for weight distribution. (b) G_3. (c) G_1. (d) G_2.

L_1 penalty forces most of the negative weights to become non-negative.

B. Implication of Imposing Non-Negative Parameters With Composite Decay Function

The graphical illustration of the relation between the weight distribution and the composite decay function is shown in Fig. 1. Ideally, addition of Frobenius norm of the weight matrix ($\alpha||\mathbf{W}||_F^2$) to the reconstruction error in (2) imposes a Gaussian prior on the weight distribution as shown in curve G_3 in Fig. 1(a). However, using the composite function in (3) results in imposition of positively skewed deformed Gaussian distribution as in curves G_1 and G_2. The degree of non-negativity can be adjusted using parameters α_1 and α_2. Both parameters have to be carefully chosen to enforce non-negativity while simultaneously ensuring good supervised learning outcomes. The effect of L_1 ($\alpha_2 = 0$), L_2 ($\alpha_1 = 0$), and L_1/L_2 ($\alpha_1 \neq 0$ and $\alpha_2 \neq 0$), non-negativity penalty terms, on weight updates for weight distributions G_1, G_2, and G_3 are, respectively, shown in Fig. 1(b)–(d). It can be observed for all the three distributions that L_1/L_2 regularization enforces stronger weight decay than individual L_1 and L_2 regularizations. Other observation from Fig. 1 is that the more positively skewed the weight distribution becomes, the lesser the weight decay function.

The consequences of minimizing (1) are that: 1) the average reconstruction error is reduced; 2) the sparsity of the hidden layer activations is increased because more negative weights are forced to zero, thereby leading to sparsity enhancement; and 3) the number of non-negative weights is also increased. The resultant effect of penalizing the weights simultaneously with L_1 and L_2 norms is that large positive connections are preserved while their magnitudes are shrunk. However, the L_1 norm in (3) is non-differentiable at the origin, and this can lead to numerical instability during simulations. To circumvent this drawback, one of the well-known smoothing function that approximates L_1 norm as in (3) is utilized. Given any finite dimensional vector $\mathbf{z}$ and positive constant κ, the following smoothing function approximates L_1 norm:

$$\Gamma(\mathbf{z}, \kappa) = \begin{cases} ||\mathbf{z}|| & ||\mathbf{z}|| > \kappa \\ \dfrac{||\mathbf{z}||^2}{2\kappa} + \dfrac{\kappa}{2} & ||\mathbf{z}|| \leq \kappa \end{cases} \tag{8}$$

with gradient

$$\nabla_{\mathbf{z}}\Gamma(\mathbf{z}, \kappa) = \begin{cases} \dfrac{\mathbf{z}}{||\mathbf{z}||} & ||\mathbf{z}|| > \kappa \\ \dfrac{\mathbf{z}}{\kappa} & ||\mathbf{z}|| \leq \kappa. \end{cases} \tag{9}$$

For convenience, we adopt (8) to smoothen the L_1 penalty function and κ is experimentally set to 0.1.

III. Experiments

In the experiments, three data sets are used, namely, MNIST [28], NORB normalized-uniform [29], and Reuters-21578 text categorization data set. The Reuters-21578 text categorization data set comprises documents that featured in 1987 Reuters newswire. The ModApte split was employed to limit the data set to 10 most frequent classes. The ModApte split was utilized to limit the categories to 10 most frequent categories. The bag-of-words format that has been stemmed and stop-word removed was used (see http://people.kyb.tuebingen.mpg.de/pgehler/rap/ for further clarification). The data set contains 11 413 documents with 12 317 dimensions. Two techniques were used to reduce the dimensionality of each document in order to preserve the most informative and less correlated words [30]. To reduce the dimensionality of each document to contain the most informative and less correlated words, words were first sorted based on their frequency of occurrence in the data set. Words with frequency below 4 and above 70 were then eliminated. The most informative words that do not occur in every topic were selected based on information gain with the class attribute. The remaining words (or features) in the data set were sorted using this method, and the less important features were removed based on the desired dimension of documents. In this paper, the length of the feature vector for each of the documents was reduced to 200.

In the preliminary experiment, the subsets 1, 2, and 6 from the MNIST handwritten digits are extracted for the purpose of understanding how the deep network constructed using L_1/L_2-NCSAE processes and classify their input. For easy interpretation, a small deep network was constructed and trained by stacking two AEs with 10 hidden neurons each and 3 softmax neurons. The number of hidden neurons was chosen to obtain reasonably good classification accuracy

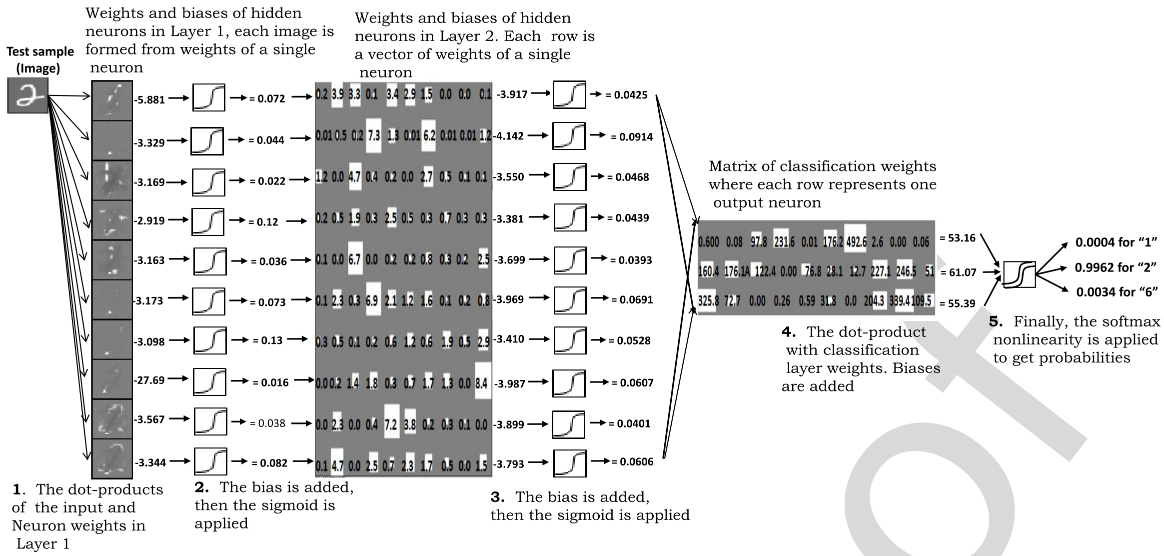


Fig. 2. Filtering the signal through the L_1/L_2-NCSAE trained using the reduced MNIST data set with class labels 1, 2, and 6. The test image is a 28×28 pixels image unrolled into a vector of 784 values. Both the input test sample and the receptive fields of the first autoencoding layer are presented as images. The weights of the output layer are plotted as a diagram with one row for each output neuron and one column for every hidden neuron in $(L-1)$th layer. The architecture is 784-10-10-3. The range of weights is scaled to $[-1, 1]$ and mapped to the graycolor map. $w = -1$ is assigned to black, $w = 0$ to gray, and $w = 1$ is assigned to white color. That is, black pixels indicate negative, gray pixels indicate zero-valued weights, and white pixels indicate positive weights.

while keeping the network reasonably small. The network is intentionally kept small because the full MNIST data would require larger hidden layer size and this may limit network interpretability. An image of digit 2 is then filtered through the network, and it can be observed in Fig. 2 that sparsification of the weights in all the layers is one of the aftermath of non-negativity constraints imposed on the network. Another observation is that most of the weights in the network have been confined to non-negative domain, which removes the opaqueness of the DL process. It can be seen that the fourth and seventh receptive fields of the first AE layer have dominant activations (with activation values 0.12 and 0.13, respectively) and they capture most information about the test input. Also, they are able to filter distinct part of input digit. The outputs of the first layer sigmoid constitute higher level features extracted from test image with emphasis on the fourth and seventh features. Subsequently, in the second layer, the second, sixth, eight, and tenth neurons have dominant activations (with activation values 0.0914, 0.0691, 0.0607, and 0.0606, respectively) because they have stronger connections with the dominant neurons in the first layer than the rest. Last, in the softmax layer, the second neuron was 99.62% activated because it has the strongest connections with the dominant neurons in the second layer, thereby classifying the test image as "2."

The fostering of interpretability is also demonstrated using a subset of NORB normalized-uniform data set [29] with class labels "four-legged animals," "human figures," and "airplanes." The 1024-10-5-3 network configuration was trained on the subset of the NORB data using two stacked L_1/L_2-NCSAEs and a Softmax layer. Fig. 3(b) shows the randomly sampled test patterns, and the weights and activations of the first and second AE layers are shown in Fig. 3(a). The bar charts indicate the activations of hidden units for the sample input patterns. The features learned by units in each layer are localized, sparse, and allow easy interpretation of isolated data parts. The features mostly show non-negative weights making it easier to visualize to what input object patterns they respond. It can be seen that units in the network discriminate among objects in the images and react differently to input patterns. Third, sixth, eight, and ninth hidden units of layer 1 capture features that are common to objects in class "2" and react mainly to them as shown in the first layer activations. Also, the features captured by the second layer activations reveal that the second and fifth hidden units are mainly stimulated by objects in class "2."

The outputs of Softmax layer represent the *a posteriori* class probabilities for a given sample and are denoted as Softmax scores. An important observation from Fig. 3(a)–(c) is that hidden units in both layers did not capture significant representative features for class "1" white color-coded test sample. This is one of the reasons why it is misclassified into class "3" with a probability of 0.57. The argument also goes for class "1" dark-gray color-coded test sample misclassified into class "3" with a probability of 0.60. In contrast, hidden units in both layers capture significant representative features for class "2" test samples of all color codes. This is why all class "2" test samples are classified correctly with high probabilities as shown in Fig. 3(d). Last, the network contains a good number of representative features for class "3" test samples and was able to classify 4 out of 5 correctly as given in Fig. 3(e).

IV. Results and Discussion

A. Unsupervised Feature Learning of Image Data

In the first set of experiments, three-layer L_1/L_2-NCSAE, NCSAE [11], DpAE [19], and conventional SAE network with 196 hidden neurons were trained using MNIST data set

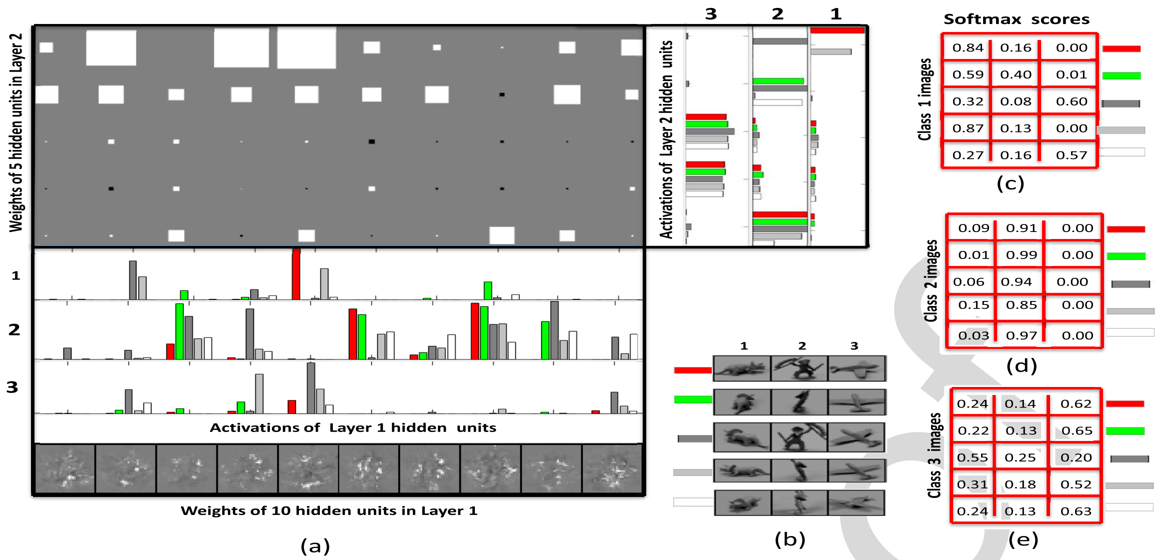


Fig. 3. Weights were trained using two stacked L_1/L_2-NCSAEs. RFs learned from the reduced NORB data set are plotted as images at the bottom part of (a). The intensity of each pixel is proportional to the magnitude of the weight connected to that pixel in the input image with negative value indicating black, positive values white, and the value 0 corresponding to gray. The biases are not shown. The activations of first layer hidden units for the NORB objects presented in (b) are depicted on the bar chart on top of the RFs. The weights of the second layer AE are plotted as a diagram at the topmost part of (a). Each row of the plot corresponds to the weight of each hidden unit of second AE and each column for weight of every hidden unit of the first layer AE. The magnitude of the weight corresponds to the area of each square; white indicates positive, gray indicates zero, and black negative sign. The activations of second layer hidden units are shown as bar chart in the right-hand side of the second layer weight diagram. Each column shows the activations of each hidden unit for five color-coded examples of the same object. The outputs of Softmax layer for color-coded test objects with class labels (c) "fourlegged animals" tagged as class 1, (d) "human figures" as class 2, and (e) "airplanes" as class 3.

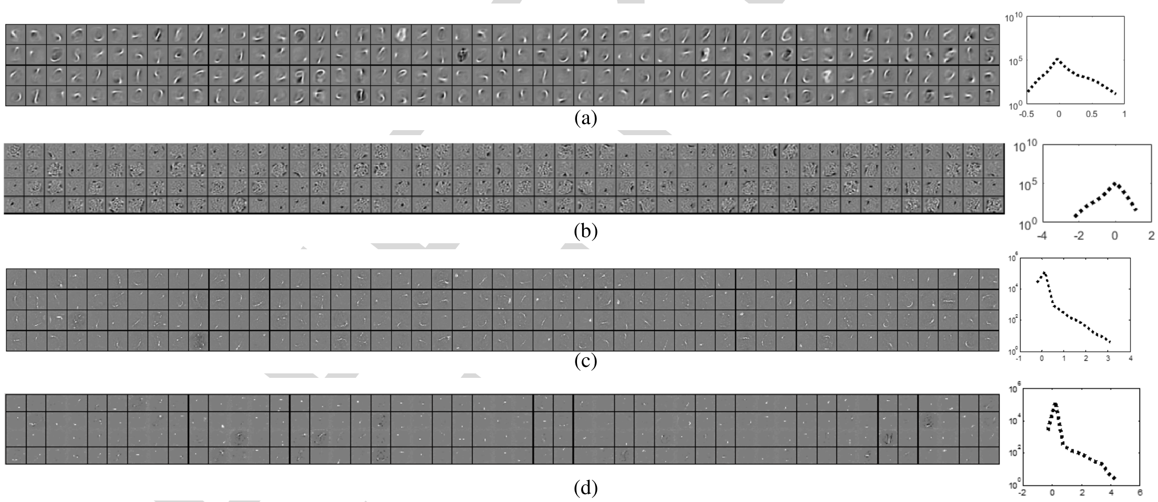
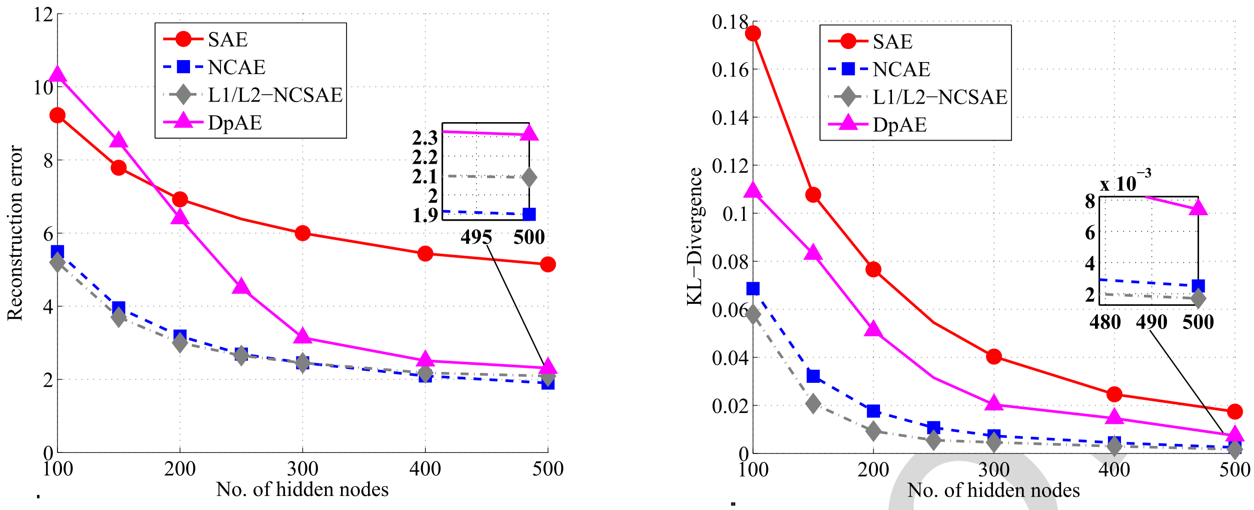


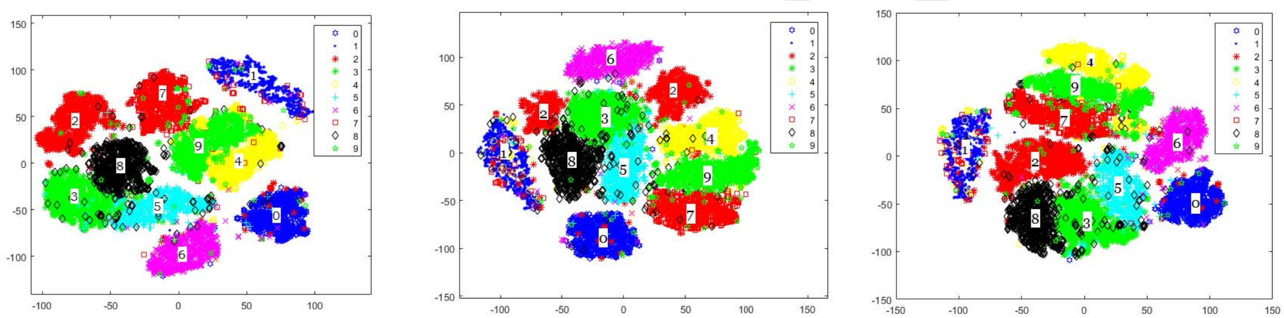
Fig. 4. 196 receptive fields ($\mathbf{W}^{(1)}$) with weight histograms learned from MNIST digit data set using (a) SAE, (b) DpAE, (c) NCSAE, and (d) L_1/L_2-NCSAE. Black pixels indicate negative, and white pixels indicate positive weights. The range of weights are scaled to $[-1,1]$ and mapped to the graycolor map. $w = -1$ is assigned to black, $w = 0$ to gray, and $w = 1$ is assigned to white color.

of handwritten digits and their ability to discover patterns in high dimensional data are compared. These experiments were run one time and recorded. The encoding weights $\mathbf{W}^{(1)}$, also known as receptive fields or filters as in the case of image data, are reshaped, scaled, centered in a 28×28 pixel box, and visualized. The filters learned by L_1/L_2-NCSAE are compared with that learned by its counterparts, NCSAE and SAE. It can be easily observed from the results in Fig. 4 that L_1/L_2-NCSAE learned receptive fields that are more sparse and localized than those of SAE, DpAE, and NCSAE. It is remarked that the black pixels in both SAE and DpAE features are results of the negative weights whose values and numbers are reduced in NCSAE with non-negativity constraints, which are further reduced by imposing an additional L_1 penalty term in L_1/L_2-NCSAE as shown in the histograms located on the right side of the figure. In the case of L_1/L_2-NCSAE, tiny strokes and dots that constitute the basic part of handwritten digits are unearthed compared to SAE, DpAE, and NCSAE. Most of the features learned by SAE are major parts of the digits or the blurred version of the digits, which are obviously not as sparse as those learned by L_1/L_2-NCSAE. Also, the features learned by DpAE are fuzzy compared to those of



(a) (b)

Fig. 5. (a) Reconstruction error and (b) sparsity of hidden units measured by KL-divergence using MNIST train data set with $p = 0.05$.



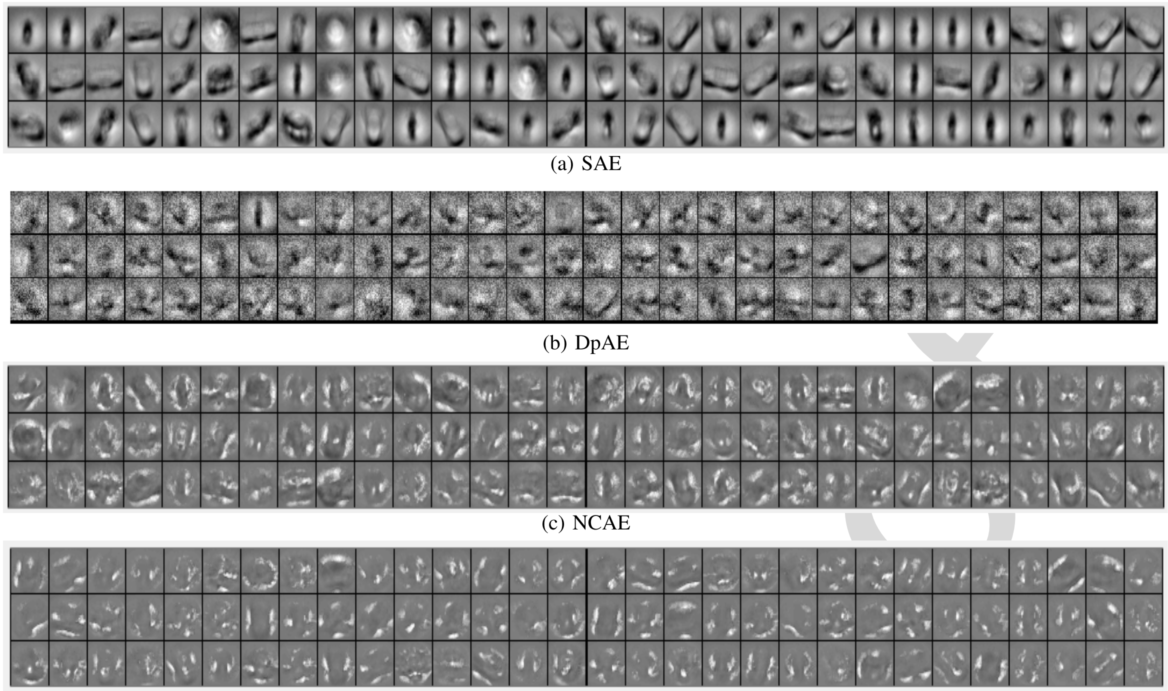
(a) (b) (c)

Fig. 6. t-SNE projection [31] of 196D representations of MNIST handwritten digits using (a) DpAE, (b) NCSAE, and (c) L_1/L_2-NCSAE.

L_1/L_2-NCSAE that are sparse and distinct. Therefore, the achieved sparsity in the encoding can be traced to the ability of L_1 and L_2 regularizations in enforcing high degree of weights' non-negativity in the network.

Likewise in Fig. 5(a), L_1/L_2-NCSAE with other AEs are compared in terms of reconstruction error, while varying the number of hidden nodes. As expected, it can be observed that L_1/L_2-NCSAE yields a reasonably lower reconstruction error on the MNIST training set compared to SAE, DpAE, and NCSAE. Although, a close scrutiny of the result also reveals that the reconstruction error of L_1/L_2-NCSAE deteriorates compared to NCSAE when the hidden size grows beyond 400. However, on the average, L_1/L_2-NCSAE reconstructs better than other AEs considered. It can also be observed that DpAE with 50% dropout has high reconstruction error when the hidden layer size is relatively small (100 or less). This is because the few neurons left are unable to capture the dynamics in the data, which subsequently results in underfitting the data. However, the reconstruction error improves as the hidden layer size is increased. Lower reconstruction error in the case of L_1/L_2-NCSAE and NCSAE is an indication that non-negativity constraint facilitates the learning of parts of digits that are essential for reconstructing the digits. In addition, the KL-divergence sparsity measure reveals that L_1/L_2-NCSAE has more sparse hidden activations than SAE, DpAE, and NCSAE for different hidden layer size, as shown in Fig. 5(b). Again, averaging over all the training examples, L_1/L_2-NCSAE yields less activated hidden neurons compared to its counterparts. Also, using t-distributed stochastic neighbor embedding (t-SNE) to project the 196-D representation of MNIST handwritten digits to 2-D space, the distribution of features encoded by 196 encoding filters of DpAE, NCSAE, and L_1/L_2-NCSAE are, respectively, visualized in Fig. 6(a)–(c). A careful look at Fig. 6(a) reveals that digits "4" and "9" are overlapping in DpAE, and this will inevitably increase the chance of misclassifying these two digits. It can also be observed in Fig. 6(b) corresponding to NCSAE that digit "2" is projected with two different landmarks. In sum, the manifolds of digits with L_1/L_2-NCSAE are more separable than its counterpart as shown in Fig. 6(c), aiding the classifier to map out the separating boundaries among the digits more easily.

In the second experiment, SAE, NCSAE, L_1/L_2-NCSAE, and DpAE with 200 hidden nodes were trained using the NORB normalized-uniform data set. The NORB normalized-uniform data set, which is the second data set, contains



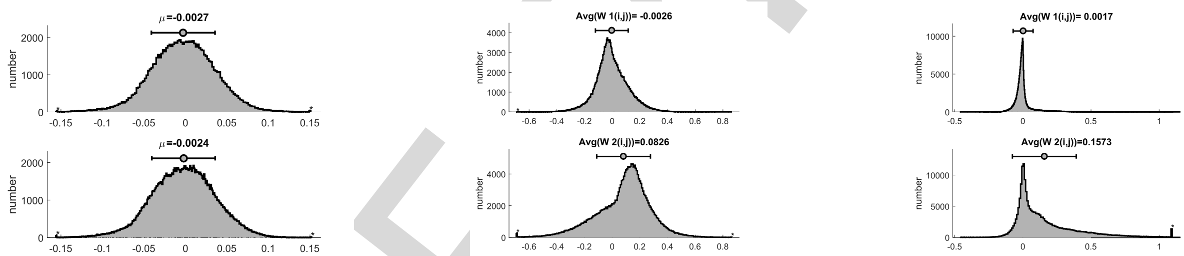
(a) SAE

(b) DpAE

(c) NCAE

(d) L_1/L_2-NCSAE

Fig. 7. Weights of randomly selected 90 out of 200 receptive filters of (a) SAE, (b) DpAE, (c) NCSAE, and (d) L_1/L_2-NCSAE using NORB data set. The range of weights are scaled to $[-1,1]$ and mapped to the graycolor map. $w <= -1$ is assigned to black, $w = 0$ to gray, and $w >= 1$ is assigned to white color.



(a) (b) (c)

Fig. 8. Distribution of 200 encoding ($\mathbf{W}^{(1)}$) and decoding filters ($\mathbf{W}^{(2)}$) weights learned from NORB data set using (a) DpAE, (b) NCSAE, and (c) L_1/L_2-NCSAE.

24 300 training images and 24 300 test images of 50 toys from five generic categories: four-legged animals, human figures, airplanes, trucks, and cars. The training and testing sets consist of five instances of each category. Each image consists of two channels, each of size 96×96 pixels. The inner 64×64 pixels of one of the channels cropped out and resized using bicubic interpolation to 32×32 pixels that form a vector with 1024 entries as the input. Randomly selected weights of 90 out of 200 neurons are plotted in Fig. 7. It can be seen that L_1/L_2-NCSAE learned more sparse features compared to features learned by all other AEs considered. The receptive fields learned by L_1/L_2-NCSAE captured the real actual edges of the toys while the edges captured by NCSAE are fuzzy, and those learned by DpAE and SAE are holistic. As shown in the weight distribution depicted in Fig. 8, L_1/L_2-NCSAE has both its encoding and decoding weights centered around zero with most of its weights positive when compared with those of DpAE and NCSAE that have weights distributed almost even on both sides of the origin.

B. Unsupervised Semantic Feature Learning From Text Data

In this experiment, DpAE, NCSAE, and L_1/L_2-NCSAE are evaluated and compared based on their ability to extract semantic features from text data, and how they are able to discover the underlined structure in text data. For this purpose, the Reuters-21578 text categorization data set with 200 features is utilized to train all the three types of AEs with 20 hidden nodes. A subset of 500 examples belonging to categories "grain," "crude," and "money-fx" was extracted from the test set. The experiments were run three times, averaged, and recorded. In Fig. 9, the 20-dimensional representations of the Reuters data subset using DpAE, NCSAE, and L_1/L_2-NCSAE are visualized. It can be observed that L_1/L_2-NCSAE is able to disentangle the documents into three distinct categories with more linear manifolds than NCSAE. In addition, L_1/L_2-NCSAE is able to group documents that are closer in the semantic space into the same categories than DpAE that finds it difficult to group the documents into any distinct categories with less overlap.

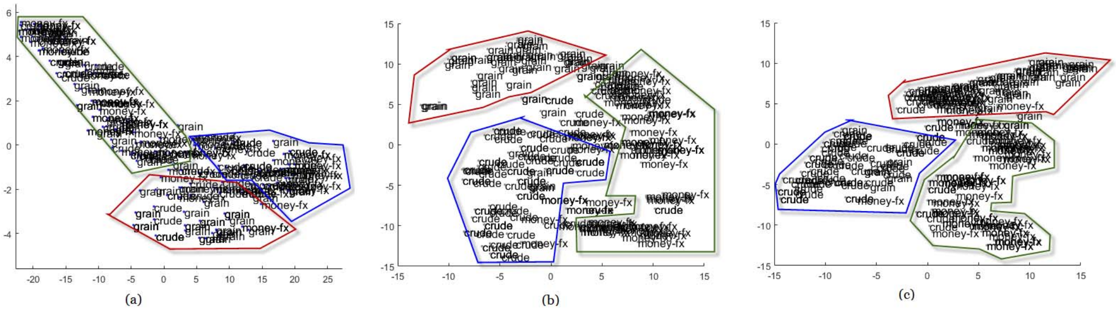


Fig. 9. Visualizing 20D representations of a subset of Reuters documents data using (a) DpAE, (b) NCSAE, and (c) L_1/L_2-NCSAE.

TABLE I

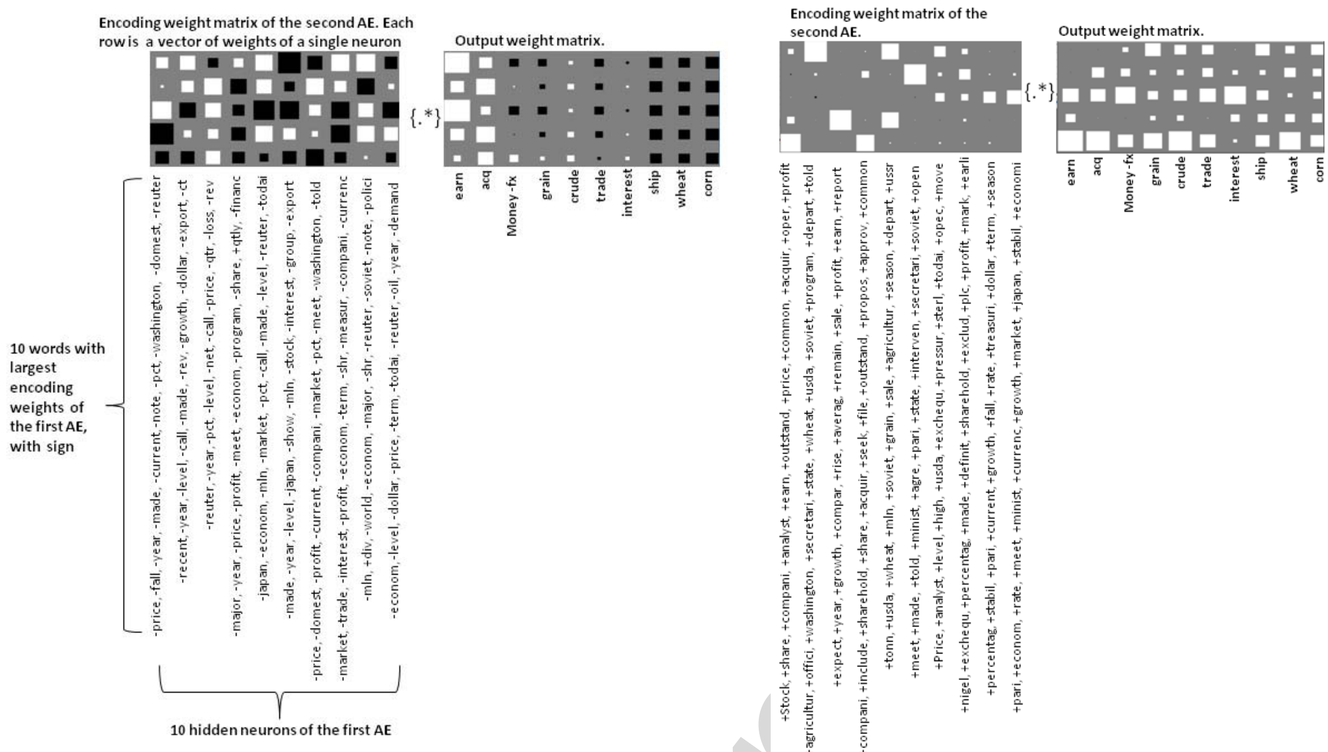
CLASSIFICATION ACCURACY ON MNIST AND NORB DATA SET

Dataset		*Before* fine-tuning		*After* fine-tuning		
		Mean ($\pm$ SD)	*p*-value	Mean ($\pm$ SD)	*p*-value	# Epochs
MNIST	SAE	0.735 $\pm$ 0.015	<0.001	**0.977** $\pm$ 0.0007	<0.001	400
	NCAE	0.844 ($\pm$0.0085)	0.0018	0.974 ($\pm$0.0012)	0.812	126
	NNSAE	0.702 ($\pm$0.027)	<0.0001	0.970 ($\pm$0.001)	<0.0001	400
	L_1/L_2-NCSAE	**0.847** ($\pm$0.0077)	-	0.974 ($\pm$0.0087)	-	84
	DAE (50% input dropout)	0.551 ($\pm$0.011)	<0.0001	0.972 ($\pm$0.0021)	0.034	400
	DpAE (50% hidden dropout)	0.172 ($\pm$0.0021)	<0.0001	0.964 ($\pm$0.0017)	<0.0001	400
	AAE	-	-	0.912 ($\pm$0.0016)	<0.0001	1000
NORB	SAE	0.562 $\pm$ 0.0245	<0.0001	0.814 $\pm$ 0.0099	0.041	400
	NCAE	**0.696** ($\pm$0.021)	0.406	**0.817** ($\pm$0.0095)	0.001	305
	NNSAE	0.208 ($\pm$0.025)	<0.0001	0.738 ($\pm$ 0.012)	<0.001	400
	L_1/L_2-NCSAE	0.695 ($\pm$0.0084)	-	0.812 ($\pm$0.0001)	-	196
	DAE (50% input dropout)	0.461 ($\pm$0.0019)	<0.0001	0.807 ($\pm$0.0015)	0.0103	400
	DpAE (50% hidden dropout)	0.491 ($\pm$0.0013)	<0.0001	0.815 ($\pm$0.0038)	<0.0001	400
	AAE	-	-	0.791 ($\pm$0.041)	<0.0001	1000

C. Supervised Learning

In the last set of experiments, a deep network was constructed using two stacked L_1/L_2-NCSAE and a softmax layer for classification to test if the enhanced ability of the network to shatter data into parts and lead to improved classification. Eventually, the entire deep network is fine-tuned to improve the accuracy of the classification. In this set of experiments, the performance of pretraining a deep network with L_1/L_2-NCSAE is compared with those pretrained with recent AE architectures. The MNIST and NORB data sets were utilized, and every run of the experiments is repeated ten times and averaged to combat the effect of random initialization. The classification accuracy of the deep network pretrained with NNSAE [18], DpAE [19], DAE [32], AAE [22], NCSAE, and L_1/L_2-NCSAE using MNIST and NORB data, respectively, are detailed in Table I. The network architectures are 784-196-20-10 and 1024-200-20-5 for MNIST and NORB data set, respectively. It is remarked that for training of AAE with two layers of 196 hidden units in the encoder, decoder, discriminator, and other hyperparameters tuned as described in [22], the accuracy was 83.67%. The AAE reported in Table I used encoder, decoder, and discriminator each with two layers of 1000 hidden units and trained for 1000 epochs. The classification accuracy and speed of convergence are the figures of merit used to benchmark L_1/L_2-NCSAE with other AEs.

It is observed from the result that L_1/L_2-NCSAE-based deep network gives an improved accuracy before fine-tuning compared to methods such as NNSAE, NCSAE, and DpAE. However, the performance in terms of classification accuracy after fine-tuning is very competitive. In fact, it can be inferred from the p-value of the experiments conducted on MNIST and NORB in Table I that there is no significant difference in the accuracy after fine-tuning between NCSAE and L_1/L_2-NCSAE even though most of the weights in L_1/L_2-NCSAE are non-negativity constrained. Therefore, it is remarked that even though the interpretability of the deep network has been fostered by constraining most of the weights to be non-negative and sparse, nothing significant has been lost in terms of accuracy. In addition, network trained with L_1/L_2-NCSAE was also observed to converge faster than its counterparts. On the other hand, NNSAE also has non-negative weights but with deterioration in accuracy, which is more conspicuous especially before the fine-tuning stage. The improved accuracy before fine-tuning in L_1/L_2-NCSAE-based network can be traced to its ability to decompose data more into distinguishable parts. Although the performance of L_1/L_2-NCSAE after fine-tuning is similar to those of DAE



(a) (b)

Fig. 10. Deep network trained on Reuters-21578 data using (a) DpAE and (b) L_1/L_2-NCSAE. The area of each square is proportional to the weight's magnitude. The range of weights is scaled to [−1,1] and mapped to the graycolor map. $w = -1$ is assigned to black, $w = 0$ to gray, and $w = 1$ is assigned to white color.

and NCSAE but better than NNSAE, DpAE, and AAE, L_1/L_2-NCSAE constrains most of the weights to be nonnegative and sparse to foster transparency than for other AEs. However, DpAE and NCSAE performed slightly more accurate than L_1/L_2-NCSAE on NORB after network fine-tuning.

In light of constructing an interpretable deep network, an L_1/L_2-NCSAE pretrained deep network with 10 hidden neurons in the first AE layer, 5 hidden neurons in the second AE, and 10 output neurons (one for each category) in the softmax layer was constructed. It was trained on Reuters data and compared with that pretrained using DpAE. The interpretation of the encoding layer of the first AE is provided by listing words associated with 10 strongest weights, and the interpretation of the encoding layer of the second AE is portrayed as images characterized by both the magnitude and sign of the weights. Compared to the AE with weights of both signs shown in Fig. 10(a), Fig. 10(b) allows for much better insight into the categorization of the topics.

Topic *earn* in the output weight matrix resonates with the fifth hidden neuron most, lesser with the third, and somewhat with the fourth. This resonance can happen only when the fifth hidden neuron reacts to input by words of columns 1 and 4, and in addition, to a lesser degree, when the third hidden neuron reacts to input by words of the third column of words. So, in tandem, the dominant columns 1 and 4 and then also 3 are sets of words that trigger the category *earn*.

Analysis of the term words for the topic *acq* leads to a similar conclusion. This topic also resonates with the two dominant hidden neurons 5 and 3 and somewhat with neuron 2. These neurons 5 and 3 are driven again by the columns of words 1, 4, and 3. The difference between the categories is now that to a lesser degree, the category *acq* is influenced by the sixth column of words. An interesting point is in contribution of the third column of words. The column connects only to the fourth hidden neuron but weights from this neuron in the output layer are smaller and hence less significant than for any other of the five neurons (or rows) of the output weight matrix. Hence, this column is of least relevance in the topical categorization.

D. Experiment Running Times

The training time for networks with and without the nonnegativity constraints was compared. The constrained network converges faster and requires a lesser number of training epochs. In addition, the unconstrained network requires more time per epoch than the constrained one. The running time experiments were performed using full MNIST benchmark data set on Intel(r) Core i7-6700 CPU at 3.40 Ghz and a 64 GB of RAM running a 64-b Windows 10 Enterprise edition. The software implementation has been with MATLAB 2015b with batch gradient descent method, and LBFGS in minFunc [33] is used to minimize the objective function. The usage times for constrained and unconstrained networks were also compared. We consider the usage time in milliseconds (ms), and as the time elapsed in ms, a fully trained deep network requires to classify all the test samples. The unconstrained network took 48 ms per epoch in the training phase, while the constrained counterpart took 46 ms. Also, the unconstrained network required 59.9-ms usage time, whereas the network with non-negative weights took 55 ms. From the above observations, it is remarked that the non-negativity constraint simplifies the resulting network.

V. Conclusion

This paper addresses the concept and properties of special regularization of DL AE that takes advantage of non-negative encodings and at the same time of special regularization. It has been shown that using both L_1 and L_2 to penalize the negative weights, most of them are forced to be non-negative and sparse, and hence, the network interpretability is enhanced. In fact, it is also observed that most of the weights in the Softmax layer become non-negative and sparse. In sum, it has been observed that encouraging non-negativity in NCSAE-based deep architecture forces the layers to learn part-based representation of their input and leads to a comparable classification accuracy before fine-tuning the entire deep network and not-so-significant accuracy deterioration after fine-tuning. It has also been shown on select examples that concurrent L_1 and L_2 regularizations improve the network interpretability. The performance of the proposed method was compared in terms of sparsity, reconstruction error, and classification accuracy with the conventional SAE and NCSAE, and we utilized MNIST handwritten digits, Reuters documents, and the NORB data set to illustrate the proposed concepts.

References

[1] Y. Bengio and Y. LeCun, "Scaling learning algorithms towards AI," *Large-Scale Kernel Mach.*, vol. 34, no. 5, pp. 1–41, 2007.

[2] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.

[3] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[4] L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning," *APSIPA Trans. Signal Inf. Process.*, vol. 3, p. e2, 2014. AQ:3

[5] S. Bengio, L. Deng, H. Larochelle, H. Lee, and R. Salakhutdinov, "Guest editors' introduction: Special section on learning deep architectures," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1795–1797, Aug. 2013.

[6] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in Neural Information Processing Systems*, vol. 19. Cambridge, MA, USA: MIT Press, 2007, p. 153.

[7] B. Ayinde and J. Zurada, "Clustering of receptive fields in autoencoders," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2016, pp. 1310–1317.

[8] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, Oct. 1999.

[9] J. Chorowski and J. M. Zurada, "Learning understandable neural networks with nonnegative weight constraints," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 1, pp. 62–69, Jan. 2015. AQ:4

[10] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607–609, 1996.

[11] E. Hosseini-Asl, J. M. Zurada, and O. Nasraoui, "Deep learning of part-based representation of data using sparse autoencoders with nonnegativity constraints," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 12, pp. 2486–2498, Dec. 2016.

[12] M. Ranzato, Y.-L. Boureau, and Y. LeCun, "Sparse feature learning for deep belief networks," in *Advances in Neural Information Processing Systems*, vol. 20. Red Hook, NY, USA: Curran Associates, 2007, pp. 1185–1192.

[13] M. Ishikawa, "Structural learning with forgetting," *Neural Netw.*, vol. 9, no. 3, pp. 509–521, Apr. 1996.

[14] P. L. Bartlett, "The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network," *IEEE Trans. Inf. Theory*, vol. 44, no. 2, pp. 525–536, Mar. 1998.

[15] G. Gnecco and M. Sanguineti, "Regularization techniques and suboptimal solutions to optimization problems in learning from data," *Neural Comput.*, vol. 22, no. 3, pp. 793–829, 2010.

[16] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Advances in Neural Information Processing Systems*, vol. 4. San Mateo, CA, USA: Morgan Kaufmann, 1995, pp. 950–957.

[17] O. E. Ogundijo, A. Elmas, and X. Wang, "Reverse engineering gene regulatory networks from measurement with missing values," *EURASIP J. Bioinform. Syst. Biol.*, vol. 2017, no. 1, p. 2, 2017. AQ:5

[18] A. Lemme, R. F. Reinhart, and J. J. Steil, "Online learning and generalization of parts-based image representations by non-negative sparse autoencoders," *Neural Netw.*, vol. 33, pp. 194–203, Sep. 2012.

[19] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. (Jul. 2012). "Improving neural networks by preventing co-adaptation of feature detectors." [Online]. Available: https://arxiv.org/abs/1207.0580

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[21] D. P. Kingma and M. Welling. (Dec. 2013). "Auto-encoding variational Bayes." [Online]. Available: https://arxiv.org/abs/1312.6114

[22] A. Makhzani, J. Shlens, N. Jaitly, and I. Goodfellow. (Nov. 2015). "Adversarial autoencoders." [Online]. Available: https://arxiv.org/abs/1511.05644

[23] Y. Burda, R. Grosse, and R. Salakhutdinov. (Sep. 2015). "Importance weighted autoencoders." [Online]. Available: https://arxiv.org/abs/1509.00519

[24] B. O. Ayinde, E. Hosseini-Asl, and J. M. Zurada, "Visualizing and understanding nonnegativity constrained sparse autoencoder in deep learning," in *Artificial in Intelligence and Soft Computing* (Lecture Notes in Computer Science), vol. 9692, L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. Zadeh, and J. Zurada, Eds. Springer, 2016, pp. 3–14. AQ:6

[25] S. J. Wright and J. Nocedal, *Numerical Optimization*, vol. 2. New York, NY, USA: Springer, 1999.

[26] T. D. Nguyen, T. Tran, D. Phung, and S. Venkatesh, "Learning parts-based representations with nonnegative restricted Boltzmann machine," in *Proc. Asian Conf. Mach. Learn.*, 2013, pp. 133–148.

[27] A. Ng. Sparse Autoencoder. *CS294A Lecture Notes*. Stanford University, (2011). [Online]. Available: https://web.stanford.edu/class/cs294a/sparseAutoencoder_2011new.pdf

[28] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

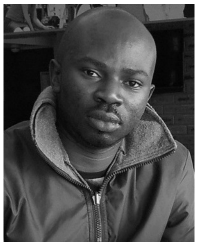
[29] Y. LeCun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2. Jul. 2004, pp. II-97–II-104.

[30] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*, vol. 1. Boston, MA, USA: Addison-Wesley, 2006.

[31] L. V. der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 11, pp. 2579–2605, 2008.

[32] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 1096–1103.

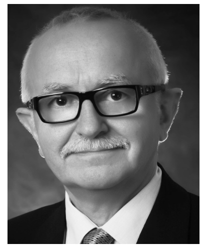
[33] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM J. Sci. Comput.*, vol. 16, no. 5, pp. 1190–1208, 1995.



Babajide O. Ayinde (S'09) received the M.Sc. degree in engineering systems and control from the King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia. He is currently pursuing the Ph.D. degree with the University of Louisville, Louisville, KY, USA.

His current research interests include unsupervised feature learning and deep learning techniques and applications.

Mr. Ayinde was a recipient of the University of Louisville fellowship.



Jacek M. Zurada (M'82–SM'83–F'96–LF'14) received the Ph.D. degree from the Gdansk Institute of Technology, Gdansk, Poland.

He currently serves as a Professor of electrical and computer engineering with the University of Louisville, Louisville, KY, USA. He has authored or co-authored several books and over 380 papers in computational intelligence, neural networks, machine learning, logic rule extraction, and bioinformatics, and delivered over 100 presentations throughout the world.

Dr. Zurada ha s been a Board Member of the IEEE CIS and IJCNN. AQ:7 He was a recipient of the 2013 Joe Desch Innovation Award, the 2015 Distinguished Service Award, and five honorary professorships. He served as the IEEE V-President and the Technical Activities Board (TAB) Chair in 2014. He was the Chair of the IEEE TAB Periodicals Committee and the TAB Periodicals Review and Advisory Committee. From 2004 to 2005, he was the President of the IEEE Computational Intelligence Society. He was the Editor-in-Chief of the IEEE TRANSACTIONS ON NEURAL NETWORKS (1997–2003) and an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, NEURAL NETWORKS and the PROCEEDINGS OF THE IEEE. He is an Associate Editor of Neurocomputing and several other journals.

AUTHOR QUERIES

AUTHOR PLEASE ANSWER ALL QUERIES

PLEASE NOTE: We cannot accept new source files as corrections for your paper. If possible, please annotate the PDF proof we have sent you with your corrections and upload it via the Author Gateway. Alternatively, you may send us your corrections in list format. You may also upload revised graphics via the Author Gateway.

AQ:1 = Please note that the “Non-negativity Constrained Sparse Autoencoder” and “model predictive control” have been provided as the expansions for MPC. Please confirm which one to follow.

AQ:2 = The abbreviation NCAE has been changed to NCSAE throughout the document. Since the same expansion “Non-negativity Constrained Sparse Autoencoder” is given for both the abbreviation. Please confirm.

AQ:3 = Please provide the issue no. or month for ref. [4].

AQ:4 = Please note that there were discrepancies between the accepted pdf [TNNLS-2016-P-6572.pdf] and the [TNNLS-2016-P-6572.R2.tex] in reference section. We have followed [TNNLS-2016-P-6572.R2.tex].

AQ:5 = Please confirm the volume no. for ref. [17].

AQ:6 = Please confirm the volume no. for ref. [24]. also provide the publisher location.

AQ:7 = Please confirm whether the edits made in the sentence “Dr. Zurada has been….” are OK.

Deep Learning of Constrained Autoencoders for Enhanced Understanding of Data

Babajide O. Ayinde, *Student Member, IEEE*, and Jacek M. Zurada, *Life Fellow, IEEE*

***Abstract*— Unsupervised feature extractors are known to perform an efficient and discriminative representation of data. Insight into the mappings they perform and human ability to understand them, however, remain very limited. This is especially prominent when multilayer deep learning architectures are used. This paper demonstrates how to remove these bottlenecks within the architecture of non-negativity constrained autoencoder. It is shown that using both L1 and L2 regularizations that induce non-negativity of weights, most of the weights in the network become constrained to be non-negative, thereby resulting into a more understandable structure with minute deterioration in classification accuracy. Also, this proposed approach extracts features that are more sparse and produces additional output layer sparsification. The method is analyzed for accuracy and feature interpretation on the MNIST data, the NORB normalized uniform object data, and the Reuters text categorization data set.**

***Index Terms*— Deep learning (DL), part-based representation, receptive field, sparse autoencoder (SAE), white-box model.**

I. INTRODUCTION

DEEP learning (DL) networks take the form of heuristic and rich architectures that develop unique intermediate data representation. The complexity of architectures is reflected by both the sizes of layers and, for a large number of data sets reported in the literature, also by the processing. In fact, the architectural complexity and the excessive number of weights and units are often built into the DL data representation by design and are deliberate [1]–[5]. Although deep architectures are capable of learning highly complex mappings, they are difficult to train, and it is usually hard to interpret what each layer has learned. Moreover, gradient-based optimization with random initialization used in training is susceptible to converging to local minima [6], [7].

In addition, it is generally believed that humans analyze complex interactions by breaking them into isolated and understandable hierarchical concepts. The emergence of part-based representation in human cognition can be conceptually tied to the non-negativity constraints [8]. One way to enable easier human understandability of concepts in neural networks is to constrain the network's weights to be non-negative. Note that such representation through non-negative weights of a multilayer network perceptron can implement any shattering of points provided suitable negative bias values are used [9].

Drawing inspiration from the idea of non-negative matrix factorization (NMF) and sparse coding [8], [10], the hidden structure of data can be unfolded by learning features that have capabilities to model the data in parts. Although NMF enforces the encoding of both the data and features to be non-negative thereby resulting in additive data representation, however, incorporating sparse coding within NMF for the purpose of encoding data is computationally expensive, while with AEs, this incorporation is learning-based and fast. In addition, the performance of a deep network can be enhanced using non-negativity constrained sparse autoencoder (NCSAE) with part-based data representation capability [11], [12].

It is remarked that weight regularization is a concept that has been employed both in the understandability and generalization context. It is used to suppress the magnitudes of the weights by reducing the sum of their squares. Enhancement in sparsity can also be achieved by penalizing the sum of absolute values of the weights rather than the sum of their squares [13]–[17]. In this paper, the work proposed in [11] is extended by modifying the cost function to extract more sparse features, encouraging the non-negativity of the network weights, and enhancing the understandability of the data. Other related model is the non-negative sparse autoencoder (NNSAE) trained with an online algorithm with tied weights and linear output activation function to mitigate the training hassle [18]. While Lemme *et al.* [18] uses a piecewise linear decay function to enforce non-negativity and focuses on shallow architecture, the proposed uses a composite norm with focus on deep architectures. Dropout is another recently introduced and widely used heuristic to sparsify AEs and prevent overfitting by randomly dropping units and their connections from the neural network during training [19], [20].

More recently, different paradigms of AEs that constrain the output of encoder to follow a chosen prior distribution have been proposed [21]–[23]. In variational autoencoding, the decoder is trained to reconstruct the input from samples that follow chosen prior using variational inference [21]. Realistic data points can be reconstructed in the original data space by feeding the decoder with samples from chosen prior distribution. On the other hand, adversarial AE matches the encoder's output distribution to an arbitrary prior distribution using adversarial training with discriminator and the

Manuscript received June 1, 2016; revised December 19, 2016 and June 12, 2017; accepted August 14, 2017. This work was supported by NSF under Grant 1641042. *(Corresponding author: Jacek M. Zurada.)*

B. O. Ayinde is with the Department of Electrical and Computer Engineering, University of Louisville, Louisville, KY 40292 USA.

J. M. Zurada is with the Department of Electrical and Computer Engineering, University of Louisville, Louisville, KY 40292 USA, and also with the Information Technology Institute, University of Social Science, 90-113 Łódz, Poland (e-mail: jacek.zurada@louisville.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2017.2747861

generator [22]. Upon adversarial training, encoder learns to map data distribution to the prior distribution.

The problem addressed here is threefold.

1) The interpretability of AE-based deep layer architecture fostered by enforcing high degree of weight's non-negativity in the network. This improves on NCSAEs that show negative weights despite imposing non-negativity constraints on the network's weights [11].
2) It is demonstrated how the proposed architecture can be utilized to extract meaningful representations that unearth the hidden structure of a high-dimensional data.
3) It is shown that the resulting non-negative AEs do not deteriorate their classification performance.

This paper considerably expands the scope of the AE model first introduced in [24] by: 1) introducing smoothing function for L_1 regularization for numerical stability; 2) illustrating the connection between the proposed regularization and weights' non-negativity; 3) drawing more insight into a variety of data set; 4) comparing the proposed with recent AE architectures; and 5) supporting the interpretability claim with new experiments on text categorization data. This paper is structured as follows. Section II introduces the network configuration and the notation for non-negative sparse feature extraction. Section III discusses the experimental designs and Section IV presents the results. Finally, conclusions are drawn in Section V.

II. Non-Negative Sparse Feature Extraction Using Constrained Autoencoders

As shown in [8], one way of representing data is by shattering it into various distinct pieces in a manner that additive merging of these pieces can reconstruct the original data. Mapping this intuition to AEs, the idea is to sparsely disintegrate data into parts in the encoding layer and subsequently additively process the parts to recombine the original data in the decoding layer. This disintegration can be achieved by imposing non-negativity constraint on the network's weights [11], [25], [26].

A. L_1/L_2-Non-Negativity Constrained Sparse Autoencoder (L_1/L_2-NCSAE)

In order to encourage higher degree of non-negativity in network's weights, a composite penalty term (1) is added to the objective function, resulting in the cost function expression for L_1/L_2-NCSAE

$$\begin{aligned} &J_{L_1/L_2\text{-NCSAE}}(\mathbf{W},\mathbf{b}) \\ &\quad = J_{AE} + \beta \sum_{r=1}^{n'} D_{\mathrm{KL}}\left(p \,\middle\|\, \frac{1}{m}\sum_{k=1}^{m} h_r(\mathbf{x}^{(k)})\right) \\ &\qquad + \sum_{l=1}^{2}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}} f_{L_1/L_2}\big(w_{ij}^{(l)}\big) \end{aligned} \tag{1}$$

where $\mathbf{W} = \{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}\}$ and $\mathbf{b} = \{\mathbf{b}_x, \mathbf{b}_h\}$ represent the weights and biases of encoding and decoding layers, respectively; s_l is the number of neurons in layer l; and $w_{ij}^{(l)}$ represents the connection between jth neuron in layer $l-1$ and ith neuron in layer l and for given input $\mathbf{x}$

$$J_{\mathrm{AE}} = \frac{1}{m}\sum_{k=1}^{m}\left\|\sigma\left(\mathbf{W}^{(2)}\sigma\left(\mathbf{W}^{(1)}\mathbf{x}^{(k)} + \mathbf{b}_x\right) + \mathbf{b}_h\right) - \mathbf{x}^{(k)}\right\|_2^2 \tag{2}$$

where m is the number of training examples, $||\,.\,||_2$ is the Euclidean norm, $D_{\mathrm{KL}}(.)$ is the Kullback–Leibler (KL) divergence for sparsity control [27] with p denoting the desired activation and the average activations of hidden units, n' is the number of hidden units, $h_j(\mathbf{x}^{(k)}) = \sigma(\mathbf{W}_j^{(1)}\mathbf{x}^{(k)} + b_{x,j})$ denotes the activation of hidden unit j due to input $\mathbf{x}^{(k)}$, and $\sigma(.)$ is the element-wise application of the logistic sigmoid, $\sigma(\mathbf{x}) = 1/(1+exp(-\mathbf{x}))$, β controls the sparsity penalty term, and

$$f_{L_1/L_2}(w_{ij}) = \begin{cases} \alpha_1\Gamma(w_{ij},\kappa) + \dfrac{\alpha_2}{2}||w_{ij}||^2 & w_{ij} < 0 \\ 0 & w_{ij} \geq 0 \end{cases} \tag{3}$$

where α_1 and α_2 are L_1 and L_2 non-negativity-constraint weight penalty factors, respectively. p, β, α_1, and α_2 are experimentally set to 0.05, 3, 0.0003, and 0.003, respectively, using 9000 randomly sampled images from the training set as a held-out validation set for hyperparameter tuning, and the network is retrained on the entire data set. The weights are updated as below using the error backpropagation

$$w_{ij}^{(l)} = w_{ij}^{(l)} - \xi\frac{\partial}{\partial w_{ij}^{(l)}} J_{L_1/L_2\text{-NCSAE}}(\mathbf{W},\mathbf{b}) \tag{4}$$

$$b_i^{(l)} = b_i^{(l)} - \xi\frac{\partial}{\partial b_i^{(l)}} J_{L_1/L_2\text{-NCSAE}}(\mathbf{W},\mathbf{b}) \tag{5}$$

where $\xi > 0$ is the learning rate and the gradient of L_1/L_2-NCSAE loss function is computed as

$$\begin{aligned} &\frac{\partial}{\partial w_{ij}^{(l)}} J_{L_1/L_2\text{-NCSAE}}(\mathbf{W},\mathbf{b}) \\ &\quad = \frac{\partial}{\partial w_{ij}^{(l)}} J_{\mathrm{AE}}(\mathbf{W},\mathbf{b}) \\ &\qquad + \beta\frac{\partial}{\partial w_{ij}^{(l)}} D_{KL}\left(p \,\middle\|\, \frac{1}{m}\sum_{k=1}^{m} h_j(\mathbf{x}^{(k)})\right) + g\big(w_{ij}^{(l)}\big) \end{aligned} \tag{6}$$

where $g(w_{ij})$ is a composite function denoting the derivative of $f_{L_1/L_2}(w_{ij})$ (3) with respect to w_{ij} as

$$g(w_{ij}) = \begin{cases} \alpha_1\nabla_{\mathbf{w}}\left\|w_{ij}\right\| + \alpha_2 w_{ij} & w_{ij} < 0 \\ 0 & w_{ij} \geq 0. \end{cases} \tag{7}$$

Although the penalty function in (1) is an extension of NCSAE (obtained by setting α_1 to zero), a close scrutiny of the weight distribution of both the encoding and decoding layer in NCSAE reveals that many weights are still not non-negative despite imposing non-negativity constraints. The reason for this is that the original L_2 norm used in NCSAE penalizes the negative weights with bigger magnitudes stronger than those with smaller magnitudes. This forces a good number of the weights to take on small negative values. This paper uses additional L_1 to even out this occurrence, that is, the

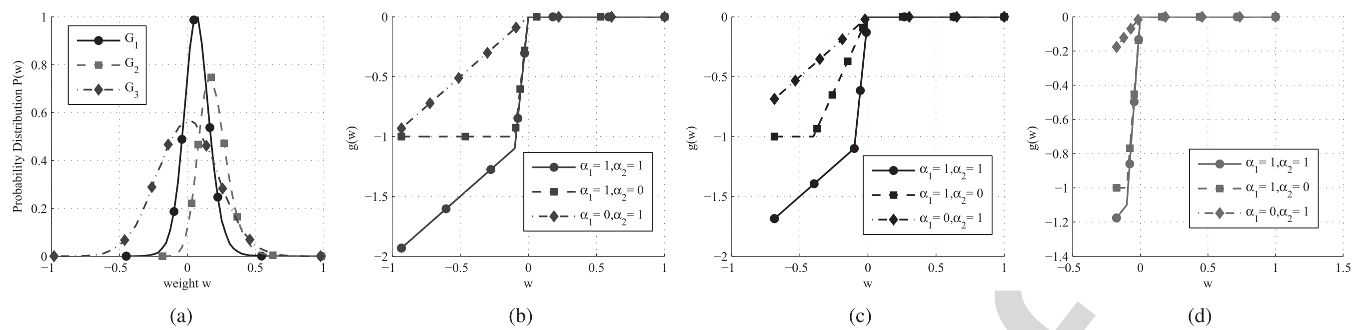


Fig. 1. (a) Symmetric (G_3) and skewed (G_1 and G_2) weight distributions. Decay function with three values of α_1 and α_2 for weight distribution. (b) G_3. (c) G_1. (d) G_2.

L_1 penalty forces most of the negative weights to become non-negative.

B. Implication of Imposing Non-Negative Parameters With Composite Decay Function

The graphical illustration of the relation between the weight distribution and the composite decay function is shown in Fig. 1. Ideally, addition of Frobenius norm of the weight matrix ($\alpha||\mathbf{W}||_F^2$) to the reconstruction error in (2) imposes a Gaussian prior on the weight distribution as shown in curve G_3 in Fig. 1(a). However, using the composite function in (3) results in imposition of positively skewed deformed Gaussian distribution as in curves G_1 and G_2. The degree of non-negativity can be adjusted using parameters α_1 and α_2. Both parameters have to be carefully chosen to enforce non-negativity while simultaneously ensuring good supervised learning outcomes. The effect of L_1 ($\alpha_2 = 0$), L_2 ($\alpha_1 = 0$), and L_1/L_2 ($\alpha_1 \neq 0$ and $\alpha_2 \neq 0$), non-negativity penalty terms, on weight updates for weight distributions G_1, G_2, and G_3 are, respectively, shown in Fig. 1(b)–(d). It can be observed for all the three distributions that L_1/L_2 regularization enforces stronger weight decay than individual L_1 and L_2 regularizations. Other observation from Fig. 1 is that the more positively skewed the weight distribution becomes, the lesser the weight decay function.

The consequences of minimizing (1) are that: 1) the average reconstruction error is reduced; 2) the sparsity of the hidden layer activations is increased because more negative weights are forced to zero, thereby leading to sparsity enhancement; and 3) the number of non-negative weights is also increased. The resultant effect of penalizing the weights simultaneously with L_1 and L_2 norms is that large positive connections are preserved while their magnitudes are shrunk. However, the L_1 norm in (3) is non-differentiable at the origin, and this can lead to numerical instability during simulations. To circumvent this drawback, one of the well-known smoothing function that approximates L_1 norm as in (3) is utilized. Given any finite dimensional vector $\mathbf{z}$ and positive constant κ, the following smoothing function approximates L_1 norm:

$$\Gamma(\mathbf{z},\kappa) = \begin{cases} ||\mathbf{z}|| & ||\mathbf{z}|| > \kappa \\ \dfrac{||\mathbf{z}||^2}{2\kappa} + \dfrac{\kappa}{2} & ||\mathbf{z}|| \leq \kappa \end{cases} \quad (8)$$

with gradient

$$\nabla_{\mathbf{z}}\Gamma(\mathbf{z},\kappa) = \begin{cases} \dfrac{\mathbf{z}}{||\mathbf{z}||} & ||\mathbf{z}|| > \kappa \\ \dfrac{\mathbf{z}}{\kappa} & ||\mathbf{z}|| \leq \kappa. \end{cases} \quad (9)$$

For convenience, we adopt (8) to smoothen the L_1 penalty function and κ is experimentally set to 0.1.

III. Experiments

In the experiments, three data sets are used, namely, MNIST [28], NORB normalized-uniform [29], and Reuters-21578 text categorization data set. The Reuters-21578 text categorization data set comprises documents that featured in 1987 Reuters newswire. The ModApte split was employed to limit the data set to 10 most frequent classes. The ModApte split was utilized to limit the categories to 10 most frequent categories. The bag-of-words format that has been stemmed and stop-word removed was used (see http://people.kyb.tuebingen.mpg.de/pgehler/rap/ for further clarification). The data set contains 11 413 documents with 12 317 dimensions. Two techniques were used to reduce the dimensionality of each document in order to preserve the most informative and less correlated words [30]. To reduce the dimensionality of each document to contain the most informative and less correlated words, words were first sorted based on their frequency of occurrence in the data set. Words with frequency below 4 and above 70 were then eliminated. The most informative words that do not occur in every topic were selected based on information gain with the class attribute. The remaining words (or features) in the data set were sorted using this method, and the less important features were removed based on the desired dimension of documents. In this paper, the length of the feature vector for each of the documents was reduced to 200.

In the preliminary experiment, the subsets 1, 2, and 6 from the MNIST handwritten digits are extracted for the purpose of understanding how the deep network constructed using L_1/L_2-NCSAE processes and classify their input. For easy interpretation, a small deep network was constructed and trained by stacking two AEs with 10 hidden neurons each and 3 softmax neurons. The number of hidden neurons was chosen to obtain reasonably good classification accuracy

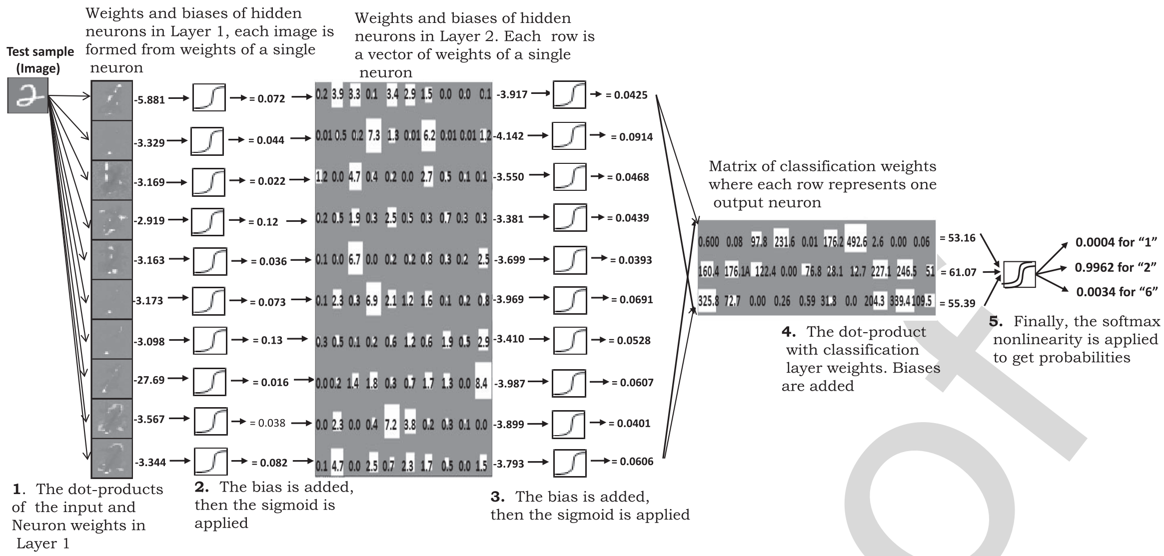


Fig. 2. Filtering the signal through the L_1/L_2-NCSAE trained using the reduced MNIST data set with class labels 1, 2, and 6. The test image is a 28×28 pixels image unrolled into a vector of 784 values. Both the input test sample and the receptive fields of the first autoencoding layer are presented as images. The weights of the output layer are plotted as a diagram with one row for each output neuron and one column for every hidden neuron in $(L-1)$th layer. The architecture is 784-10-10-3. The range of weights is scaled to $[-1, 1]$ and mapped to the graycolor map. $w = -1$ is assigned to black, $w = 0$ to gray, and $w = 1$ is assigned to white color. That is, black pixels indicate negative, gray pixels indicate zero-valued weights, and white pixels indicate positive weights.

while keeping the network reasonably small. The network is intentionally kept small because the full MNIST data would require larger hidden layer size and this may limit network interpretability. An image of digit 2 is then filtered through the network, and it can be observed in Fig. 2 that sparsification of the weights in all the layers is one of the aftermath of non-negativity constraints imposed on the network. Another observation is that most of the weights in the network have been confined to non-negative domain, which removes the opaqueness of the DL process. It can be seen that the fourth and seventh receptive fields of the first AE layer have dominant activations (with activation values 0.12 and 0.13, respectively) and they capture most information about the test input. Also, they are able to filter distinct part of input digit. The outputs of the first layer sigmoid constitute higher level features extracted from test image with emphasis on the fourth and seventh features. Subsequently, in the second layer, the second, sixth, eight, and tenth neurons have dominant activations (with activation values 0.0914, 0.0691, 0.0607, and 0.0606, respectively) because they have stronger connections with the dominant neurons in the first layer than the rest. Last, in the softmax layer, the second neuron was 99.62% activated because it has the strongest connections with the dominant neurons in the second layer, thereby classifying the test image as "2."

The fostering of interpretability is also demonstrated using a subset of NORB normalized-uniform data set [29] with class labels "four-legged animals," "human figures," and "airplanes." The 1024-10-5-3 network configuration was trained on the subset of the NORB data using two stacked L_1/L_2-NCSAEs and a Softmax layer. Fig. 3(b) shows the randomly sampled test patterns, and the weights and activations of the first and second AE layers are shown in Fig. 3(a). The bar charts indicate the activations of hidden units for the sample input patterns. The features learned by units in each layer are localized, sparse, and allow easy interpretation of isolated data parts. The features mostly show non-negative weights making it easier to visualize to what input object patterns they respond. It can be seen that units in the network discriminate among objects in the images and react differently to input patterns. Third, sixth, eight, and ninth hidden units of layer 1 capture features that are common to objects in class "2" and react mainly to them as shown in the first layer activations. Also, the features captured by the second layer activations reveal that the second and fifth hidden units are mainly stimulated by objects in class "2."

The outputs of Softmax layer represent the *a posteriori* class probabilities for a given sample and are denoted as Softmax scores. An important observation from Fig. 3(a)–(c) is that hidden units in both layers did not capture significant representative features for class "1" white color-coded test sample. This is one of the reasons why it is misclassified into class "3" with a probability of 0.57. The argument also goes for class "1" dark-gray color-coded test sample misclassified into class "3" with a probability of 0.60. In contrast, hidden units in both layers capture significant representative features for class "2" test samples of all color codes. This is why all class "2" test samples are classified correctly with high probabilities as shown in Fig. 3(d). Last, the network contains a good number of representative features for class "3" test samples and was able to classify 4 out of 5 correctly as given in Fig. 3(e).

IV. Results and Discussion

A. Unsupervised Feature Learning of Image Data

In the first set of experiments, three-layer L_1/L_2-NCSAE, NCSAE [11], DpAE [19], and conventional SAE network with 196 hidden neurons were trained using MNIST data set

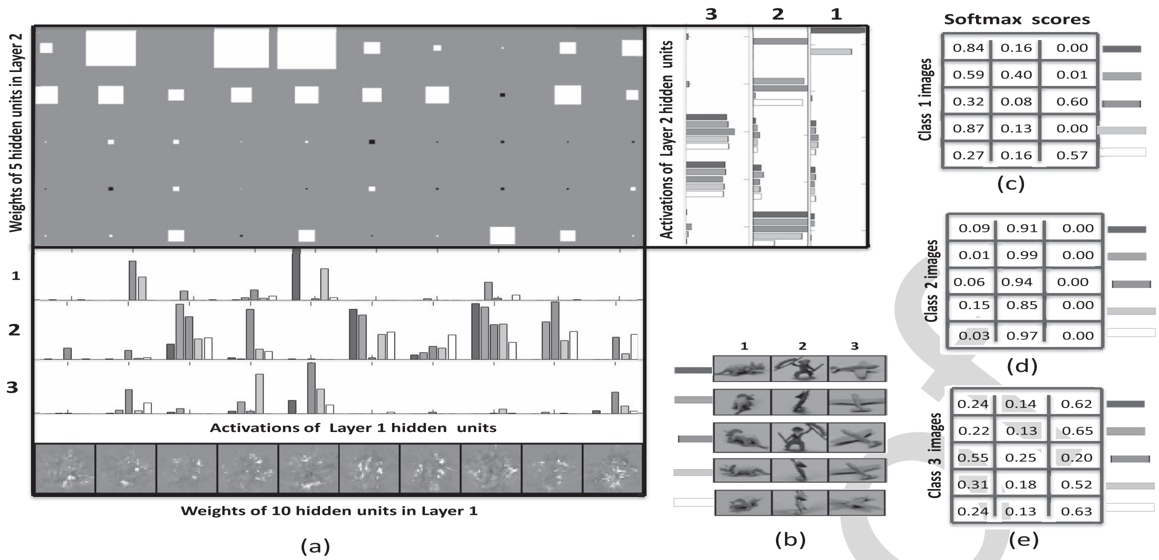


Fig. 3. Weights were trained using two stacked L_1/L_2-NCSAEs. RFs learned from the reduced NORB data set are plotted as images at the bottom part of (a). The intensity of each pixel is proportional to the magnitude of the weight connected to that pixel in the input image with negative value indicating black, positive values white, and the value 0 corresponding to gray. The biases are not shown. The activations of first layer hidden units for the NORB objects presented in (b) are depicted on the bar chart on top of the RFs. The weights of the second layer AE are plotted as a diagram at the topmost part of (a). Each row of the plot corresponds to the weight of each hidden unit of second AE and each column for weight of every hidden unit of the first layer AE. The magnitude of the weight corresponds to the area of each square; white indicates positive, gray indicates zero, and black negative sign. The activations of second layer hidden units are shown as bar chart in the right-hand side of the second layer weight diagram. Each column shows the activations of each hidden unit for five color-coded examples of the same object. The outputs of Softmax layer for color-coded test objects with class labels (c) "fourlegged animals" tagged as class 1, (d) "human figures" as class 2, and (e) "airplanes" as class 3.

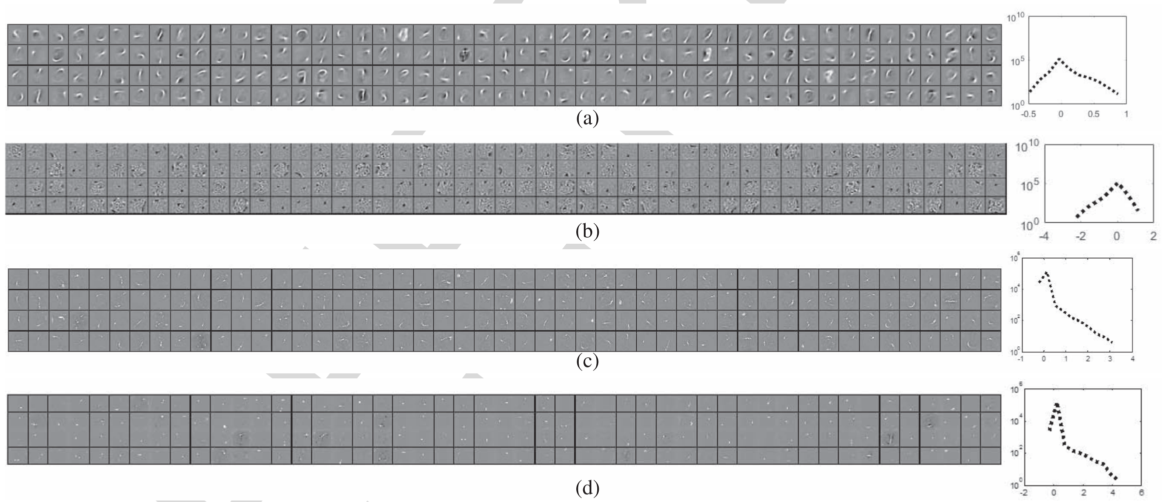


Fig. 4. 196 receptive fields ($\mathbf{W}^{(1)}$) with weight histograms learned from MNIST digit data set using (a) SAE, (b) DpAE, (c) NCSAE, and (d) L_1/L_2-NCSAE. Black pixels indicate negative, and white pixels indicate positive weights. The range of weights are scaled to $[-1,1]$ and mapped to the graycolor map. $w = -1$ is assigned to black, $w = 0$ to gray, and $w = 1$ is assigned to white color.

of handwritten digits and their ability to discover patterns in high dimensional data are compared. These experiments were run one time and recorded. The encoding weights $\mathbf{W}^{(1)}$, also known as receptive fields or filters as in the case of image data, are reshaped, scaled, centered in a 28×28 pixel box, and visualized. The filters learned by L_1/L_2-NCSAE are compared with that learned by its counterparts, NCSAE and SAE. It can be easily observed from the results in Fig. 4 that L_1/L_2-NCSAE learned receptive fields that are more sparse and localized than those of SAE, DpAE, and NCSAE. It is remarked that the black pixels in both SAE and DpAE features are results of the negative weights whose values and numbers are reduced in NCSAE with non-negativity constraints, which are further reduced by imposing an additional L_1 penalty term in L_1/L_2-NCSAE as shown in the histograms located on the right side of the figure. In the case of L_1/L_2-NCSAE, tiny strokes and dots that constitute the basic part of handwritten digits are unearthed compared to SAE, DpAE, and NCSAE. Most of the features learned by SAE are major parts of the digits or the blurred version of the digits, which are obviously not as sparse as those learned by L_1/L_2-NCSAE. Also, the features learned by DpAE are fuzzy compared to those of

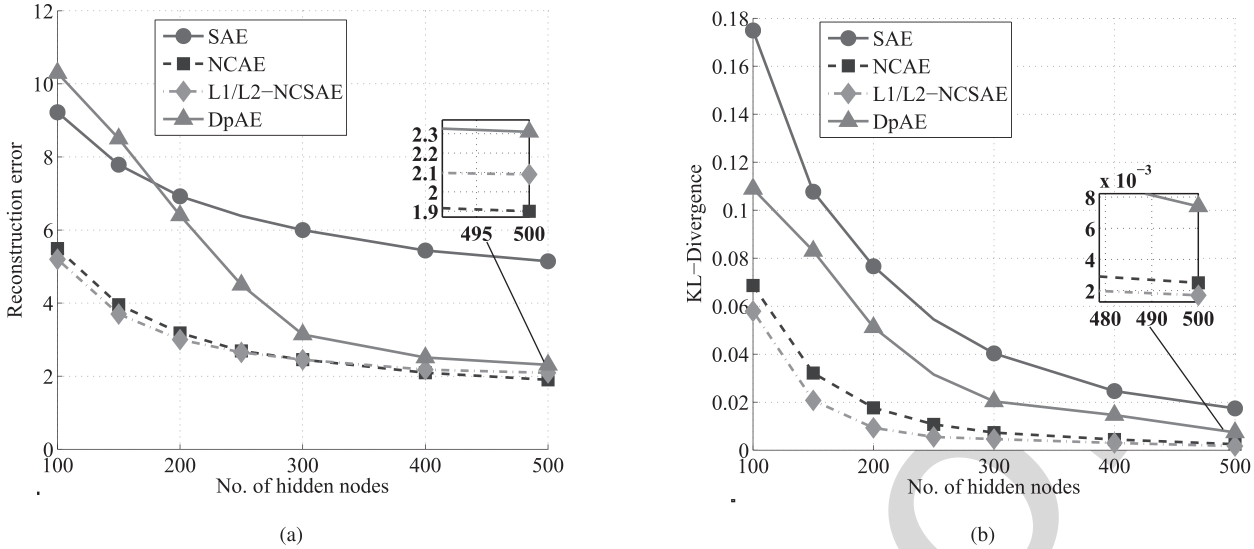


Fig. 5. (a) Reconstruction error and (b) sparsity of hidden units measured by KL-divergence using MNIST train data set with $p = 0.05$.

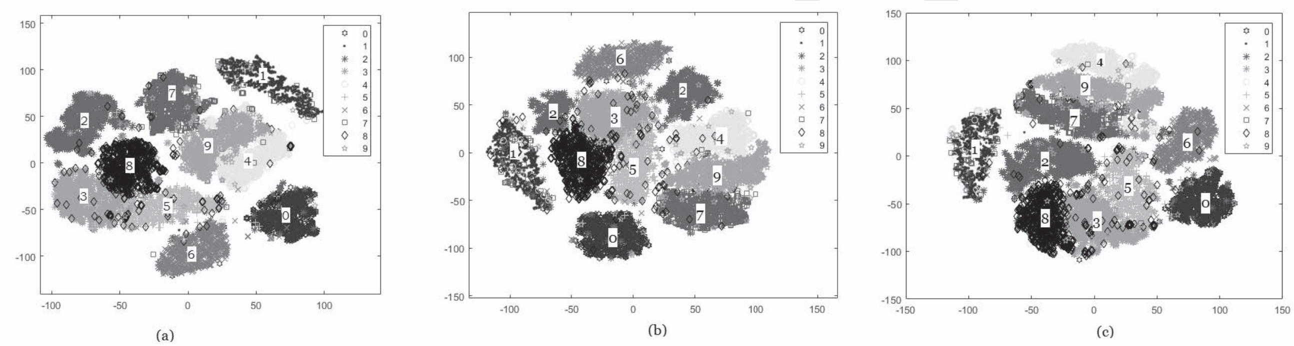
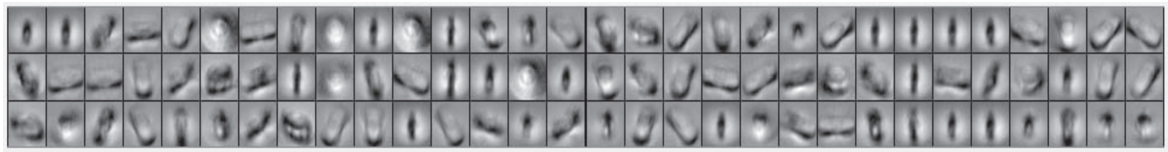


Fig. 6. t-SNE projection [31] of 196D representations of MNIST handwritten digits using (a) DpAE, (b) NCSAE, and (c) L_1/L_2-NCSAE.

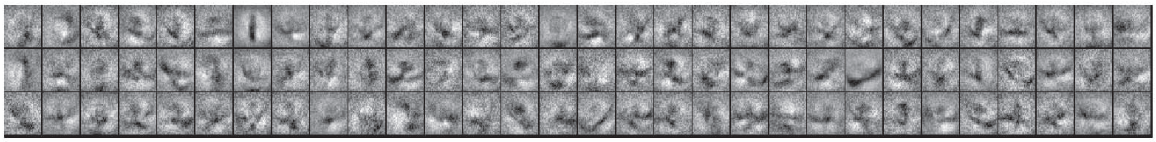
L_1/L_2-NCSAE that are sparse and distinct. Therefore, the achieved sparsity in the encoding can be traced to the ability of L_1 and L_2 regularizations in enforcing high degree of weights' non-negativity in the network.

Likewise in Fig. 5(a), L_1/L_2-NCSAE with other AEs are compared in terms of reconstruction error, while varying the number of hidden nodes. As expected, it can be observed that L_1/L_2-NCSAE yields a reasonably lower reconstruction error on the MNIST training set compared to SAE, DpAE, and NCSAE. Although, a close scrutiny of the result also reveals that the reconstruction error of L_1/L_2-NCSAE deteriorates compared to NCSAE when the hidden size grows beyond 400. However, on the average, L_1/L_2-NCSAE reconstructs better than other AEs considered. It can also be observed that DpAE with 50% dropout has high reconstruction error when the hidden layer size is relatively small (100 or less). This is because the few neurons left are unable to capture the dynamics in the data, which subsequently results in underfitting the data. However, the reconstruction error improves as the hidden layer size is increased. Lower reconstruction error in the case of L_1/L_2-NCSAE and NCSAE is an indication that non-negativity constraint facilitates the learning of parts of digits that are essential for reconstructing the digits. In addition, the KL-divergence sparsity measure reveals that L_1/L_2-NCSAE has more sparse hidden activations than SAE, DpAE, and NCSAE for different hidden layer size, as shown in Fig. 5(b). Again, averaging over all the training examples, L_1/L_2-NCSAE yields less activated hidden neurons compared to its counterparts. Also, using t-distributed stochastic neighbor embedding (t-SNE) to project the 196-D representation of MNIST handwritten digits to 2-D space, the distribution of features encoded by 196 encoding filters of DpAE, NCSAE, and L_1/L_2-NCSAE are, respectively, visualized in Fig. 6(a)–(c). A careful look at Fig. 6(a) reveals that digits "4" and "9" are overlapping in DpAE, and this will inevitably increase the chance of misclassifying these two digits. It can also be observed in Fig. 6(b) corresponding to NCSAE that digit "2" is projected with two different landmarks. In sum, the manifolds of digits with L_1/L_2-NCSAE are more separable than its counterpart as shown in Fig. 6(c), aiding the classifier to map out the separating boundaries among the digits more easily.

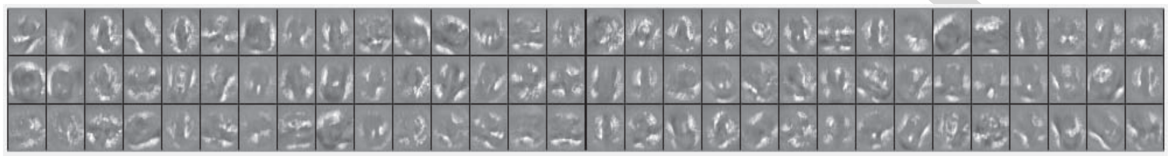
In the second experiment, SAE, NCSAE, L_1/L_2-NCSAE, and DpAE with 200 hidden nodes were trained using the NORB normalized-uniform data set. The NORB normalized-uniform data set, which is the second data set, contains



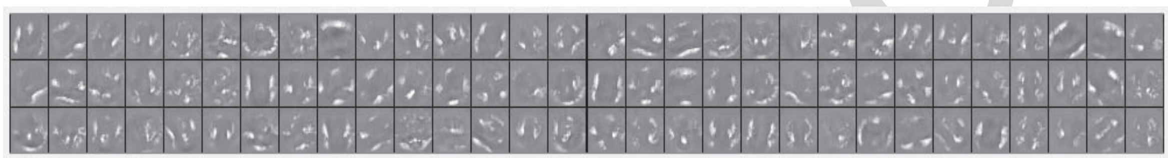
(a) SAE



(b) DpAE

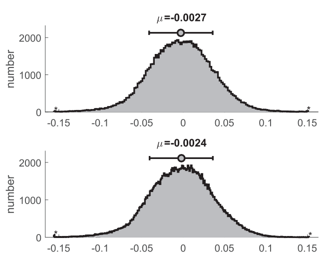


(c) NCAE

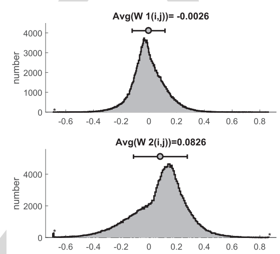


(d) L_1/L_2-NCSAE

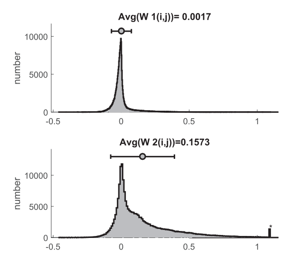
Fig. 7. Weights of randomly selected 90 out of 200 receptive filters of (a) SAE, (b) DpAE, (c) NCSAE, and (d) L_1/L_2-NCSAE using NORB data set. The range of weights are scaled to $[-1,1]$ and mapped to the graycolor map. $w <= -1$ is assigned to black, $w = 0$ to gray, and $w >= 1$ is assigned to white color.



(a)



(b)



(c)

Fig. 8. Distribution of 200 encoding ($\mathbf{W}^{(1)}$) and decoding filters ($\mathbf{W}^{(2)}$) weights learned from NORB data set using (a) DpAE, (b) NCSAE, and (c) L_1/L_2-NCSAE.

24 300 training images and 24 300 test images of 50 toys from five generic categories: four-legged animals, human figures, airplanes, trucks, and cars. The training and testing sets consist of five instances of each category. Each image consists of two channels, each of size 96×96 pixels. The inner 64×64 pixels of one of the channels cropped out and resized using bicubic interpolation to 32×32 pixels that form a vector with 1024 entries as the input. Randomly selected weights of 90 out of 200 neurons are plotted in Fig. 7. It can be seen that L_1/L_2-NCSAE learned more sparse features compared to features learned by all other AEs considered. The receptive fields learned by L_1/L_2-NCSAE captured the real actual edges of the toys while the edges captured by NCSAE are fuzzy, and those learned by DpAE and SAE are holistic. As shown in the weight distribution depicted in Fig. 8, L_1/L_2-NCSAE has both its encoding and decoding weights centered around zero with most of its weights positive when compared with those of DpAE and NCSAE that have weights distributed almost even on both sides of the origin.

B. Unsupervised Semantic Feature Learning From Text Data

In this experiment, DpAE, NCSAE, and L_1/L_2-NCSAE are evaluated and compared based on their ability to extract semantic features from text data, and how they are able to discover the underlined structure in text data. For this purpose, the Reuters-21578 text categorization data set with 200 features is utilized to train all the three types of AEs with 20 hidden nodes. A subset of 500 examples belonging to categories "grain," "crude," and "money-fx" was extracted from the test set. The experiments were run three times, averaged, and recorded. In Fig. 9, the 20-dimensional representations of the Reuters data subset using DpAE, NCSAE, and L_1/L_2-NCSAE are visualized. It can be observed that L_1/L_2-NCSAE is able to disentangle the documents into three distinct categories with more linear manifolds than NCSAE. In addition, L_1/L_2-NCSAE is able to group documents that are closer in the semantic space into the same categories than DpAE that finds it difficult to group the documents into any distinct categories with less overlap.

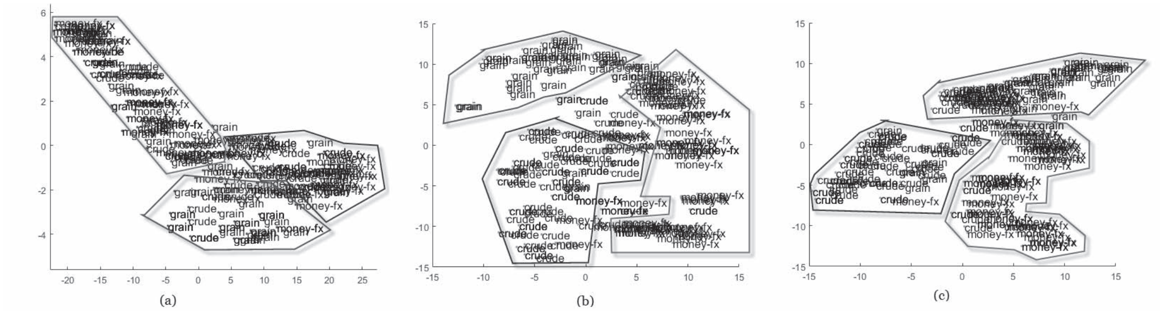


Fig. 9. Visualizing 20D representations of a subset of Reuters documents data using (a) DpAE, (b) NCSAE, and (c) L_1/L_2-NCSAE.

TABLE I
CLASSIFICATION ACCURACY ON MNIST AND NORB DATA SET

Dataset		*Before* fine-tuning		*After* fine-tuning		
		Mean ($\pm$ SD)	p-value	Mean ($\pm$ SD)	p-value	# Epochs
MNIST	SAE	0.735 $\pm$ 0.015	<0.001	**0.977** $\pm$ 0.0007	<0.001	400
	NCAE	0.844 ($\pm$0.0085)	0.0018	0.974 ($\pm$0.0012)	0.812	126
	NNSAE	0.702 ($\pm$0.027)	<0.0001	0.970 ($\pm$0.001)	<0.0001	400
	L_1/L_2-NCSAE	**0.847** ($\pm$0.0077)	-	0.974 ($\pm$0.0087)	-	84
	DAE (50% input dropout)	0.551 ($\pm$0.011)	<0.0001	0.972 ($\pm$0.0021)	0.034	400
	DpAE (50% hidden dropout)	0.172 ($\pm$0.0021)	<0.0001	0.964 ($\pm$0.0017)	<0.0001	400
	AAE	-	-	0.912 ($\pm$0.0016)	<0.0001	1000
NORB	SAE	0.562 $\pm$ 0.0245	<0.0001	0.814 $\pm$ 0.0099	0.041	400
	NCAE	**0.696** ($\pm$0.021)	0.406	**0.817** ($\pm$0.0095)	0.001	305
	NNSAE	0.208 ($\pm$0.025)	<0.0001	0.738 ($\pm$ 0.012)	<0.001	400
	L_1/L_2-NCSAE	0.695 ($\pm$0.0084)	-	0.812 ($\pm$0.0001)	-	196
	DAE (50% input dropout)	0.461 ($\pm$0.0019)	<0.0001	0.807 ($\pm$0.0015)	0.0103	400
	DpAE (50% hidden dropout)	0.491 ($\pm$0.0013)	<0.0001	0.815 ($\pm$0.0038)	<0.0001	400
	AAE	-	-	0.791 ($\pm$0.041)	<0.0001	1000

C. Supervised Learning

In the last set of experiments, a deep network was constructed using two stacked L_1/L_2-NCSAE and a softmax layer for classification to test if the enhanced ability of the network to shatter data into parts and lead to improved classification. Eventually, the entire deep network is fine-tuned to improve the accuracy of the classification. In this set of experiments, the performance of pretraining a deep network with L_1/L_2-NCSAE is compared with those pretrained with recent AE architectures. The MNIST and NORB data sets were utilized, and every run of the experiments is repeated ten times and averaged to combat the effect of random initialization. The classification accuracy of the deep network pretrained with NNSAE [18], DpAE [19], DAE [32], AAE [22], NCSAE, and L_1/L_2-NCSAE using MNIST and NORB data, respectively, are detailed in Table I. The network architectures are 784-196-20-10 and 1024-200-20-5 for MNIST and NORB data set, respectively. It is remarked that for training of AAE with two layers of 196 hidden units in the encoder, decoder, discriminator, and other hyperparameters tuned as described in [22], the accuracy was 83.67%. The AAE reported in Table I used encoder, decoder, and discriminator each with two layers of 1000 hidden units and trained for 1000 epochs. The classification accuracy and speed of convergence are the figures of merit used to benchmark L_1/L_2-NCSAE with other AEs.

It is observed from the result that L_1/L_2-NCSAE-based deep network gives an improved accuracy before fine-tuning compared to methods such as NNSAE, NCSAE, and DpAE. However, the performance in terms of classification accuracy after fine-tuning is very competitive. In fact, it can be inferred from the p-value of the experiments conducted on MNIST and NORB in Table I that there is no significant difference in the accuracy after fine-tuning between NCSAE and L_1/L_2-NCSAE even though most of the weights in L_1/L_2-NCSAE are non-negativity constrained. Therefore, it is remarked that even though the interpretability of the deep network has been fostered by constraining most of the weights to be non-negative and sparse, nothing significant has been lost in terms of accuracy. In addition, network trained with L_1/L_2-NCSAE was also observed to converge faster than its counterparts. On the other hand, NNSAE also has non-negative weights but with deterioration in accuracy, which is more conspicuous especially before the fine-tuning stage. The improved accuracy before fine-tuning in L_1/L_2-NCSAE-based network can be traced to its ability to decompose data more into distinguishable parts. Although the performance of L_1/L_2-NCSAE after fine-tuning is similar to those of DAE

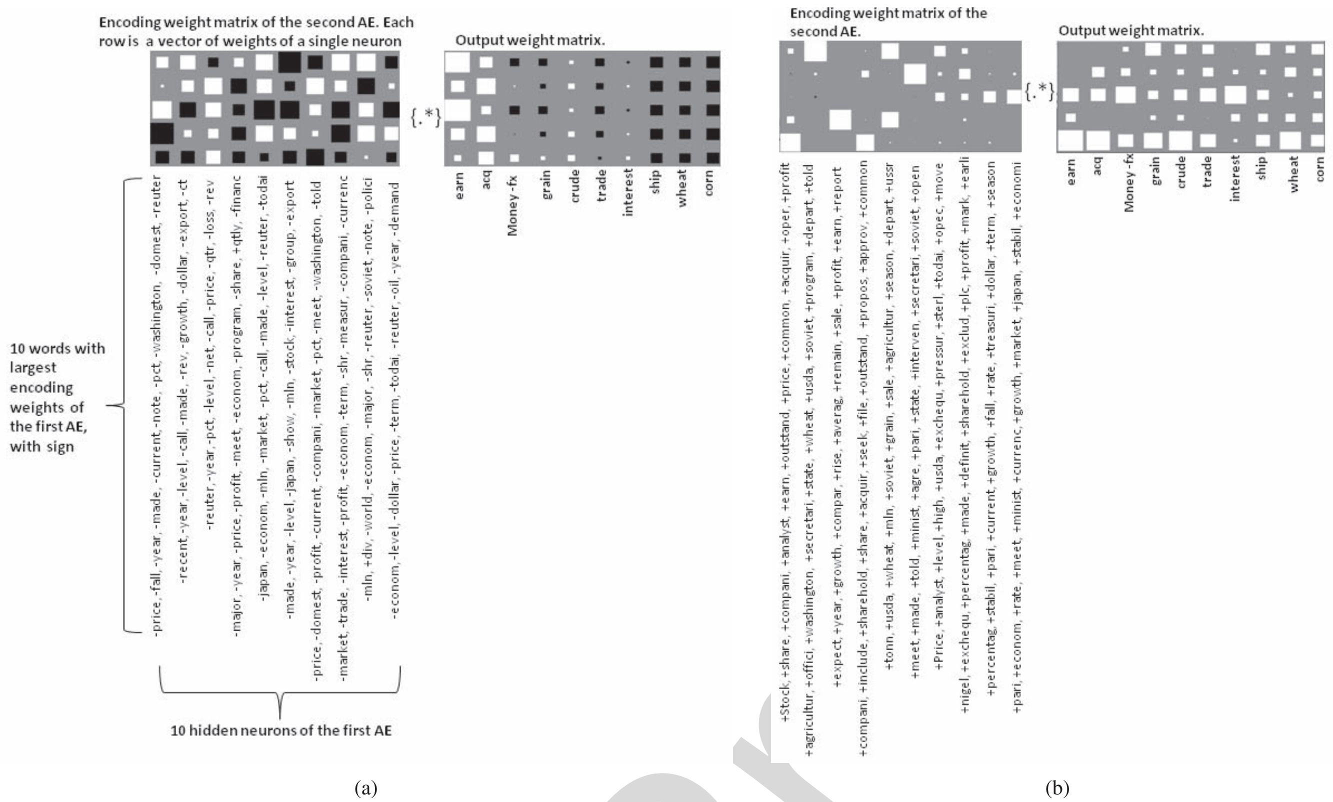


Fig. 10. Deep network trained on Reuters-21578 data using (a) DpAE and (b) L_1/L_2-NCSAE. The area of each square is proportional to the weight's magnitude. The range of weights is scaled to [−1,1] and mapped to the graycolor map. $w = -1$ is assigned to black, $w = 0$ to gray, and $w = 1$ is assigned to white color.

and NCSAE but better than NNSAE, DpAE, and AAE, L_1/L_2-NCSAE constrains most of the weights to be non-negative and sparse to foster transparency than for other AEs. However, DpAE and NCSAE performed slightly more accurate than L_1/L_2-NCSAE on NORB after network fine-tuning.

In light of constructing an interpretable deep network, an L_1/L_2-NCSAE pretrained deep network with 10 hidden neurons in the first AE layer, 5 hidden neurons in the second AE, and 10 output neurons (one for each category) in the softmax layer was constructed. It was trained on Reuters data and compared with that pretrained using DpAE. The interpretation of the encoding layer of the first AE is provided by listing words associated with 10 strongest weights, and the interpretation of the encoding layer of the second AE is portrayed as images characterized by both the magnitude and sign of the weights. Compared to the AE with weights of both signs shown in Fig. 10(a), Fig. 10(b) allows for much better insight into the categorization of the topics.

Topic *earn* in the output weight matrix resonates with the fifth hidden neuron most, lesser with the third, and somewhat with the fourth. This resonance can happen only when the fifth hidden neuron reacts to input by words of columns 1 and 4, and in addition, to a lesser degree, when the third hidden neuron reacts to input by words of the third column of words. So, in tandem, the dominant columns 1 and 4 and then also 3 are sets of words that trigger the category *earn*.

Analysis of the term words for the topic *acq* leads to a similar conclusion. This topic also resonates with the two dominant hidden neurons 5 and 3 and somewhat with neuron 2. These neurons 5 and 3 are driven again by the columns of words 1, 4, and 3. The difference between the categories is now that to a lesser degree, the category *acq* is influenced by the sixth column of words. An interesting point is in contribution of the third column of words. The column connects only to the fourth hidden neuron but weights from this neuron in the output layer are smaller and hence less significant than for any other of the five neurons (or rows) of the output weight matrix. Hence, this column is of least relevance in the topical categorization.

D. Experiment Running Times

The training time for networks with and without the non-negativity constraints was compared. The constrained network converges faster and requires a lesser number of training epochs. In addition, the unconstrained network requires more time per epoch than the constrained one. The running time experiments were performed using full MNIST benchmark data set on Intel(r) Core i7-6700 CPU at 3.40 Ghz and a 64 GB of RAM running a 64-b Windows 10 Enterprise edition. The software implementation has been with MATLAB 2015b with batch gradient descent method, and LBFGS in minFunc [33] is used to minimize the objective function. The usage times for constrained and unconstrained networks were also compared. We consider the usage time in milliseconds (ms), and as the time elapsed in ms, a fully trained deep network requires to classify all the test samples. The unconstrained network took 48 ms per epoch in the training phase, while the constrained counterpart took 46 ms. Also, the unconstrained network required 59.9-ms usage time, whereas the network with non-negative weights took 55 ms. From the above observations, it is remarked that the non-negativity constraint simplifies the resulting network.

V. Conclusion

This paper addresses the concept and properties of special regularization of DL AE that takes advantage of non-negative encodings and at the same time of special regularization. It has been shown that using both L_1 and L_2 to penalize the negative weights, most of them are forced to be non-negative and sparse, and hence, the network interpretability is enhanced. In fact, it is also observed that most of the weights in the Softmax layer become non-negative and sparse. In sum, it has been observed that encouraging non-negativity in NCSAE-based deep architecture forces the layers to learn part-based representation of their input and leads to a comparable classification accuracy before fine-tuning the entire deep network and not-so-significant accuracy deterioration after fine-tuning. It has also been shown on select examples that concurrent L_1 and L_2 regularizations improve the network interpretability. The performance of the proposed method was compared in terms of sparsity, reconstruction error, and classification accuracy with the conventional SAE and NCSAE, and we utilized MNIST handwritten digits, Reuters documents, and the NORB data set to illustrate the proposed concepts.

References

[1] Y. Bengio and Y. LeCun, "Scaling learning algorithms towards AI," *Large-Scale Kernel Mach.*, vol. 34, no. 5, pp. 1–41, 2007.

[2] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.

[3] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[4] L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning," *APSIPA Trans. Signal Inf. Process.*, vol. 3, p. e2, 2014. AQ:3

[5] S. Bengio, L. Deng, H. Larochelle, H. Lee, and R. Salakhutdinov, "Guest editors' introduction: Special section on learning deep architectures," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1795–1797, Aug. 2013.

[6] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in Neural Information Processing Systems*, vol. 19. Cambridge, MA, USA: MIT Press, 2007, p. 153.

[7] B. Ayinde and J. Zurada, "Clustering of receptive fields in autoencoders," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2016, pp. 1310–1317.

[8] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, Oct. 1999.

[9] J. Chorowski and J. M. Zurada, "Learning understandable neural networks with nonnegative weight constraints," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 1, pp. 62–69, Jan. 2015. AQ:4

[10] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607–609, 1996.

[11] E. Hosseini-Asl, J. M. Zurada, and O. Nasraoui, "Deep learning of part-based representation of data using sparse autoencoders with nonnegativity constraints," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 12, pp. 2486–2498, Dec. 2016.

[12] M. Ranzato, Y.-L. Boureau, and Y. LeCun, "Sparse feature learning for deep belief networks," in *Advances in Neural Information Processing Systems*, vol. 20. Red Hook, NY, USA: Curran Associates, 2007, pp. 1185–1192.

[13] M. Ishikawa, "Structural learning with forgetting," *Neural Netw.*, vol. 9, no. 3, pp. 509–521, Apr. 1996.

[14] P. L. Bartlett, "The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network," *IEEE Trans. Inf. Theory*, vol. 44, no. 2, pp. 525–536, Mar. 1998.

[15] G. Gnecco and M. Sanguineti, "Regularization techniques and suboptimal solutions to optimization problems in learning from data," *Neural Comput.*, vol. 22, no. 3, pp. 793–829, 2010.

[16] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Advances in Neural Information Processing Systems*, vol. 4. San Mateo, CA, USA: Morgan Kaufmann, 1995, pp. 950–957.

[17] O. E. Ogundijo, A. Elmas, and X. Wang, "Reverse engineering gene regulatory networks from measurement with missing values," *EURASIP J. Bioinform. Syst. Biol.*, vol. 2017, no. 1, p. 2, 2017. AQ:5

[18] A. Lemme, R. F. Reinhart, and J. J. Steil, "Online learning and generalization of parts-based image representations by non-negative sparse autoencoders," *Neural Netw.*, vol. 33, pp. 194–203, Sep. 2012.

[19] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. (Jul. 2012). "Improving neural networks by preventing co-adaptation of feature detectors." [Online]. Available: https://arxiv.org/abs/1207.0580

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[21] D. P. Kingma and M. Welling. (Dec. 2013). "Auto-encoding variational Bayes." [Online]. Available: https://arxiv.org/abs/1312.6114

[22] A. Makhzani, J. Shlens, N. Jaitly, and I. Goodfellow. (Nov. 2015). "Adversarial autoencoders." [Online]. Available: https://arxiv.org/abs/1511.05644

[23] Y. Burda, R. Grosse, and R. Salakhutdinov. (Sep. 2015). "Importance weighted autoencoders." [Online]. Available: https://arxiv.org/abs/1509.00519

[24] B. O. Ayinde, E. Hosseini-Asl, and J. M. Zurada, "Visualizing and understanding nonnegativity constrained sparse autoencoder in deep learning," in *Artificial in Intelligence and Soft Computing* (Lecture Notes in Computer Science), vol. 9692, L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. Zadeh, and J. Zurada, Eds. Springer, 2016, pp. 3–14. AQ:6

[25] S. J. Wright and J. Nocedal, *Numerical Optimization*, vol. 2. New York, NY, USA: Springer, 1999.

[26] T. D. Nguyen, T. Tran, D. Phung, and S. Venkatesh, "Learning parts-based representations with nonnegative restricted Boltzmann machine," in *Proc. Asian Conf. Mach. Learn.*, 2013, pp. 133–148.

[27] A. Ng. Sparse Autoencoder. *CS294A Lecture Notes*. Stanford University, (2011). [Online]. Available: https://web.stanford.edu/class/cs294a/sparseAutoencoder_2011new.pdf

[28] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

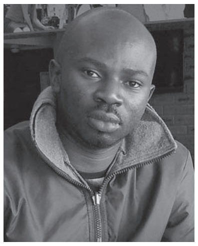
[29] Y. LeCun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2. Jul. 2004, pp. II-97–II-104.

[30] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*, vol. 1. Boston, MA, USA: Addison-Wesley, 2006.

[31] L. V. der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 11, pp. 2579–2605, 2008.

[32] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 1096–1103.

[33] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM J. Sci. Comput.*, vol. 16, no. 5, pp. 1190–1208, 1995.



Babajide O. Ayinde (S'09) received the M.Sc. degree in engineering systems and control from the King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia. He is currently pursuing the Ph.D. degree with the University of Louisville, Louisville, KY, USA.

His current research interests include unsupervised feature learning and deep learning techniques and applications.

Mr. Ayinde was a recipient of the University of Louisville fellowship.

Jacek M. Zurada (M'82–SM'83–F'96–LF'14) received the Ph.D. degree from the Gdansk Institute of Technology, Gdansk, Poland.

He currently serves as a Professor of electrical and computer engineering with the University of Louisville, Louisville, KY, USA. He has authored or co-authored several books and over 380 papers in computational intelligence, neural networks, machine learning, logic rule extraction, and bioinformatics, and delivered over 100 presentations throughout the world.

Dr. Zurada ha s been a Board Member of the IEEE CIS and IJCNN. AQ:7 He was a recipient of the 2013 Joe Desch Innovation Award, the 2015 Distinguished Service Award, and five honorary professorships. He served as the IEEE V-President and the Technical Activities Board (TAB) Chair in 2014. He was the Chair of the IEEE TAB Periodicals Committee and the TAB Periodicals Review and Advisory Committee. From 2004 to 2005, he was the President of the IEEE Computational Intelligence Society. He was the Editor-in-Chief of the IEEE TRANSACTIONS ON NEURAL NETWORKS (1997–2003) and an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, NEURAL NETWORKS and the PROCEEDINGS OF THE IEEE. He is an Associate Editor of Neurocomputing and several other journals.

AUTHOR QUERIES

AUTHOR PLEASE ANSWER ALL QUERIES

PLEASE NOTE: We cannot accept new source files as corrections for your paper. If possible, please annotate the PDF proof we have sent you with your corrections and upload it via the Author Gateway. Alternatively, you may send us your corrections in list format. You may also upload revised graphics via the Author Gateway.

AQ:1 = Please note that the “Non-negativity Constrained Sparse Autoencoder” and “model predictive control” have been provided as the expansions for MPC. Please confirm which one to follow.
AQ:2 = The abbreviation NCAE has been changed to NCSAE throughout the document. Since the same expansion “Non-negativity Constrained Sparse Autoencoder” is given for both the abbreviation. Please confirm.
AQ:3 = Please provide the issue no. or month for ref. [4].
AQ:4 = Please note that there were discrepancies between the accepted pdf [TNNLS-2016-P-6572.pdf] and the [TNNLS-2016-P-6572.R2.tex] in reference section. We have followed [TNNLS-2016-P-6572.R2.tex].
AQ:5 = Please confirm the volume no. for ref. [17].
AQ:6 = Please confirm the volume no. for ref. [24]. also provide the publisher location.
AQ:7 = Please confirm whether the edits made in the sentence “Dr. Zurada has been….” are OK.