Sparsely Connected Autoencoder

Kavya Gupta
IIIT Delhi
New Delhi, India
kavya@iiitd.ac.in

Angshul Majumdar
IIIT Delhi
New Delhi, India
angshul@iiitd.ac.in

***Abstract*— This work proposes to learn autoencoders with sparse connections. Prior studies on autoencoders enforced sparsity on the neuronal activity; these are different from our proposed approach – we learn sparse connections. Sparsity in connections helps in learning (and keeping) the important relations while trimming the irrelevant ones. We have tested the performance of our proposed method on two tasks – classification and denoising. For classification we have compared against stacked autneencoders, contractive autoencoders, deep belief network, sparse deep neural network and optimal brain damage neural network; the denoising performance was compared against denoising autoencoder and sparse (activity) autoencoder. In both the tasks our proposed method yields superior results.**

***Index Terms*— autoencoder, classification, sparsity, denoising**

I. Introduction

There is a plethora of work in sparse neural networks. Broadly it can be segregated into i) sparse activity and ii) sparse connectivity. Sparsity can arise in two contexts. The sparse activity property means that only a small fraction of neurons is active at any time. The sparse connectivity property means that each neuron is connected to only a limited number of other neurons.

Since its onset, neural networks have been claimed to mimic the human brain. For a certain activity / task only a portion of the brain (neurons) are active. The whole brain is never used. In fact, it is a widely circulated myth that we use only 10% of our brain; the myth is untrue. But it is well known that only a certain portion of the brain is active for a certain task; i.e. given the whole brain only a sparse set of neurons are actually active (for the given task). Therefore, if indeed the neural network is an approximate representative of the brain, we would expect to have sparse connections. This aspect has been captured by LeCun's work on 'optimal brain damage' [1]. He devised a technique to trim connections of a neural network (hence the name brain damage) without degrading its performance.

In recent times the idea has been revisited. In [2, 3] sparsity is enforced both on the activity and on the connectivity. In [4] sparsity on activity was promoted for rectifier neural network; it was used in [5] to improve the performance on phone recognition. Sparsity in connections was exploited in [6] for the problem of speech recognition. In [7] and [8] sparsity in connections is enforced on convolutional neural network and recurrent neural network respectively. In most of these studies the common observation is that introduction of sparsity leads to a slight dip in performance but reduces the complexity of the network significantly.

In this work we are specifically interested in autoencoders. Although there has been a lot of work on sparse connectivity and sparse activity on neural networks, all prior studies in sparse autoencoders enforced sparsity on the activity; there is no prior study that promoted sparsity in connections. This is the first work to do so. In [9] sparsity was introduced in terms of firing neurons. If the neurons are of high value (near about 1), it is allowed to be fired, the rest are not. In [10], only the top K high valued neurons are fired; in [11] only the neurons beyond a predefined threshold were fired. In [12], a comparison of different sparsity promoting terms (on activites) were compared; these were the KL divergence [9] and variations of l_1-norm [11]. It was shown in [13] that by combining the output of several such sparse autoencoders (trained as in [9]), one is able to improve performance of several image recognition tasks.

We compare our proposed sparsely connected autoencoder with several variants of autoencoders (for classification and denoising) and deep belief network (for classification). We find that our proposed technique yields better results compared to existing techniques.

Although most of the readers of this paper will be abreast with literature on neural networks in general and autoencoders in particular, we provide a brief review of autoencoders for the sake of completeness in the next section. In section 4, our proposed method is described in detail. The experimental evaluation is reported in section 5. The conclusions of this work is discussed in section 6.

II. Background

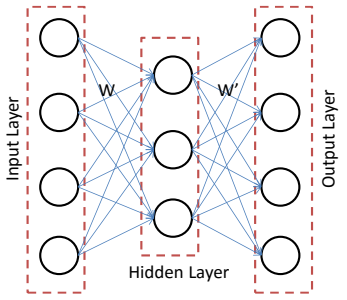

Fig. 1. Single Layer Autoencoder

An auto encoder (as seen in Fig. 1) consists of two parts – the encoder maps the input to a latent space, and the decoder maps the latent representation to the data. For a given input vector (including the bias term) x, the latent space is expressed as:

$$h = Wx \tag{1}$$

Here the rows of W are the link weights from all the input nodes to the corresponding latent node. The mapping can be linear, but in most cases it is non-linear (sigmoid, tanh etc.):

$$h = \phi(Wx) \tag{2}$$

The decoder portion reverse maps the latent features to the data space.

$$x = W'\phi(Wx) \tag{3}$$

Since the data space is assumed to be the space of real numbers, there is no sigmoidal function here.

During training, the problem is to learn the encoding and decoding weights – W and W'. This is achieved by minimizing the Euclidean cost:

$$\arg\min_{W,W'} \left\| X - W'\phi(WX) \right\|_F^2 \tag{4}$$

Here $X = [x_1 \,|\, ... \,|\, x_N]$ consists all the training sampled stacked as columns. The problem (4) is clearly non-convex. However, it is solved easily by gradient descent techniques since the sigmoid function is smooth and continuously differentiable.

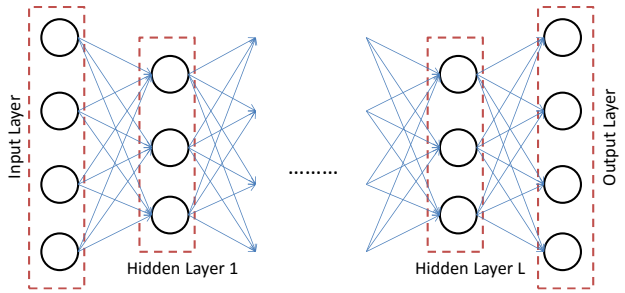

Fig. 2. Stacked Autoencoder

There are several extensions to the basic autoencoder architecture. Stacked / Deep autoencoders [9] have multiple hidden layers (see Fig. 2). The corresponding cost function is expressed as follows:

$$\arg\min_{W_1 \ldots W_{L-1}, W'_1 \ldots W'_L} \left\| X - g \circ \right. \tag{5}$$

where $g = W_1'\phi\left(W_2'\ldots W_L'\left(f(X)\right)\right)$ and

$f = \phi\left(W_{L-1}\phi\left(W_{L-2}\ldots\phi(W_1 X)\right)\right)$.

Solving the complete problem (5) is computationally challenging. The weights are usually learned in a greedy fashion – one layer at a time [14].

Stacked denoising autoencoders (SDAE) [15] are a variant of the basic autoencoder where the input consists of noisy samples and the output consists of clean samples. Here the encoder and decoder are learnt to denoise noisy input samples. The learned features appear to be more robust when learnt by SDAE.

In a recent work a marginalized denoising autoencoder was proposed [16], which does not have any intermediate nodes but learns the mapping from the input to the output. This formulation is convex (unlike regular autoencoders); the trick here is to marginalize over all possible noisy samples so that the dataset need not be augmented like SDAE. Such an autoencoder was used for domain adaptation.

Another variation for the basic autoencoder is to regularize it, i.e.

$$\arg\min_{(W)s} \left\| X - g \circ \right. \qquad + R(W, X) \tag{6}$$

The regularization can be a simple Tikhonov regularization – however that is not used in practice. It can be a sparsity promoting term [9]-[11] or a weight decay term (Frobenius norm of the Jacobian) as used in the contractive autoencoder [17]. The regularization term is usually chosen so that they are differentiable and hence minimized using gradient descent techniques.

III. Proposed SparseConnect Autoencoder

Autoencoders usually have a non-linear activation function. However, in [18] it was shown that an autoencoder usually operates in the linear region. Therefore in this work, we will use a linear activation function. This allows us to derive a more efficient algorithm which is faster than its non-linear counterparts. We also show (experimentally) that the linear autoencoder yields better results than its non-linear counterpart. The basic formulation of an autoencoder with linear activation function is given by:

$$\arg\min_{W',W} \left\| X - W'WX \right\|_F^2 \tag{7}$$

The basic autoencoder is prone to overfitting; especially when the number of training samples is limited. Denoising autoencoders use a stochastic regularization technique. However, given the Euclidean cost function of the autoencoder a more direct way to regularize it would be incorporate penalty terms to the basic formulation. For example, a contractive autoencoder with linear activation function would lead to the following formulation:

$$\arg\min_{W',W} \left\| X - W'WX \right\|_F^2 + \lambda\left(\left\|W\right\|_F^2 + \left\|W'\right\|_F^2\right) \tag{8}$$

The Frobenius norm on the weights regularizes the network to have small values. The regularization prevents overfitting of the network.

We propose to regularize the autoencoder such that it has sparse connections both at the encoder and the decoder. The idea of trimming irrelevant connections in neural networks is not new; it was first proposed back in 1990 in the form of optimal brain damage [1]. In recent times, learning sparse structures in neural networks has gained momentum [2-8]. However, to the best of our knowledge, there is no work that learns autonencoders with sparse connections. Prior studies in sparse autoencoder [9-12] concentrate on sparse activities; not on sparse connections. In this respect ours is the first work to propose sparsely connected autoencoders.

Just as a human brain does not require all its neurons for a specific task, we postulate that an autoencoder does not need to utilize all its connections either. The issue of maintaining important connections without over-fitting is taken care of, if we have sparse weights. The portions which are not useful for representation are pruned and only the important connections in the network are maintained. Such a sparse connection is easily achieved from the following proposed formulation,

$$\arg\min_{W',W} \|X - W'WX\|_F^2 + \lambda\left(\|W\|_{1/0} + \|W'\|_{1/0}\right) \quad (9)$$

We have abused the notation a bit, the subscript 1/0 denotes either an l_1-norm or an l_0-norm and is defined on the vectorial representation of the weights. The l_1-norm is convex, and has been widely used in recent times by Compressed Sensing [19], [20]. But the l_0-norm does not ideally yield sparse weights, the l_0-norm does. Unfortunately l_0-norm minimization is an NP hard problem [21]. However there are approximate techniques to solve such l_0-minimization problems.

Autoencoders with non-linear activation function are solved using gradient descent techniques. Such techniques cannot be directly applicable for our proposed formulation. This is because the l_1/l_0-norm penalties are not differentiable everywhere. In this work we follow a Majorization Minimization approach to solve the said problem.

A. Majorization Minimization

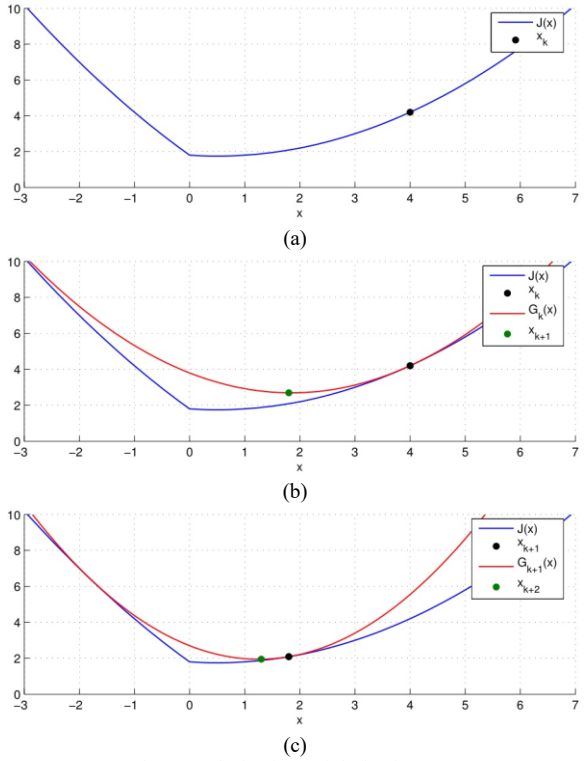

Fig. 3. Majorization-Minimization [22]

Fig. 3 shows the geometrical interpretation behind the Majorization-Minimization (MM) approach. The figure depicts the solution path for a simple scalar problem but essentially captures the MM idea.

Let, $J(x)$ be the function to be minimized. Start with an initial point (at $k=0$) x_k (Fig. 3a). A smooth function $G_k(x)$ is constructed through x_k which has a higher value than $J(x)$ for all values of x apart from x_k, at which the values are the same. This is the Majorization step. The function $G_k(x)$ is constructed such that it is smooth and easy to minimize. At each step, minimize $G_k(x)$ to obtain the next iterate x_{k+1} (Fig. 3b). A new $G_{k+1}(x)$ is constructed through x_{k+1} which is now minimized to obtain the next iterate x_{k+2} (Fig. 3c). As can be seen, the solution at every iteration gets closer to the actual solution.

For convenience we express the problem (9) in a slightly different manner in terms of transposes –

$$\arg\min_{H} \left\|X^T - X^T H^T\right\|_F^2 + R(H) \quad (10)$$

Here $H=W'W$ and $R(H)$ denotes the penalty.

Only the least squares part need to be majorized; the penalty terms are not affected.

$$J(H) = \left\|X^T - X^T H^T\right\|_F^2 + R(H) \quad (11)$$

For this minimization problem, $G_k(x)$, the majorizer of $J(x)$ is chosen to be,

$$G_k(H) = \left\|X^T - X^T H^T\right\|_F^2 + (H^T - H_k^T)^T (aI - XX^T)(H^T - H_k^T) \quad (12)$$

where a is the maximum eigenvalue of the matrix XX^T and I is the identity.

One can check that at $H=H_k$ the expression $G_k(H)$ reduces to $J(H)$. At all other points it is larger than $J(H)$; the value of 'a' assures that the second term is positive definite.

$$G_k(H) = \left\|X^T - X^T H^T\right\|_F^2 + \left(H^T - H_k^T\right)^T \left(aI - XX^T\right)\left(H - H_k^T\right) + R(H)$$

$$\Rightarrow XX^T - 2XX^T H^T + HXX^T H^T + \left(H^T - H_k^T\right)^T \left(aI - XX^T\right)\left(H - H_k^T\right) + R(H)$$

$$\Rightarrow XX^T + H_k(aI - XX^T)H_k^T - 2\left(XX^T + H_k\left(aI - XX^T\right)\right)H_k^T + aHH^T + R(H)$$

$$\Rightarrow a(-2BH^T - HH^T) + C + R(H)$$

where $B^T = H_k^T + \frac{1}{a}X(X^T - X^T H_k)$,

$$C = XX^T + H_k(aI - XX^T)H_k^T$$

Using the identity $\|A - D\|_2^2 = A^T A - 2A^T D + D^T D$, one can write,

$$G_k(H) = a\|B - H\|_2^2 - aB^T B + C + R(H)$$

$$= a\|B - H\|_2^2 + R(H) + K$$

where K consists of terms independent of x.

Therefore, minimizing $G_k(x)$ is the same as minimizing the following,

$$G_k'(H) = \|B - H\|_2^2 + \frac{1}{a}R(H) \quad (13)$$

where $B^T = H_k^T + \frac{1}{a} X(X^T - X^T H_k)$.

This update is known as the Landweber iteration.

B. l_1-norm penalty

First we derive the algorithm for solving the l_1-norm minimization problem.

$$\arg\min_{W',W} \|B - W'W\|_F^2 + \frac{\lambda}{a}\left(\|W\|_1 + \|W'\|_1\right) \quad (14)$$

This is a bilinear problem and we propose to solve it alternately, i.e. we fix W' and solve W and then solve W' assuming W is fixed. These two steps are done in every iteration.

$$W_{k+1} = \arg\min_{W} \|B - W_k'W\|_F^2 + \frac{\lambda}{a}\|W\|_1 \quad (15a)$$

$$W'_{k+1} = \arg\min_{W'} \|B - W'W_{k+1}\|_F^2 + \frac{\lambda}{a}\|W'\|_1 \quad (15b)$$

In both cases, the problem remains the same – that of a least squares minimization with l_1-norm penalty.

Let us take the first problem and work out the solution for it; the solution for the other problem will remain the same.

To solve (15a) we invoke the majorization approach once again. Therefore (15a) can be expressed as,

$$\arg\min_{W} \|P - W\|_F^2 + \frac{\lambda}{\alpha a}\|W\|_1 \quad (16)$$

where $P = W_k + \frac{1}{\alpha} W_k^T(B - W_k'W_k)$, α is the maximum eigenvalue of $W^T'W'$

The above function (16) is actually de-coupled, i.e.

$$\|P - W\|_F^2 + \lambda\|W\|_1 = \sum_i (P_i - W_i)^2 + \frac{\lambda}{\alpha a}|W_i| \quad (17)$$

Therefore, (17) can be minimized term by term, i.e.

$$\frac{\partial\left(\|P - W\|_F^2 + \lambda\|W\|_1\right)}{\partial W_i} = 2P_i - 2W_i + \frac{\lambda}{a\alpha}\,signum(W_i) \quad (18)$$

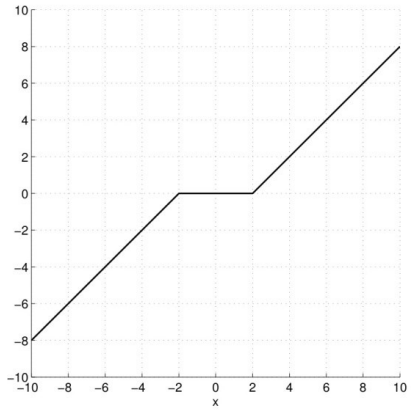

Fig. 4. Soft Threshold Rule (with τ=2)

Setting the partial derivatives to zero and solving for W gives the graph shown in Fig. 4 with threshold $\frac{\lambda}{2a\alpha}$. That is, the minimizer of (16) is obtained by applying the soft-threshold rule to P with threshold $\frac{\lambda}{2a\alpha}$. The soft-threshold rule is the non-linear function defined as,

$$soft(x,\tau) = \begin{cases} x+\tau & x < -\tau \\ 0 & |x| = \tau \\ x-\tau & x > \tau \end{cases} \quad (19)$$

Or more compactly,

$$W = signum(P)\max(0, |P| - \frac{\lambda}{2a\alpha}) \quad (20)$$

This concludes the steps for solving (15a); the steps for (15b) are exactly the same. In a compact fashion, the algorithm for solving the l_1-norm penalty problem is given as:

Initialization:

$H = \arg\min_H \|X^T - X^T H\|_F^2$

$H = USV^T$

$W_0' = US$ and $W_0 = V^T$

In every iteration 'k'

Compute $B^T = H_k^T + \frac{1}{a} X(X^T - X^T H_k)$

Update $W_{k+1} = \arg\min_W \|B - W_k'W\|_F^2 + \frac{\lambda}{a}\|W\|_1$

$P = W_k + \frac{1}{\alpha} W_k^T(B - W_k'W_k)$

$W_{k+1} = signum(P)\max(0, |P| - \frac{\lambda}{2a\alpha})$

Similarly update $W'_{k+1} = \arg\min_{W'} \|B - W'W_{k+1}\|_F^2 + \frac{\lambda}{a}\|W'\|_1$

Our initialization is deterministic, hence the results are results are repeatable – there is no variation between trials as long as other parameters remain same.

C. l_0-norm penalty

The l_1-norm penalty is basically a shrinkage function defined by the soft thresholding. It cannot get an exactly sparse solution, it only shrinks the values of unwanted weights. To get a sparse solution in every iteration, one needs to solve the l_0-norm minimization problem. This is an NP hard problem but has approximate solutions. The more common practice is to solve (40) using a greedy approach based on Orthogonal Matching Pursuit [23], [24]. However, these are not efficient for solving large scale problems. To solve a k-sparse problem, k iterations are required. A better approach to solve (21) is based on Iterative Hard Thresholding [25].

$$\arg\min_{W',W} \|B - W'W\|_F^2 + \frac{\lambda}{a}\left(\|W\|_0 + \|W'\|_0\right) \quad (21)$$

We solve it via alternating minimization.

$$W_{k+1} = \arg\min_{W} \|B - W_k'W\|_F^2 + \frac{\lambda}{a}\|W\|_0 \quad (22a)$$

$$W'_{k+1} = \arg\min_{W'} \|B - W'W_{k+1}\|_F^2 + \frac{\lambda}{a}\|W'\|_0 \quad (22b)$$

As before, both the problems remain the same. We only derive the algorithm to solve (22a). To solve it, we invoke the majorization approach once again. Therefore (22a) can be expressed as,

$$\arg\min_{W} \|P - W\|_F^2 + \frac{\lambda}{\alpha a}\|W\|_0 \quad (23)$$

where $P = W_k + \frac{1}{\alpha} W_k^T{}'\left(B - W_k{}' W_k\right)$

This is a decoupled problem and can be expressed as,

$$\|P - W\|_F^2 + \frac{\lambda}{\alpha a}\|W\|_0 = (P_1 - W_1)^2 + \frac{\lambda}{a\alpha}|W_1|^0 + \ldots + (P_n - W_n)^2 + \frac{\lambda}{a\alpha}|W_n|^0 \quad (24)$$

We can process (24) element-wise.

To derive the minimum, two cases need to be considered: case 1 – $W_i = 0$ and case 2 – $W_i \neq 0$. The element-wise cost is 0 in the first case. For the second case, the minimum is reached when $W_i = P_i$. Comparing the cost on both cases,

0 if $W_i = 0$

$-(W_i)^2 + \frac{\lambda}{a\alpha}$ if $W_i = P_i$

This suggests the following updates rule,

$$W_i = \begin{cases} P_i \text{ when } |P_i| > \lambda / 2a\alpha \\ 0 \text{ when } |P_i| \leq \lambda / 2a\alpha \end{cases}$$

This is popularly known as hard thresholding and is represented as:

$$W_{k+1} = HardTh\left(P, \frac{\lambda}{2a\alpha}\right) \quad (25)$$

This leads to an algorithm somewhat similar to the previous one. It is succinctly represented below.

Initialization:

$H = \arg\min_{H} \left\|X^T - X^T H\right\|_F^2$

$H = USV^T$

$W_0{}' = US$ and $W_0 = V^T$

In every iteration 'k'

Compute $B^T = H_k^T + \frac{1}{a} X(X^T - X^T H_k)$

Update $W_{k+1} = \arg\min_{W} \|B - W_k{}' W\|_F^2 + \frac{\lambda}{a}\|W\|_0$

$P = W_k + \frac{1}{\alpha} W_k^T{}'\left(B - W_k{}' W_k\right)$

$W_{k+1} = HardTh\left(P, \frac{\lambda}{2a\alpha}\right)$

Similarly update $W'_{k+1} = \arg\min_{W'} \|B - W'{}' W_{k+1}\|_F^2 + \frac{\lambda}{a}\|W'\|_0$

IV. Experimental Evaluation

The MNIST digit classification task is composed of 28x28 images of the 10 handwritten digits. There are 60,000 training images with 10,000 test images in this benchmark. The images are scaled to [0,1] and we do not perform any other pre-processing.

Experiments are also carried out on the more challenging variations of the MNIST dataset. These have been used in [11] among others and were introduced as benchmark deep learning datasets. All these datasets have 12,000 training (we do not need validation) and 50,000 test samples. The size of the image as before is 28x28 and the number of classes are 10.

Dataset	Description
basic	Smaller subset of MNIST.
basic-rot	Smaller subset of MNIST with random rotations.
bg-rand	Smaller subset of MNIST with uniformly distributed random noise background.
bg-img	Smaller subset of MNIST with random image background.
bg-img-rot	Smaller subset of MNIST digits with random background image and rotation.

We have also evaluated on the problem of classifying documents into their corresponding newsgroup topic. We have used a version of the 20-newsgroup dataset [26] for which the training and test sets contain documents collected at different times, a setting that is more reflective of a practical application. The training set consists of 11,269 samples and the test set contains 7,505 examples. We have used 5000 most frequent words for the binary input features. We follow the same protocol as outlined in [27].

Our third dataset is the GTZAN music genre dataset [28, 29]. The dataset contains 10000 three-second audio clips, equally distributed among 10 musical genres: blues, classical, country, disco, hip-hop, pop, jazz, metal, reggae and rock. Each example in the set is represented by 592 Mel-Phon Coefficient (MPC) features. These are a simplified formulation of the Mel-frequency Cepstral Coefficients (MFCCs) that are shown to yield better classification performance. Since there is no predefined standard split and fewer examples, we have used 10-fold cross validation (procedure mentioned in [15]), where each fold consisted of 9000 (we do not require validation examples unlike [8]) training examples and 1000 test examples.

A. Linear vs Non-linear

Most studies in neural networks employ a non-linear activation function. We proposed linear activation owing to the ease of solution. We will show that, atleast for the benchmark datasets used in these experiments, the simple linear (Identity) activation function yields better classification accuracy than their non-linear (sigmoid) counterpart.

The linear autoencoder weights are initialized by solving the least squares problem, $\min_{Q} \|X - QX\|_F^2$ and setting W as the top (number of nodes) right singular vectors of Q. For the non-linear autoconder we use the Hinton's implementation [30].

The autoencoder architectures remains same otherwise; both (linear and non-linear) are three layer architectures with 392-196-98 hidden nodes. The representation from the deepest layer is used for classification. We employ two non-parametric classifiers – KNN (K=1) and Sparse Representation based Classification (SRC) [31]. We want to test the representation / feature extraction capability of the linear and non-linear autoencoders; this is best done using simple non-parametric classifiers. Parametric classifiers like NN and SVM may be fine tuned to yield better results, but in such a case it is difficult to gauge if the improvement in results is owing to the feature extraction or owing to the fine tuning.

TABLE I. LINEAR VS NON-LINEAR ACTIVATION

Dataset	KNN		SRC	
	Linear	*Non-linear*	*Linear*	*Non-linear*
MNIST	**97.33**	96.11	**98.33**	97.29
basic	**95.25**	94.86	**96.91**	96.43
basic-rot	**84.83**	80.71	**90.04**	84.29
bg-img	**77.16**	70.97	**84.14**	76.94
bg-rand	**86.42**	81.11	**91.03**	85.49
bg-img-rot	**52.21**	44.6	**62.46**	50.96
20-newsgroup	**71.78**	70.48	**72.56**	70.49
GTZAN	**84.08**	83.31	**84.39**	83.37

The results show that the linear one always yields better results. The improvement is small when the number of training samples are larger but for the more challenging datasets, the linear autoencoder yields improve by a large margin.

B. Classification Performance

We compare our results with the stacked autoencoder (SAE), Contractive Autoencoder (CAE) and Deep Belief Network (DBN) [28]. The SAE and CAE uses linear activation. As before, the representation from the deepest layer is used as features. For our sparsely connected autoencoder λ=0.01 is used. In each of the tables, the best results are shown in bold.

TABLE II. KNN (K=1) RESULTS

Dataset	SAE	CAE	DBN	Proposed l_0-norm	Proposed l_1-norm
MNIST	**97.33**	82.83	97.05	97.21	95.91
basic	95.25	79.92	95.37	**95.39**	92.49
basic-rot	84.83	58.56	84.71	**85.14**	81.01
bg-rand	86.42	65.61	86.36	**86.88**	68.87
bg-img	77.16	58.51	77.16	**77.22**	79.84
bg-img-rot	**52.21**	27.10	50.47	51.80	38.91
20-newsgroup	70.48	71.08	70.09	**71.74**	71.23
GTZAN	83.31	82.67	80.99	**83.89**	83.08

TABLE III. SRC RESULTS

Dataset	SAE	CAE	DBN	Proposed l_0-norm	Proposed l_1-norm
MNIST	98.33	87.19	88.43	**98.42**	97.16
basic	96.91	85.03	87.49	**97.03**	95.43
basic-rot	90.04	68.63	79.47	**90.19**	87.76
bg-rand	91.03	72.25	79.67	**91.69**	76.17
bg-img	84.14	65.68	75.09	**85.11**	85.84
bg-img-rot	62.46	34.01	49.68	**62.61**	47.75
20-newsgroup	70.49	71.08	71.02	**72.38**	71.97
GTZAN	83.37	82.70	81.21	**84.72**	83.89

TABLE IV. SVM RESULTS

Dataset	SAE	CAE	DBN	Proposed l_0-norm	Proposed l_1-norm
MNIST	98.50	88.74	98.53	**98.60**	97.70
basic	96.96	86.61	97.07	**97.12**	95.70
basic-rot	**89.43**	72.54	89.05	89.22	87.07
bg-rand	91.28	75.20	89.59	**91.73**	87.04
bg-img	84.86	68.76	85.46	**85.35**	78.01
bg-img-rot	60.53	40.97	58.25	**60.91**	46.30
20-newsgroup	70.28	70.95	71.82	**73.98**	73.54
GTZAN	82.76	82.09	82.69	**83.52**	83.04

The results show that our proposed method yields better results than SAE and DBN on an all the datasets (except for the larger MNIST with KNN). Under fair comparison (keeping the classifiers to be same and non-parametric) one can say that our method yields better representation than other deep learning techniques like SAE, CAE and DBN.

Next (Table IV) we compare the results with stacked denoising autoencoder (SDAE), deep belief network (DBN), sparse deep neural network (SDNN) [33] and optimal brain damage (OBD) [1]. We repeat the results from l_0-norm sparsely connected autoencoder with SRC (since these are the best results we obtained). SDAE and DBN uses a fine tuning with a neural network classifier in the final stage. SDNN is a contemporary sparse deep classifier and OBD is a classical work with a shallow architecture. The results show that our method yields results which are at par with SDAE and DBN and are better than sparse neural networks.

TABLE V. COMPARATIVE RESULTS

Dataset	SDAE	DBN	SDNN	OBD	Proposed
MNIST	98.72	**98.76**	98.57	97.99	98.42
basic	**97.16**	96.89	96.69	95.41	97.03
basic-rot	**90.47**	89.70	89.58	87.89	90.19
bg-rand	89.70	**93.27**	90.21	88.27	91.69
bg-img	83.32	83.69	82.96	80.65	**85.11**
bg-img-rot	56.24	52.61	51.63	50.19	**62.61**
20-newsgroup	70.93	**72.40**	69.74	65.44	72.38
GTZAN	83.98	81.62	80.31	77.80	**84.72**

We have compared the training times of different autoencoders and DBN. The results are shown in Table V. Both are proposed methods are significantly faster than the others. The computational cost per iteration is higher for us, but the algorithms converge faster. The results are only shown on the large MNIST dataset and the MNIST basic.

TABLE VI. TRAINING TIME IN MINUTES

Dataset →	**MNIST**	**basic**
DBN	78	21
Stacked Autoencoder (non-linear)	251	53
Stacked Autoencoder (linear)	111	35
Contractive Autoencoder (linear)	98	24
Proposed l_1-norm	4	1
Proposed l_0-norm	50	6

The configuration of the machine running these experiments is:

RAM- 24 GB
OS- Red Hat Enterprise Linux Server release 7.0 (Maipo)
CPU - Intel(R) Xeon(R) CPU E5-2430 0 @ 2.20GHz ,there are two cpus of 6 cores each
Simulation on Matlab R2014a.

C. Denoising Results

Autoencoders have been used previously for image denoising. In [11] it was shown that autoencoder with sparse features leads to good denoising results. They showed results for Gaussian and impulse denoising. It is not optimal to remove impulse noise with autoencoders; this is because impulse noise is sparse. Since we are formulating an autoencoder with l_2-norm data fidelity, we can optimally remove Gaussian noise only; this is the problem we address in this work.

For comparison, we use the standard metrics for image quality assessment - PSNR (Peak Signal to Noise Ratio) and SSIM (Structural Similarity Index) [34]. We compare our approach (SparseConnect) with the sparse autoencoder [11] and stacked denoising autoencoder (SDAE).

We use single layer autoencoders for image denoising. The number of nodes in the hidden layer is kept to be 512. The value of λ is 0.001.

We carried out experiments on the grayscale CIFAR-10 dataset. The CIFAR-10 dataset (Fig. 10) is composed of 10 classes of natural images with 50,000 training examples in total, 5,000 per class. Each image is of size 32x32. For these experiments the color images have been converted to greyscale. Zero mean Gaussian noise was added to these images. The noisy images served as the input to the autoencoders and the clean images were the output. For testing, the noisy test images were as inputs and the image obtained at the output was compared with the clean image to test the denoising performance.

The results are shown in the following table. The PSNR and the SSIM values shown here are the means over 10,000 test images.

TABLE VII. DENOISING RESULTS

Noise Variance, PSNR and SSIM for Noisy Image	SDAE	Sparse Autoencoder	Proposed SparseConnect (l_1-norm)	Proposed SparseConnect (l_0-norm)
Variance=0.01, PSNR=19.9691 SSIM=0.5691	PSNR=21.95 SSIM=0.6315	PSNR=22.94 SSIM=0.6803	PSNR=23.90 SSIM=0.7238	PSNR=**26.03** SSIM=**0.8052**
Variance=0.04 PSNR= 14.0155 SSIM=0.3256	PSNR=21.66 SSIM=0.6167	PSNR=22.63 SSIM=0.6667	PSNR=23.53 SSIM=0.7027	PSNR=**25.68** SSIM=**0.7928**
Variance=0.09 PSNR= 10.4936 SSIM=0.2011	PSNR=21.30 SSIM=0.5973	PSNR=22.25 SSIM=0.6478	PSNR=23.01 SSIM=0.6725	PSNR=**25.27** SSIM=**0.7779**

The improvement is significant. Usually in image denoising literature a PSNR improvement of 0.5 dB to 1 dB is considered to be good. In this case the improvement is near about 3dB compared to the sparse denoising autoencoder. Also the improvement in SSIM is around 0.1 – this is huge improvement. For visual evaluation some sample images are shown in Fig. 5.

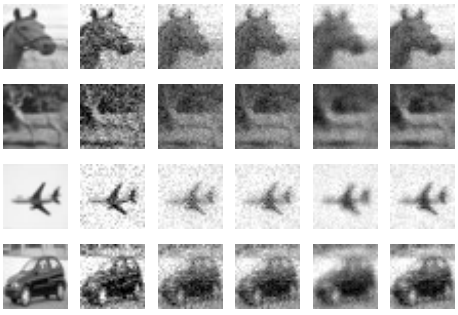

Fig. 5. Left to Right – Original test image, noisy image, SDAE, Sparse Autoencoder, Proposed l_1-norm and Proposed l_0-norm.

The denoising results may not be at par with the state-of-the-art like BM3D or KSVD, but are better than competing autoencoder based techniques. The proposed SparseConnect autoencoder gets the best denoising results, balancing noise and sharpness.

V. CONCLUSION

This work proposes the concept of sparse connections in autoencoders. Although there are several studies on sparsely connected neural networks, there is no prior study on sparsely connected autoencoder. This is the first work in that respect. All prior studies in sparse autoencoders concentrate on the problem of sparse activity.

The motivation is drawn from the success of DropOut and DropConnect neural networks where over-fitting was prevented by randomly switching off some activations or connections during training. Instead of using such a stochastic regularization technique, our proposed method deterministically 'learns' the sparse connections. It keeps the relevant connections and prunes the unimportant ones. This

is achieved by introducing sparsity promoting regularization penalties on the autoencoder weights.

Experiments were carried out for two benchmark autoencoder tasks – classification and denoising. For classification, comparison is made with the SAE, CAE, DBN, SDAE, sparse deep neural network (SDNN) and optimal brain damage (OBD). Under fair comparison (when non-parametric classifiers are used) our method outperforms others. Even with fine-tuned neural network architectures, our proposed approach yields better results than SDNN and OBD and performs at par with densely connected networks like SDAE and DBN. For denoising, we compared against the denoising autoencoder and the sparse autoencoder [3]. Even for this task our proposed approach yields considerably better results.

REFERENCES

[1] Y. LeCun, "Optimal Brain Damage", Advances in Neural Information Processing Systems, 1990.

[2] M. Thom and G. Palm, "Sparse Activity and Sparse Connectivity in Supervised Learning", Journal of Machine Learning Research, Vol. 14, pp. 1091-1143, 2013.

[3] V. Gripon, "Sparse Neural Networks With Large Learning Diversity", IEEE Transactions on Neural Networks and Learning Systems, Vol. 22 (7), pp. 1087-1096, 2011.

[4] X. Glorot, A. Bordes and Y. Bengio, "Deep Sparse Rectifer Neural Networks", AISTATS 2011.

[5] L. Toth, "Phone Recognition with Deep Sparse Rectifier Neural Networks", ICASSP 2013.

[6] D. Yu, F. Seide, G. Li and L. Deng, "Exploiting Sparseness in Deep Neural Networks for Large Vocabulary Speech Recognition", ICASSP 2012.

[7] B. Liu, M. Wang, H. Foroosh, M. Tappen and M. Penksy, "Sparse Convolutional Neural Networks", CVPR 2015.

[8] H. Awano, S. Nishide, H. Arie, J. Tani, T. Takahashi, H. G. Okuno and T. Ogata, "Use of a Sparse Structure to Improve Learning Performance of Recurrent Neural Networks", Neural Information Processing, pp. 323-331, Lecture Notes in Computer Science.

[9] Andrew Ng, "Sparse Autoencoder", CS294A Lecture notes, vol. 72, 2011

[10] A. Makhani and B. Frey, "K-sparse Autoencoder", ICLR 2014.

[11] K. H. Cho, "Simple Sparsification Improves Sparse Denoising Autoencoders in Denoising Highly Noisy Images", ICML 2013.

[12] N. Jiang, W. Rong, B. Peng, Y. Nie and Z. Xiong" An empirical analysis of different sparse penalties for autoencoder in unsupervised feature learning", IJCNN 2015.

[13] Y. Lu, L. Zhang, B. Wang and J. Yang, "Feature ensemble learning based on sparse autoencoders for image classification", IJCNN 2014.

[14] Y. Bengio, "Learning deep architectures for AI", Foundations and Trends in Machine Learning, 2 (1),1-127. 2009

[15] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio and P. -A. Manzagol, "Stacked denoising autoen coders: Learning useful representations in a deep network with a local denoising criterion", Journal of Machine Learning Research, Vol. 11, 3371-3408, 2010.

[16] M. Chen, K. Weinberger, F. Sha, Y. Bengio, "Marginalized Denoising Autoencoders for Nonlinear Representation", ICML 2014.

[17] S Rifai, P Vincent, X Muller, X Glorot, Y Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction", ICML 2011.

[18] H. M. Abbas, "Analysis and pruning of nonlinear auto-association networks", IEE Proceedings on Vision, Image and Signal Processing, Vol. 151 (1), pp. 44-50, 2004.

[19] D. Donoho, Compressed sensing. (IEEE Trans. on Information Theory, 52(4), pp. 1289 - 1306, April 2006)

[20] E. Candès and T. Tao, Near optimal signal recovery from random projections: Universal encoding strategies? (IEEE Trans. on Information Theory, 52(12), pp. 5406 - 5425, December 2006)

[21] B. K. Natarajan, "Sparse approximate solutions to linear systems", SIAM Journal on Computing, 24(1995), 227-234

[22] Sparse Signal Restoration: cnx.org/content/m32168/latest/

[23] Y. C. Pati, R. Rezaiifar and P. S. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition", Asilomar Conference on Signals, Systems and Computers, pp.40-44, 1993.

[24] J. A. Tropp and A. C. Gilbert, "Signal Recovery From Random Measurements Via Orthogonal Matching Pursuit", IEEE Transactions on Information Theory, Vol. 53 (12), pp.4655-4666, 2007.

[25] Thomas Blumensath and Mike E. Davies, "Iterative Thresholding for Sparse Approximations", Journal of Fourier Analysis Applications, Vol. 14 (5), 629-654, 2008.

[26] http://people.csail.mit.edu/jrennie/20Newsgroups/20news-bydate-matlab.tgz

[27] H. Larochelle and Y. Bengio, "Classification using Discriminative Restricted Boltzmann Machines", International Conference on Machine Learning, 2008.

[28] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals", IEEE Transactions on Audio and Speech Processing 2002.

[29] http://marsyasweb.appspot.com/download/data_sets/

[30] www.cs.toronto.edu/~hinton/MatlabForSciencePaper.html

[31] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry and Y. Ma, "Robust face recognition via sparse representation", IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(2), 210-227, 2009.

[32] http://ceit.aut.ac.ir/~keyvanrad/DeeBNet%20Toolbox.html

[33] http://www.mathworks.in/matlabcentral/fileexchange/42853-deep-neural-network

[34] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612, Apr. 2004.