Nonredundant Sparse Feature Extraction using Autoencoders with Receptive Fields Clustering

Email addresses: babajide.ayinde@louisville.edu (Babajide O. Ayinde), jacek.zurada@louisville.edu (Jacek M. Zurada)

Nonredundant Sparse Feature Extraction using Autoencoders with Receptive Fields Clustering

Babajide O. Ayinde[a,*], Jacek M. Zurada[b,a,**]

[a]*Electrical and Computer Engineering, University of Louisville, Louisville, KY, 40292 USA.*
[b]*Information Technology Institute, University of Social Science,Łódz 90-113, Poland*

Abstract

This paper proposes new techniques for data representation in the context of deep learning using agglomerative clustering. Existing autoencoder-based data representation techniques tend to produce a number of encoding and decoding receptive fields of layered autoencoders that are duplicative, thereby leading to extraction of similar features, thus resulting in filtering redundancy. We propose a way to address this problem and show that such redundancy can be eliminated. This yields smaller networks and produces unique receptive fields that extract distinct features. It is also shown that autoencoders with nonnegativity constraints on weights are capable of extracting fewer redundant features than conventional sparse autoencoders. The concept is illustrated using conventional sparse autoencoder and nonnegativity-constrained autoencoders with MNIST digits recognition, NORB normalized-uniform object data and Yale face dataset.

Keywords: Autoencoder, agglomerative clustering, deep learning, filter clustering, receptive fields.

1. Introduction

Feature extraction through constrained learning of receptive fields (RFs) offers special promise and has recently become one of the important tenets of deep learning (DL) (Bengio (2009); Hinton and Salakhutdinov (2006)). In deep autoencoding, autoencoder (AE) performs unsuper-

*Corresponding author
**Principal corresponding author
Email addresses: babajide.ayinde@louisville.edu (Babajide O. Ayinde), jacek.zurada@louisville.edu (Jacek M. Zurada)

vised learning to detect feature hierarchies which shatter the data and generate features. In this process each added AE layer adds one more abstract representation of inputs, in effect producing a cascade of encodings. Further, the architectural complexity and the excessive number of weights and units are often built in into the DL data representation by design and are deliberate (Bengio and LeCun (2007); Bengio (2009); Hinton and Salakhutdinov (2006); Deng (2014); Bengio et al. (2013)).

Our observations and those of other researchers (Bengio et al. (2013); LeCun et al. (2015)) indicate that over-sized DL architectures typically result in largely over-determined (or over-complete) systems. Unless special ad-hoc precautions are implemented to achieve acceptable accuracy such as de-noising, contraction of layers, and elimination of initially assumed and inherent redundancies, the generalization abilities of DL resulting methods suffer (Bengio et al. (2013)). The resulting architectures may therefore not be the most computationally efficient due to their size, computational complexity, and due to their over-representation of data. Such sub-optimal architectures sometime may learn local, or isolated features, especially with shallower architectures which have a limited capacity to combine inputs/features.

To address the over-representation of data mapping into the DL multilayer architectures, layers can be trained under specific and well-defined sets of constraints that remove a number of training limitations. The subsequent sections discuss the purpose and specific techniques of defining the constraints. The constraints criteria refer to the RFs (or simply filters) originally introduced in (Olshausen and Field (1996)).

In general, as a result of learning for minimum error of reconstruction or classification, the RFs manifest themselves as quasi basis functions that are usually sparse and of the same dimensionality as the input layer (or, in general, of dimensionality of the preceding processing layer). Experiments show that a large number of filters are similar or even duplicative, thus creating unnecessary amount of filtering redundancy. The same features are therefore extracted multiple times. In this paper, we aim at reducing the number of redundant RFs first in the offline setting, and in order to fully leverage on the proposed heuristic, their clustering is automated with the goal of reaching a predetermined number of filter clusters. It is remarked that the proposed approach requires final retraining of the AE to lower its reconstruction error. Further, we

focus on filters that sort classes of images.

Deep autoencoding attempts to combine many simple transformations into a more complex one. One of its implementations is a sparse AE which learns representation of data. It decomposes the input samples into a number of binary and/or continuous but preferably sparse concepts. The reconstruction is then achieved by adding "parts": ideally it could be an additive linear combination of few basis functions by the decoder that provides nonnegative scaling coefficients (Chorowski and Zurada (2015)).

The most closely related work to the described is the dropout technique - one of the recently introduced heuristics to sparsify the AEs to prevent overfitting. The key idea is to randomly drop units and their connections from the neural network during training (Srivastava et al. (2014)). Other related work is the k-sparse AE (Ngiam et al. (2011); Li et al. (2015); He et al. (2013)), which aims at reducing the number of filters by sorting the hidden units' activations and retaining k largest units, while setting the rest to zero. Our algorithm, on the other hand, aims at detecting the number of filter clusters according to the differentiation of filters based on their distances and the reconstruction error value. Hence, the proposed method in this paper leads to comparable reconstruction error with a reduced number of filters.

The proposed method is also applicable to AEs that extract non-negative latent features. This concept has been demonstrated using two AE architectures with non-negative weights that produce additive decompositions layer-wise (Lemme et al. (2012); Hosseini-Asl et al. (2016); Ayinde et al. (2016)). In this paper, the novel way of obtaining a near-optimal number of AE filters for sparse representation of data preliminarily discussed in (Ayinde and Zurada (2016)) is extended by: (i) reformulating the proposed algorithms for optimal performance, (ii) demonstrating the efficacy of the proposed heuristics with three different AEs, (iii) showing that the proposed heuristics improve network generalization, and lastly (iv) supporting with new experiments on new datasets. The rest of the paper is structured as follows: Section 2 gives the notation and discusses the high level feature extraction. The method of obtaining optimal number of filters is discussed in section 3. Section 4 discusses the experimental designs and presents the results. Finally, conclusions are drawn in section 5.

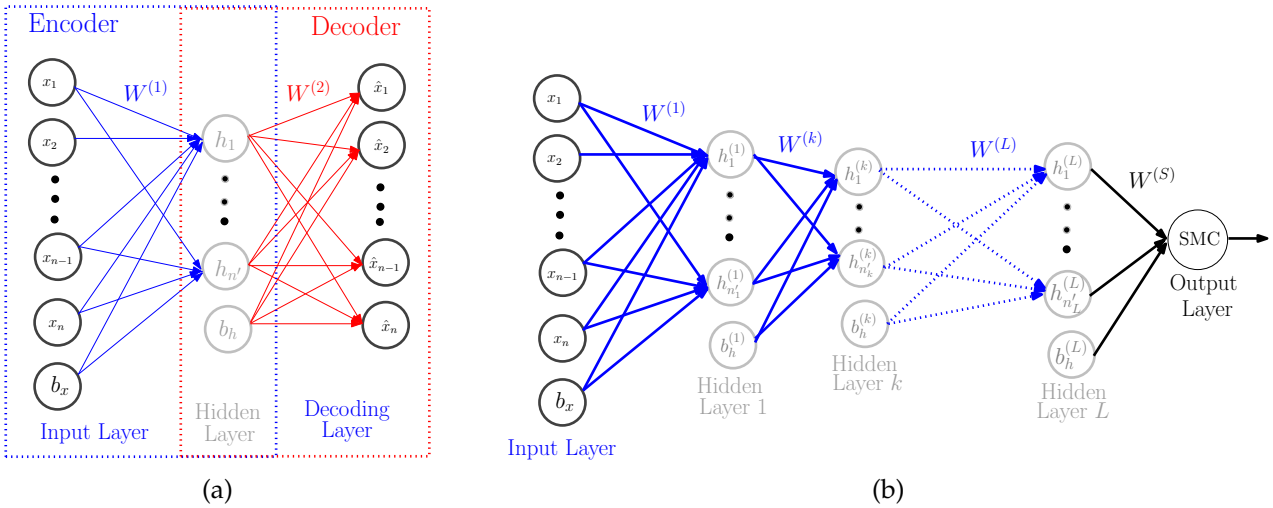


Figure 1: (a) Schematic diagram of (a) a three-layer AE or SAE and (b) a deep AE of $L+1$ layers constructed using Stacked Sparse Autoencoder (SSAE) and Softmax Classifier (SMC).

2. Deep Feature Learning using Stacked Sparse Autoencoders

Sparse autoencoder (SAE) is an unsupervised feature learning algorithm that learns sparse, high-level, structured representations of data. Generally, the input layer of SAE shown in Fig. 1a serves as an encoder that maps the input $\mathbf{x}$ to corresponding feature representation $\mathbf{h}$. The output layer can be abstracted as a decoder trained to reconstruct the original input with acceptable error. The training objective of SAE is to find parameters that minimize the norm between the input $\mathbf{x}$ and its reconstruction $\hat{\mathbf{x}}$, or the reconstruction error. The SAE learns to minimize the reconstruction error by learning encoding ($\mathbf{W}^{(1)}$), decoding ($\mathbf{W}^{(2)}$) weights, and biases ($\mathbf{b}_x, \mathbf{b}_h$) using the backpropagation algorithm. The objective function of a conventional SAE has three components (Ng (2011); Xu et al. (2014); Ngiam et al. (2011); Schulz et al. (2015)):

$$\begin{aligned}\mathcal{J}_{SAE}(\mathbf{W},\mathbf{b}) = \mathcal{J}_{AE}(\theta) + \beta \sum_{j=1}^{n'} D_{KL}\left(p \middle\| \frac{1}{m}\sum_{k=1}^{m} h_j(\mathbf{x}^{(k)})\right) \\ + \frac{\alpha}{2}\sum_{l=1}^{2}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l-1}} \left(w_{ij}^{(l)}\right)^2\end{aligned} \tag{1}$$

where $\mathbf{W} = \{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}\}$, $\mathbf{b} = \{\mathbf{b}_x, \mathbf{b}_h\}$, s_l is the number of units in layer l, $w_{ij}^{(l)}$ represents the connection between jth unit in layer $l-1$ and ith unit in layer l; m is the number of training ex-

amples, 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 $\sigma(\cdot)$ is the element-wise application of the logistic sigmoid, $\sigma(\mathbf{x}) = {}^{1}/_{(1+exp(-\mathbf{x}))}$.

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

where $||\cdot||_2$ is the Euclidean norm.

The first component of (1) is the reconstruction error over the entire data as in (2). $D_{KL}(\cdot)$ is the Kullback-Leibler (KL) divergence (Lee et al. (2007); Nair and Hinton (2009); Hinton et al. (2006); Poultney et al. (2006)) between the average activation of hidden unit j over the training set and the desired activation p; $\beta \geq 0$ controls the sparsity penalty term. Lastly, the third term of (1) is a regularizer that helps prevent overfitting by reducing the magnitude of the weights.

In order to constrain conventional SAE to extract non-negative latent features, the third term in (1) is replaced with $\mathcal{P}$ in (3) to penalize the negative weights simultaneously with both L_1 and L_2 norms yielding a L_1/L_2-Nonnegative Constrained Sparse Autoencoder (L_1/L_2-NCAE) Ayinde et al. (2016).

$$\mathcal{P} = \sum_{l=1}^{2}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l-1}}\begin{cases}\alpha_1\left\|w_{ij}^{(l)}\right\| + \frac{\alpha_2}{2}\left\|w_{ij}^{(l)}\right\|_2^2 & w_{ij}^{(l)} < 0\\ 0 & w_{ij}^{(l)} \geq 0\end{cases} \tag{3}$$

where α_1,$\alpha_2 \geq 0$. The weights are updated as below using the error backpropagation:

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

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

where $\xi > 0$ is the learning rate and the gradient of L_1/L_2-NCAE loss function is computed as in (6).

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

where $g(w_{ij})$ is a derivative of $\mathcal{P}$ (3) with respect to w_{ij} as in (7).

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

One of the advantages of constraining the weights of AEs to be nonnegative is that the average reconstruction error is reduced and the sparsity of the hidden layer activations is increased as shown in (Hosseini-Asl et al. (2016); Ayinde et al. (2016)) on variety of dataset such as MNIST handwritten digits, NORB normalized uniform object data, and the Reuters text categorization dataset. By using both L_1 and L_2 to penalize the negative weights, most of them are forced to be nonnegative and sparse, and thus enhance the network interpretability (Ayinde et al. (2016)).

In deep feature learning, basic SAEs are stacked over one another with the output of each layer feeding the input of the successive layer. A greedy layer-wise training approach is adopted to train each successive layer (Hinton et al. (2006)). The activations of the last AE are then used as the input to the Softmax layer (SMC), a supervised classifier as shown in Fig. 1b. The parameters obtained after the training yield the transformation $f : \mathcal{R}^{d_x} \rightarrow \mathcal{R}^{d_{h^{(L)}}}$ which maps input to new high level feature representation $h^{(L)}$. Since the activation of the last AE is the input to the Softmax layer, the training input of the supervised learning (classification) is given as $\{h^{(L)(k)}, y^{(k)}\}_{k=1}^{m}$ which is the pair of high level feature representation and its corresponding label. In the case of nonnegativity-constrained AEs, it must be noted that weights in the softmax layer are also nonnegativity constrained (Ayinde et al. (2016)). More details on NCAE training are in (Hosseini-Asl et al. (2016); Ayinde et al. (2016)).

3. Filter Clustering and Reduction

Training of SAEs indicate that resulting RFs are duplicative thus leading to a number of redundant RFs with the same feature to be extracted by two or more filters. In this section, two heuristics that aim at agglomerative static and dynamic clustering of filters as well as reduction of the hidden layer size are discussed. These criteria need to enforce:

1. **Static reduction of the number of redundant filters** while preserving and/or enhancing their sparsity. This reduction is analyzed and performed initially in the offline mode and

for the filters that have been pre-trained. The essence of the pre-training is to avail the filters enough iterations.

2. **Dynamic reduction and reconciliation of filters** that is undertaken concurrently with their unsupervised learning. Such clustering of filters aims at automatically choosing a good similarity threshold for RFs for a given AE architecture and detecting the number of distinct filter clusters.

The above two criteria aim at producing a reduced set of filters. The term reduction refers here to computing their smallest number according to a specific chosen measure.

3.1. Static Reduction of the Number of Redundant Filters

The objective here is to remove filters that are identical or very similar to eliminate duplicative retrieval of features. While this concept is simple, suitable similarity measures are needed to express the intra-filter distances between vectors $\mathbf{V}_s$ that define the filters. Assuming ($\mathbf{V}_s$, b_s), s=1,...n′, are weight vectors and biases, each $\mathbf{V}_s$ corresponds to the s-th row of the weight matrix $\mathbf{W}^{(1)}$ as in Fig. 1a

$$\mathbf{V}_s = \mathbf{W}_s^{(1)} \quad s = 1, ...n' \tag{8}$$

The set of n' vectors $\mathbf{V}_s$ exhibits mutual distances as follows:

$$d_{sr} = ||\mathbf{V}_s - \mathbf{V}_r|| \quad s, r = 1, ...n'; \quad s \neq r \tag{9}$$

A number of suitable agglomerative similarity testing/clustering algorithms can be applied for elimination (and possibly merger) of originally developed redundant filters. Based on a comparative review, a clustering approach from (Walter et al. (2008); Ding and He (2002)) has been adapted and reformulated for this purpose as shown in Algorithm 1.

Starting with each original filter as a potential cluster, agglomerative clustering is performed by merging the two most similar clusters C_i and C_j as long as the average similarity between their constituent filters is above a chosen cluster similarity threshold denoted as τ (Leibe et al. (2004); Manickam et al. (2000)). The similarity threshold is an hyperparameter that has to be set

Algorithm 1 Offline Feature Extraction (OFE)

function OFE(data,autoencoder(AE)):
    initialize: $\{n'_0, pretrain\_iter, max\_iter\}$
    initialize $\mathbf{W}$ and $\mathbf{b}$: $\mathbf{W}^{(\text{init})}$, $\mathbf{b}^{(\text{init})}$
    set: τ
    $\mathbf{W}$, $\mathbf{b}$ = TRAIN_AE$\big($data, n'_0, $\mathbf{W}^{(\text{init})}$, $\mathbf{b}^{(\text{init})}$, $pretrain\_iter\big)$
    $\mathcal{L}$ = COMPUTE_LOSS$\big($data, $\mathbf{W}$, $\mathbf{b}\big)$ as in (2)
    n'_{new}, $\mathbf{W}^{(agglo)}$, $\mathbf{b}^{(agglo)}$ = AGGLOMCLUSTERING$\big(\mathbf{W}, \mathbf{b}, \tau\big)$
    $\mathbf{W}^{(new)}$, $\mathbf{b}^{(new)}$ = TRAIN_AE$\big($data, n'_{new}, $\mathbf{W}^{(agglo)}$, $\mathbf{b}^{(agglo)}$, $max\_iter\big)$
end function

function AGGLOMCLUSTERING():
    Input: $\{\mathbf{W}, \mathbf{b}, \tau\}$
        Scan for cluster(s) of vectors in $\mathbf{W}$ with similarity $> \tau$
        Replace each cluster with its corresponding centroid
    Output: Number of distinct filters in $\mathbf{W}$;
        return n'_{new}, $\mathbf{W}^{(new)}$, $\mathbf{b}^{(new)}$
end function

in order to achieve optimal performance. The set of n' vectors $\mathbf{V}_s$ exhibits mutual similarities as follows:

$$similarity(C_i, C_j) = \frac{\sum_{\mathbf{V}_s \in C_i, \mathbf{V}_r \in C_j} NGC(\mathbf{V}_s, \mathbf{V}_r)}{|C_i| \times |C_j|} > \tau$$
$$i, j = 1, ...n'_{\text{new}}; \;\; s = 1, ...|C_i|, \quad r = 1, ...|C_j| \quad and \quad s \neq r \tag{10}$$

where the similarity between two filters is measured by Normalized Greyscale Correlation (NGC):

$$NGC(\mathbf{V}_s, \mathbf{V}_r) = \frac{\sum_{k=1}^{n}(p_k - \bar{p}_k)(q_k - \bar{q}_k)}{\sqrt{\sum_{k=1}^{n}(p_k - \bar{p}_k)^2 \sum_{k=1}^{n}(q_k - \bar{q}_k)^2}} \tag{11}$$

where $p_k \in \mathbf{V}_s$ and $q_k \in \mathbf{V}_r$; $\bar{p}_k = \sum_{k=1}^{n} p_k/n$ and $\bar{q}_k = \sum_{k=1}^{n} q_k/n$ and n is the size of the filter vector $\mathbf{V}_s$.

This clustering scheme guarantees that similar filters are grouped, and that the clusters stay compact (Leibe et al. (2004)). That is, the constituent filters in each cluster stay as close as possible to one another or simply put, the intra-cluster distances are as small as possible.

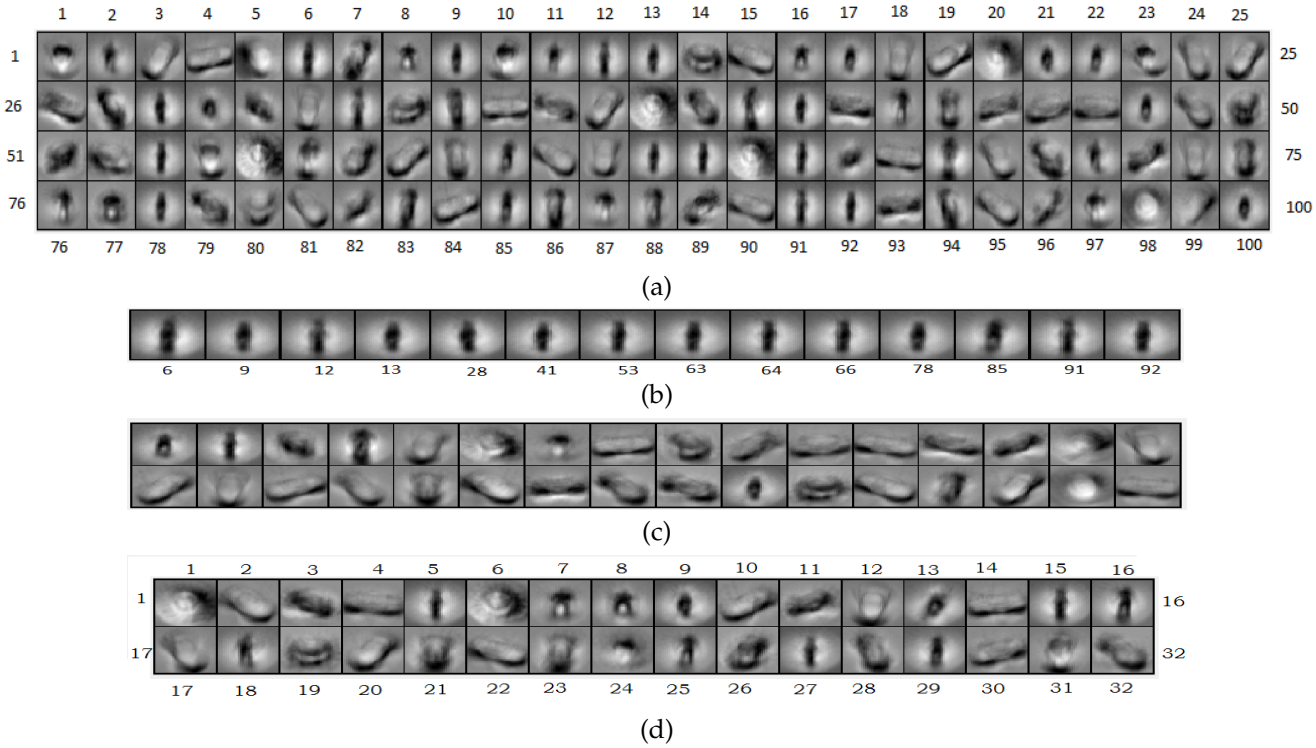


Figure 2: Filters learned from NORB data set using SAE with (a) 100 original filters (b) 14 examples of very similar filters with their corresponding indices at the bottom (c) 32 filters resulting from agglo-SAE with τ=16, and (d) 32 original filters. Black pixels indicate negative, and white pixels indicate positive weights.

3.2. Dynamic Reduction and Reconciliation of Filters

The objective here is to discover c clusters in the set of n' original filters, where $c < n'$. Further evaluation is performed on how to replace filters that are within a single cluster with this cluster prototype or centroid's representation. The algorithm has been designed to heuristically set τ to develop novel representation that expresses aggregated properties of input data across the training set. The Algorithm 2 also tries to reduce the number of hidden units based on clustering of the input weights. It trains an AE with initial number of hidden units for *pretrain_iter* epochs, which must be chosen large enough to enable formation of duplicative filters. Number of hidden units is initialized as large as practical and τ is also initialized to a small value in order to fully activate the filter reduction heuristic. The reconstruction error is evaluated using the weights and biases obtained from the pre-training stage. The weight vectors are then clustered and filters with similarity above the threshold τ are collapsed.

The resulting filters are fine-tuned for *scan_iter* epochs such that reconstruction error (2)

does not increase in comparison with the reconstruction error computed using previous τ. If the reconstruction error reduces, τ value is incremented by $\Delta\tau$ and the process is repeated. As mentioned above, the filter reduction becomes less active as τ increases, which in turn implies that reconstruction error should decrease due to increasing number of hidden units. At certain τ value and beyond, reconstruction error stops reducing which might be as a result of duplicative filters formation. Once no significant decrease in the reconstruction error associated with previous and current τ values is observed or the error starts to increase, Algorithm 2 stops and outputs the optimal number of filters and the resulting τ. The filters are then fine-tuned for *max_iter* epoch to ensure good reconstruction. It is worth mentioning that Algorithms 1 and 2 are alternative approaches and can be used independently. Also note that Algorithm 2 is an extended version of Algorithm 1 where acceptable RF cluster similarity threshold is set automatically to eliminate most redundant RFs, whereas in Algorithm 1 the hyperparameter τ is selected using trial and error. In this paper, we denote the SAE, NCAE, and L_1/L_2-NCAE trained using the approach in Algorithms 1 and 2 as agglo-SAE, agglo-NCAE and agglo-L_1/L_2-NCAE, respectively.

4. Experimental Setup

This section discusses the performance of the proposed method in redundant filter reduction and reported for three benchmark image data sets: MNIST, NORB normalized-uniform dataset and the Yale face dataset. The input to the first layer of the AE is the vector of pixel intensities. For MNIST, the size of the each example is 28×28, and this corresponds to a vector of 784 pixels intensities ($n = 784$). The vector size is 1024 for NORB and Yale face data sets. The rest of the training parameters in Table 1 have been found experimentally for hyperparameter tuning of each of the algorithm with various parameters that best minimize the cost function. In Algorithm 1, *pretrain_iter* and *max_iter* were both experimentally set to 200. Both AEs with receptive field clustering and those without clustering have very similar training time. Although clustering overhead is introduced by the proposed algorithm.

It is worth mentioning that the computational complexity of the proposed algorithm does not grow with increasing number of data points. Also, the method proposed does not require

a fully trained network but a network partially trained for few epochs (*pretrain_iter*). For instance, SAE and agglo-SAE were both trained for 400 epochs. In the case of agglo-SAE, the network was first pre-trained for 200 epochs, then clustering was performed, followed by fine-tuning for another 200 epochs. It must be noted that the last 200 epoch is faster in agglo-network than its counterpart. This compensates for the clustering overhead. In Algorithm 2, hyperparameters *pretrain_iter*, *scan_iter*, and *max_iter* were set to 200, 50, and 150 respectively. It is remarked that this choice of iteration parameters should ensure that both traditional AEs and agglo-based ones were trained with similar training time. Also in experiments using Algorithm 2, $\Delta\tau$ was set to 0.5 and τ initialized to 1.

All experiments were performed on Intel(r) Core(TM) i7-6700 CPU @ 3.40Ghz and a 64GB

Algorithm 2 Automatic Feature Extraction (AFE)

function AFE(data,autoencoder(AE)):
  initialize: n'_0, *pretrain_iter*, *scan_iter*, *max_iter*
  initialize: $\mathbf{W}^{(\text{init})}$, $\mathbf{b}^{(\text{init})}$, k, τ,$\Delta\tau$
  $\mathbf{W}$, $\mathbf{b}$ = TRAIN_AE$\big($data, n'_0, $\mathbf{W}^{(\text{init})}$, $\mathbf{b}^{(\text{init})}$, *pretrain_iter*$\big)$
  $\mathcal{L}$ = COMPUTE_LOSS$\big($data, $\mathbf{W}$, $\mathbf{b}\big)$ as in (2)
  while $\mathcal{L}^{(iter+1)} < \mathcal{L}^{(iter)}$ **do** {
    $n'_{(iter+1)}$, $\mathbf{W}^{(agglo)}$, $\mathbf{b}^{(agglo)}$ = AGGLOMCLUSTERING $\big(\mathbf{W}, \mathbf{b}, \tau\big)$
    $\mathbf{W}^{(iter+1)}$, $\mathbf{b}^{(iter+1)}$ = TRAIN_AE$\big($data, $n'_{(iter+1)}$, $\mathbf{W}^{(agglo)}$, $\mathbf{b}^{(agglo)}$, *scan_iter*$\big)$
    $\mathcal{L}^{(iter+1)}$ = COMPUTE_LOSS$\big($data, $\mathbf{W}^{(iter+1)}$, $\mathbf{b}^{(iter+1)}\big)$
    if $\mathcal{L}^{(iter+1)} \geq \mathcal{L}^{(iter)}$:
      return $n'_{(iter)}$
    $\tau \longleftarrow \tau + \Delta\tau$ }
  $\mathbf{W}$, $\mathbf{b}$ = TRAIN_AE$\big($data, $n'_{(iter)}$, $\mathbf{W}^{(iter)}$, $\mathbf{b}^{(iter)}$, max_iter$\big)$
end function

of RAM running a 64-bit Windows 10 Enterprise edition. The software implementation has been with MATLAB 2015b, and LBFGS in minFunc (Byrd et al. (1995)) is used to minimize the objective function. The usage time in seconds is the time elapsed in seconds a fully trained deep network (DN) requires to classify all the test samples.

Table 1: Parameter settings

Parameters	SAE	NCAE	L_1/L_2-NCAE
Sparsity penalty (β)	3	3	3
Sparsity parameter (p)	0.05	0.05	0.05
Weight decay penalty (α)	3e-3	-	-
Nonnegativity constraint penalty (α_1)	-	-	1e-4
Nonnegativity constraint penalty (α_2)	-	3e-3	3e-3
Maximum No. of Iterations	400	400	400

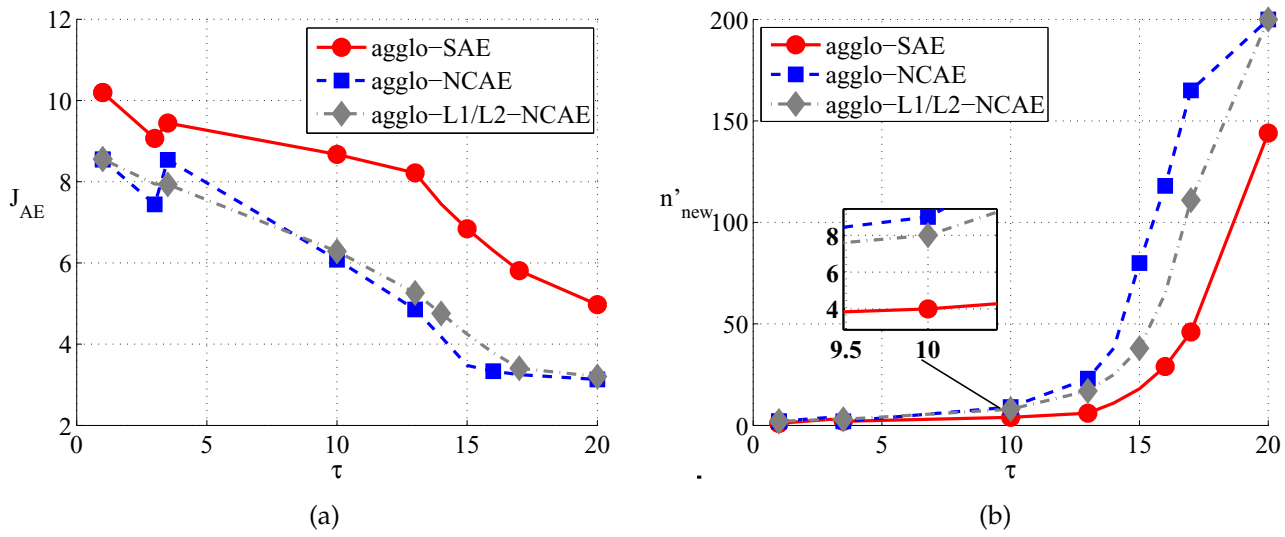


Figure 3: Performance of AE on the NORB dataset. (a) Reconstruction error vs. cluster similarity threshold. (b) Number of RFs vs. cluster similarity threshold for 196 initial filters.

4.1. Unsupervised Feature Reduction via Filter Reduction

In the first set of experiments, conventional SAE with 100 hidden units is trained using the NORB normalized-uniform dataset. This dataset contains 24,300 training images and 24,300 test images of 50 toys from 5 generic categories: four-legged animals, human figures, airplanes, trucks, and cars. The training and test sets consist of 5 instances of each category. Each image consists of two channels, each of size 96×96 pixels. We take the inner 64×64 pixels of one of the channels and resize it using bicubic interpolation (Kang et al. (2012); Han (2013)) to 32×32 pixels vector with 1024 entries as the input. To foster the claim that filter duplication is probable, the indexed receptive fields learned from NORB data are shown in Fig. 2a. Fig. 2b shows select RFs from Fig. 2a. The visual inspection indicates that a good number of these filters are very similar.

In order to eliminate the redundancy of RFs in Fig. 2a, SAE is retrained with the Algorithm 1.

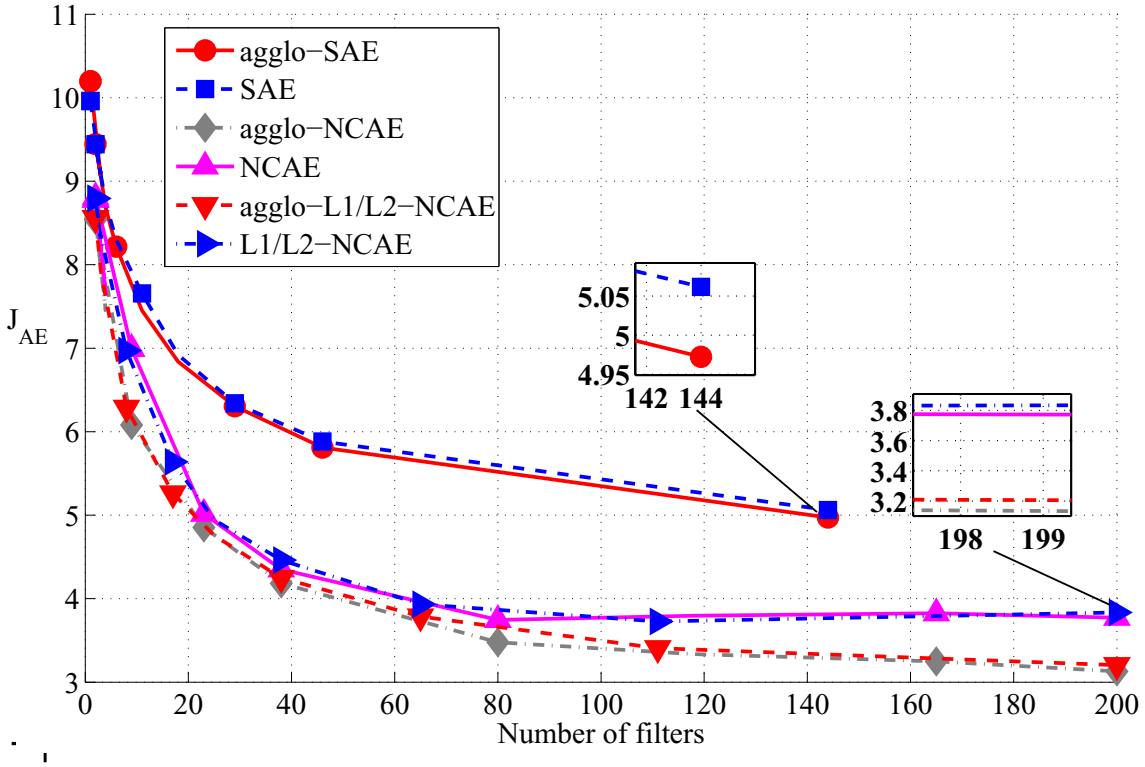


Figure 4: Reconstruction error using SAE, NCAE, and L_1 / L_2-NCAE trained with the same number of hidden units from experiment with agglo-SAE, agglo-NCAE, and agglo-L_1 / L_2-NCAE on NORB data.

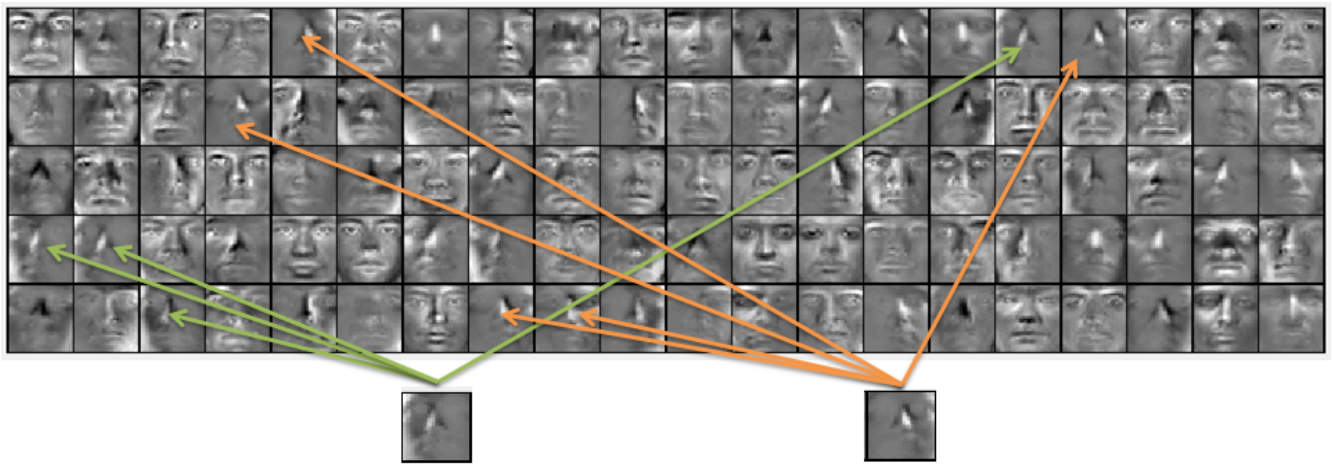


Figure 5: 100 receptive fields learned from Yale Face Dataset using SAE with examples of two duplicative RFs.

Fig. 2c shows the 32 RFs learned using agglo-SAE with similarity threshold $\tau = 16$. A quick inspection reveals that most duplicative filters in Fig. 2a have been grouped in Fig. 2c. In order

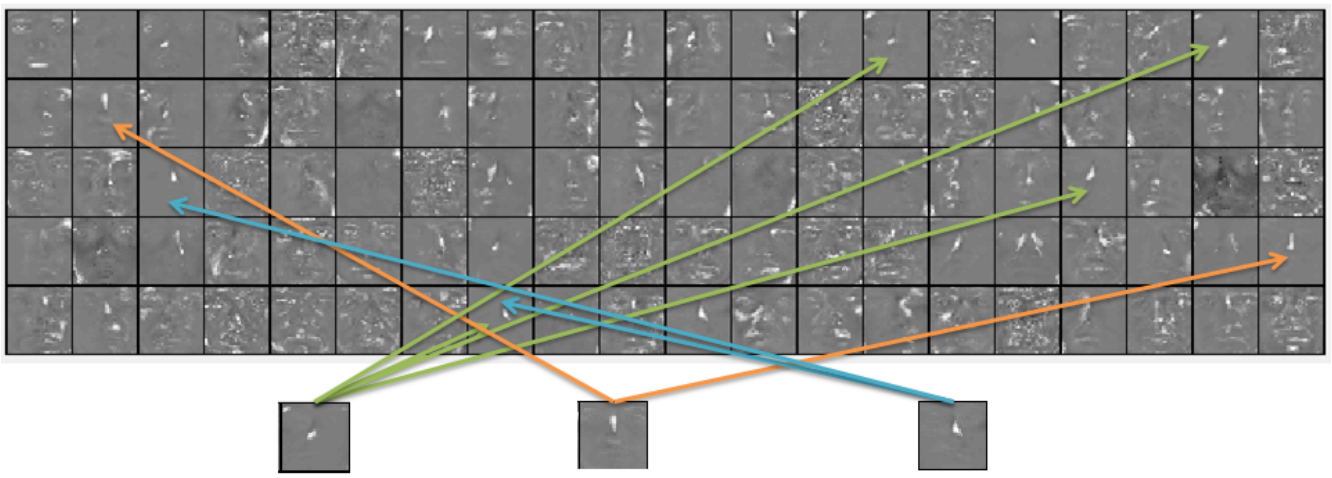


Figure 6: 100 receptive fields learned from Yale Face Dataset using NCAE with examples of duplicative RFs.

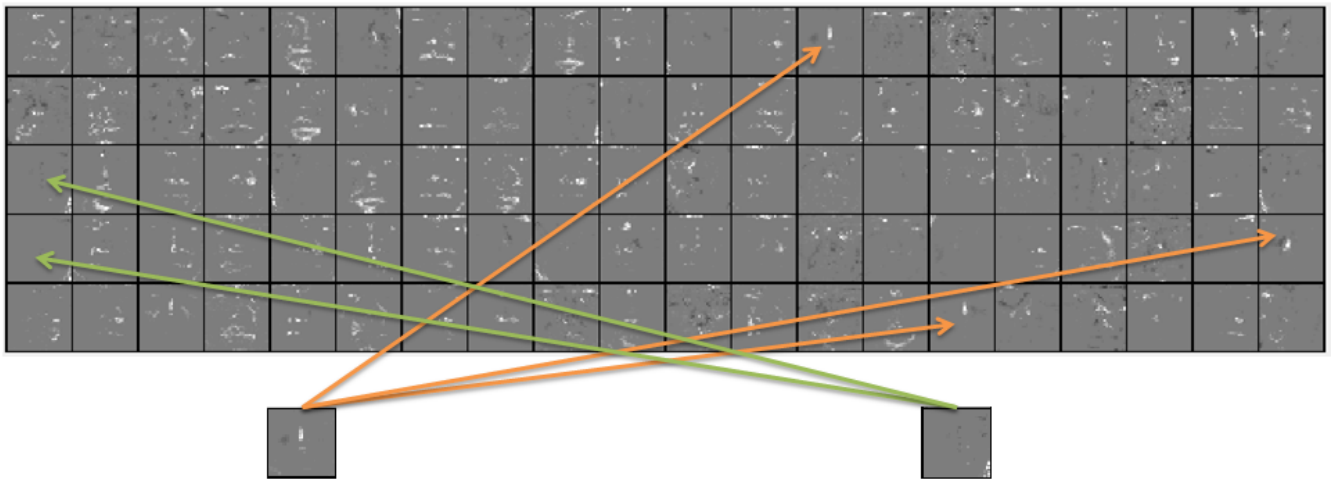
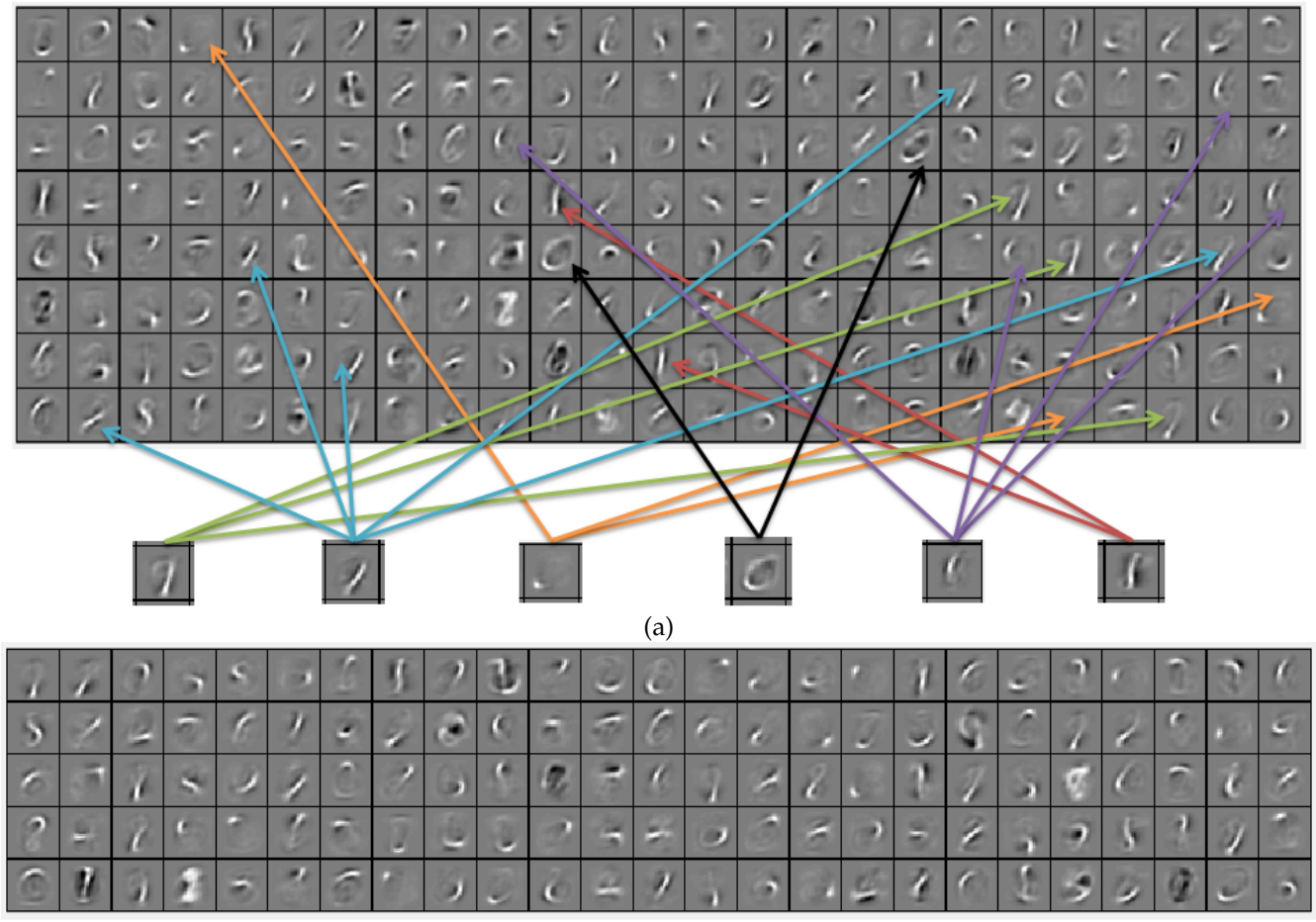


Figure 7: 100 receptive fields learned using L_1/L_2-NCAE from Yale Face Dataset, with examples of duplicative RFs.

to compare filter duplication and the consequent filtering redundancy with and without the agglomerative clustering approach, conventional SAE was trained with the same number of 32 hidden units. As shown in Fig. 2d, a number of RFs resulting from SAE training with 32 hidden units can still be observed even though the network was trained with the same number of hidden units as produced by agglo-SAE heuristic.

Similarly, Nonnegativity Constrained Autoencoder (NCAE) (Hosseini-Asl et al. (2016)) and L_1/L_2-NCAE (Ayinde et al. (2016)) with 200 hidden units each were trained using the NORB data. As observed in Fig 3a, both agglo-NCAE and agglo-L_1/L_2-NCAE achieve better recon-



(a)

(b)

Figure 8: Filters learned from MNIST data set using SAE (a) 200 RFs with examples of duplicative filters, (b) 125 RFs for agglo-SAE with τ=6.

struction accuracy than agglo-SAE. It can also be observed in Fig. 3b that more distinct filters are produced as τ is varied in agglo-NCAE and agglo-L_1/L_2-NCAE than in agglo-SAE. This indicates that imposing nonnegativity constraint on the network's weights helps in learning an elevated number of distinct features, while also improving the reconstruction. Again, SAE, NCAE, and L_1/L_2-NCAE were trained with the same number of hidden units that resulted from the experiment in Fig. 3b. The error curves shown in Fig 4 reveal that networks trained with agglo-SAE, agglo-NCAE and agglo-L_1/L_2-NCAE also yield a lower reconstruction error in comparison with those trained without agglomerative-clustering-based approach. Fig 4 is averaged over ten experiments to show the statistical significance of improved reconstruction

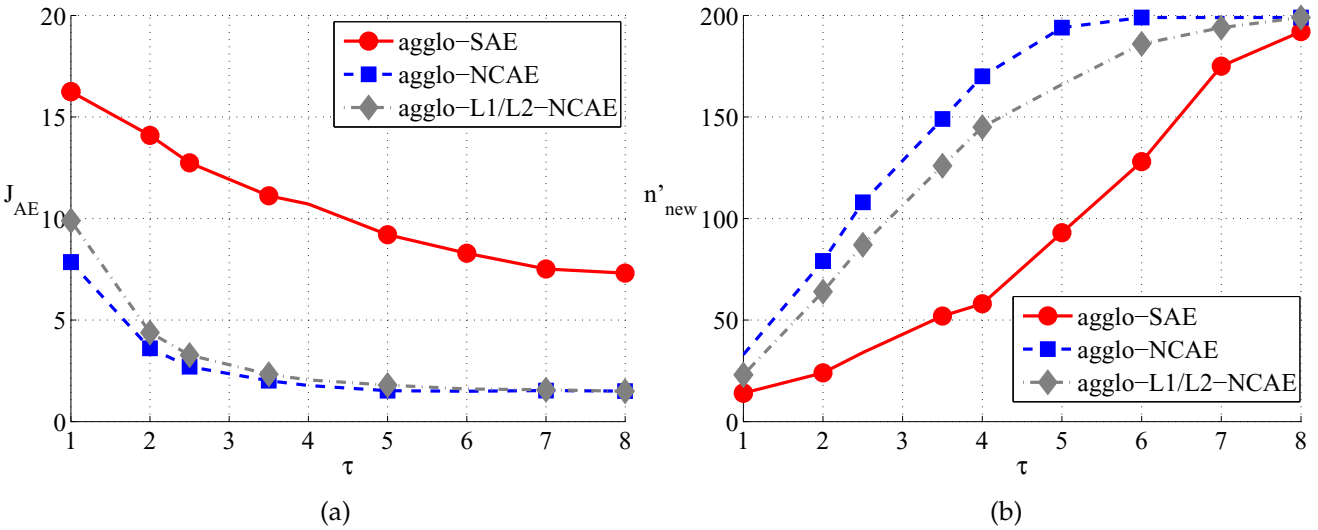


Figure 9: Performance of AE on the MNIST dataset vs. cluster similarity threshold (a) Reconstruction error (b) RF size (n'_{new}) using 200 initial filters.

capability of agglo-AEs over the traditional AEs considered (SAE, NCAE and L1/L2-NCAE).

The second set of experiments is to show redundant feature extraction using AEs trained on Yale Face Dataset (Belhumeur et al. (1997)). The database contains 11 images of 15 individuals, one per different facial expression or configuration: center-light, w/glasses, happy, left-light, w/no glasses, normal, right-light, sad, sleepy, surprised, and wink. The original size of each image is 320×243 with 256 gray levels per pixel. We resize each image to 32×32 to reduce the computational time and normalize between 0 and 1 (Cai et al. (2007)). SAE, NCAE and L_1/L_2-NCAE each with 100 hidden units were trained on this dataset and the RFs learned are shown in Figs 5, 6 and 7, respectively. It was observed that some of the filters are very similar and will thereby extract similar features. Again, it is observed that both types of nonnegativity-constrained AEs also learned duplicative filters that produce redundant filtering. This experiment shows that the tendency to learn redundant features is not specific to conventional SAE only but to a variety AE architectures.

In the third experiment, SAE, NCAE and L_1/L_2-NCAE were trained using the MNIST digits dataset. The standard MNIST dataset has 60000 training and 10000 testing examples. Each example is a grayscale image of an handwritten digit scaled and centered in a 28 × 28 pixel box (LeCun et al. (1998)). A careful look at the 200 RFs of trained SAE in Fig. 8a shows that

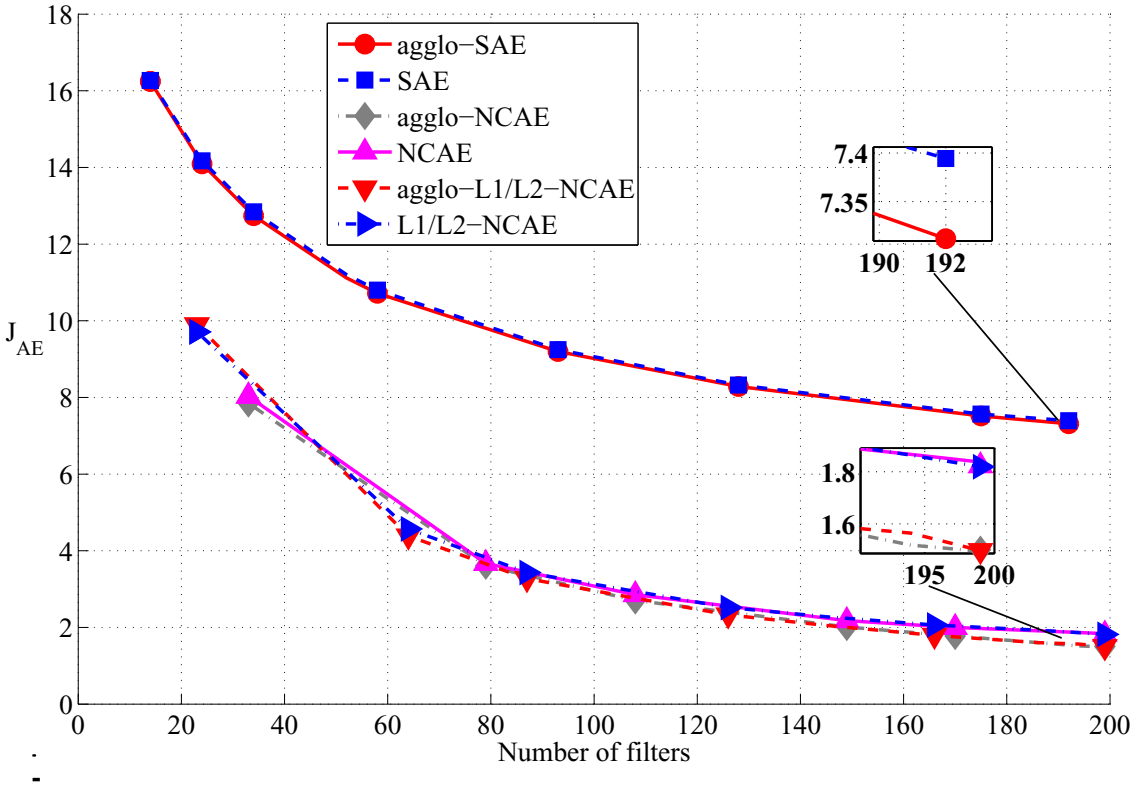


Figure 10: Reconstruction error using SAE, NCAE, and L_1/L_2-NCAE trained with the same number of hidden units from experiment with agglo-SAE, agglo-NCAE, and agglo-L_1/L_2-NCAE on MNIST data.

many of the filters are duplicative and redundant, and thereby resulting in redundant over-representation of data and increased computational complexity. By visual inspection it can be observed that a good number of filters can be considered as redundant and eliminated with no significant loss of reconstruction accuracy. By deploying Algorithm 1, 75 redundant filters were eliminated and the resulting distinct agglomerative filters are shown in Fig. 8b. In the case of agglo-SAE, it was observed that the reconstruction error does not decrease significantly beyond the similarity threshold of 7 in Fig. 9a, which corresponds to approximately 172 filters in Fig. 9b. No significant reduction in the reconstruction error was observed for agglo-NCAE and agglo-L_1/L_2-NCAE for values of τ greater than 5 corresponding to 194 and 166 distinct filters, respectively. It is remarked that decreasing τ will decrease the number of RFs and hence increase the error. By implication, increasing τ will lead to the increase of duplicative RFs. The

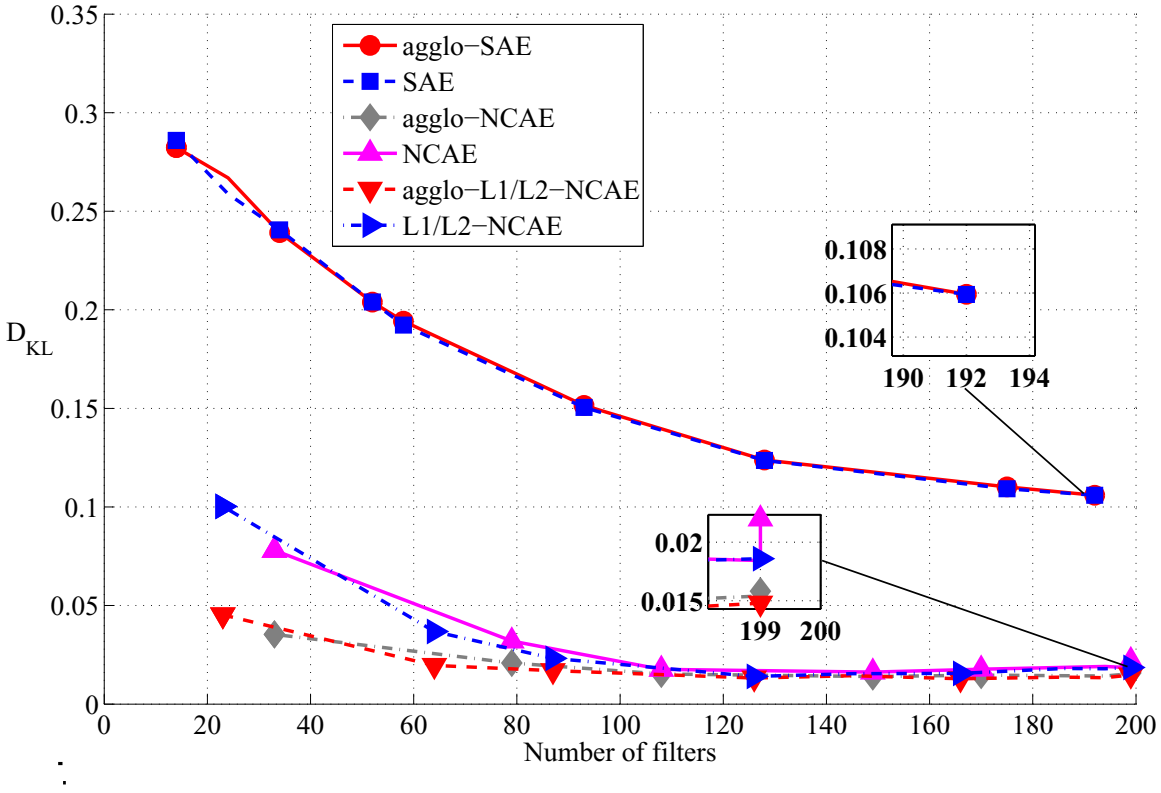


Figure 11: KL-Divergence sparsity measure with respect to a desired $p = 0.05$ using SAE, NCAE, and L_1/L_2-NCAE trained with the same number of hidden units (n'_{new}) from experiment with agglo-SAE, agglo-NCAE, and agglo-L_1/L_2-NCAE on MNIST data.

two bottom curves on Fig. 9a also indicate that nonnegativity constraints yield better reconstruction quality on this data set. It is again shown in Fig. 9b that imposition of nonnegativity constraints on the network's weights results in networks with larger n'_{new} for a large range of similarity τ compared to SAE.

Similarly, SAE, NCAE, and L_1/L_2-NCAE were trained with the same n'_{new} that resulted from the experiment in Fig. 9b using the MNIST handwritten digits data. It can be observed in Fig. 10 that the proposed agglomerative-based heuristic improves the reconstruction accuracy for all three AEs. Also, the sparsity has increased in agglo-NCAE and agglo-L_1/L_2-NCAE, respectively, compared to their counterparts NCAE and L_1/L_2-NCAE as shown in Fig. 11.

However, no obvious sparsity improvement was noticed in the case of agglo-SAE. The proposed heuristic has been also evaluated based on the distribution of data in high level feature

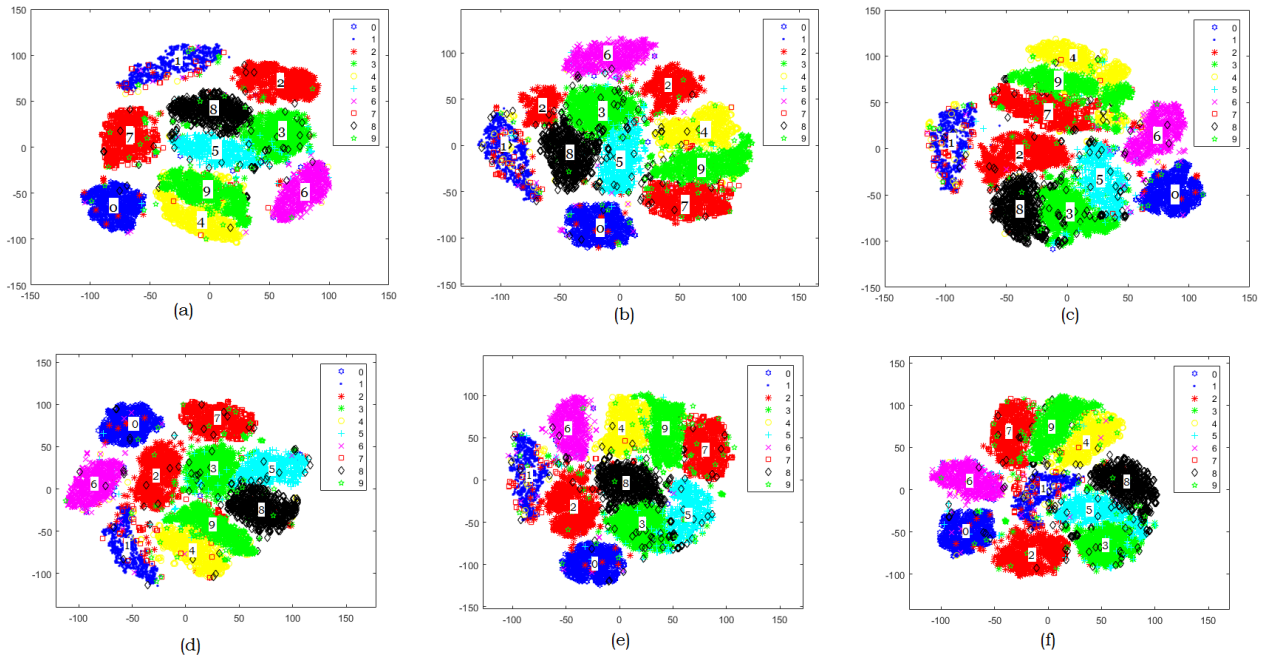


Figure 12: t-SNE projection (der Maaten and Hinton (2008)) of 200D representations of MNIST handwritten digits using (a) SAE (b) NCAE (c) L_1/L_2-NCAE, (d) 174D representations using agglo-SAE (e) 153D representations using agglo-NCAE, and (f) 172D representations using agglo-L_1/L_2-NCAE.

space. In this regard, t-distributed stochastic neighbor embedding (t-SNE) was used to project the high-level representations (that is, the hidden activations) of SAE, NCAE, L_1/L_2-NCAE, agglo-SAE, agglo-NCAE, and agglo-L_1/L_2-NCAE to 2D space (der Maaten and Hinton (2008)). The 2D projections of high-level (200D) representations of MNIST handwritten digits test set corresponding to hidden activities of SAE, NCAE, and L_1/L_2-NCAE are respectively visualized in Figs. 12a,b, and c. For the purpose of comparison, 174, 153, and 172D representations of MNIST handwritten digits for agglo-SAE, agglo-NCAE, and agglo-L_1/L_2-NCAE are respectively depicted in Figs. 12d,e, and f.

The t-SNE projections in Fig. 12 no visible/obvious deterioration in the manifolds of the 174D representations of agglo-SAE compared to 200-D representations of SAE. This is an indication that more than 25 hidden activations are redundant and eliminating them does not deteriorate the manifold shown in Fig. 12a and d. Similarly as shown in Fig. 12b and e, approximatively 50 hidden activations can be eliminated without overlapping the manifolds that enclose the 200-D representations of MNIST digits using NCAE. In a similar manner, elimina-

tion of more than 25 duplicative hidden activities has not adversely affected the manifolds of the digits' projections as shown in Fig. 12f of agglo-L_1/L_2-NCAE in comparison with Fig. 12c of L_1/L_2-NCAE.

Table 2: Classification performance on MNIST dataset using initial network configuration 784-200-20-10.

Architecture		Mean (±SD)	p-value	Usage time (±SD) (ms)
SAE (n'_1=200)	Λ_b	**0.860** (±0.007)	0.1517	59.9 (±3.51)
	Λ_a	0.977 (±0.0011)	0.8268	60.6 (±2.51)
agglo-SAE (τ=7)	$n'_{1,new}$	174 (±2)	-	-
	Λ_b	0.855 (±0.005)	-	59.2 (±1.93)
	Λ_a	**0.978** (±5.8e-4)	-	54.0 (±2.09)
NCAE (n'_1=200)	Λ_b	0.849 (±0.008)	0.028	59.2 (±2.14)
	Λ_a	**0.975** (±8.94e-4)	0.0645	55 (±2.27)
agglo-NCAE (τ=3.5)	$n'_{1,new}$	153 (±3)	-	-
	Λ_b	**0.852** (±0.0061)	-	50.9 (±2.44)
	Λ_a	0.974 (±0.0016)	-	48.6 (±1.29)
L_1/L_2-NCAE (n'_1 = 200)	Λ_b	**0.845** (±0.205)	0.108	62.1 (±4.13)
	Λ_a	0.974 (±0.0014)	0.7374	56.3 (±1.90)
agglo-L_1/L_2-NCAE (τ=5)	$n'_{1,new}$	172 (±3)	-	-
	Λ_b	0.842 (±0.0067)	-	58.2 (±1.84)
	Λ_a	0.974 (±8.9e-4)	-	53.1 (±1.12)

4.2. Supervised Learning

In the last set of experiments, a DN was tested using two stacked AEs and a softmax classification output to evaluate the effect of filter reduction on classification. The network was fine tuned by backpropagation algorithm to improve the classification accuracy. MNIST dataset was utilized for the first set of experiments, and it is noted that the filter reduction algorithm is only implemented on the first layer of the DN for experiments reported in Table 2. However, as will be shown below, this concept can also be applied to all layers of the deep architectures.

For convenience of presentation on Tables 2-7, Λ_b and Λ_a are respectively defined in (12) and (13); $n'_{1,new}$ and $n'_{2,new}$ are the number of agglomeratve filters in the first and second layer, respectively.

$$\Lambda_b = \frac{\text{Number of correctly classified test cases before supervised fine-tuning}}{\text{Total number of test cases}} \tag{12}$$

$$\Lambda_a = \frac{\text{Number of correctly classified test cases after supervised fine-tuning}}{\text{Total number of test cases}} \tag{13}$$

where Λ_b is commonly referred to as accuracy before fine-tuning and Λ_a as accuracy after fine-tuning.

The classification accuracy and testing time are reported in Table 2 for the three benchmark AEs. Reported are averaged results of 10 independent trials and related mean values and standard deviation (SD) of 6 simulation series. It can be observed from the results that removing redundant filters in agglo-SAE-pretrained network does not deteriorate the classification performance of the DN. $\tau = 7$ was chosen based on the result in Fig. 9a, which shows that the reconstruction error does not significantly decrease beyond $\tau = 7$. This indicates that no matter how many filters are added beyond n'_{new} corresponding to $\tau = 7$, the performance of the DN is not likely to improve. One of the direct aftermaths of filter reduction is the drastic reduction in training time and the usage time. It is apparent from the results that more than 6ms have been saved on the usage time, which is significant when dealing with very large datasets encountered in real world scenarios. However, for agglo-NCAE-pretrained DN, a slight improvement of $\approx 1\%$ in classification accuracy was observed before fine-tuning and this can be due to the features extracted in agglo-NCAE that are more sparse than those obtained using conventional NCAE. As can be inferred from the p-values, there is a significant difference in the performance before fine-tuning and no significant improvement observed after fine-tuning. Again τ=3.5 for NCAE was chosen in connection with Fig. 9a.

The observations from experiments with DN pretrained using agglo-L_1/L_2-NCAE also reveal that usage time is reduced and no significant difference was noticed in the accuracy before and after finetuning compared to the L_1/L_2-NCAE-pretrained network. It is worthy of note that the hidden size of the second layer used in the first set of experiments was chosen to be

Table 3: Classification performance on MNIST dataset using initial network configuration 784-1000-20-10.

Architecture		Mean (±SD)	p-value	Usage time (±SD) (ms)
SAE (n'_1=1000, n'_2=20)	Λ_b	0.8063 (±0.0124)	0.0004	281.8 (±25.0)
	Λ_a	0.9782 (± 0.000704)	0.0030	257.1 (± 34.2)
agglo-SAE (τ=7)	$n'_{1,new}$	889 (±6)	-	-
	$n'_{2,new}$	17 (±1)	-	-
	Λ_b	**0.8514** (±0.0106)	-	147.9 (±2.5)
	Λ_a	**0.9819** (±9.2e-4)	-	146.7 (± 0.35)
DpAE (20% dropout)	Λ_b	0.7314 (± 0.0363)	0.0017	209.7 (± 30.9)
	Λ_a	0.9585 (±0.0107)	0.0090	203.4 (± 38.5)

20 for computational reasons and negligible reduction of RFs number was noticed for most of τ values considered. However, the size of the hidden layers of all the AEs were chosen to be as large as the hidden layer size of the first AE in subsequent experiments and near-optimal number of RFs were obtained using the proposed heuristic.

In the next large-scale experiment reported in Tables 3-6, the effect of removing redundant filters on network performance is demonstrated using the MNIST dataset. Every run of the experiments is repeated five times and averaged for statistical significance. The classification performance of the DN pre-trained with DpAE (Hinton et al. (2012)) was used as a benchmark. In Table 3, the number of hidden units in the first and second layers were set to 1000 and 20 respectively. It can be observed from the results that more than 100 redundant filters have been removed in the first layer and 2 in second layer. An improved classification accuracy in agglo-SAE was also observed after eliminating the redundancy compared to SAE and DpAE counterparts. In addition, the usage times of SAE and DpAE-trained networks have been respectively reduced more than 40% and 28% in agglo-SAE. Similar trends were observed in the classification accuracy after fine-tuning when the number of hidden units of the first and second layer were respectively set to 1000 and 200 as detailed in Table 4.

Table 4: Classification performance on MNIST dataset using initial network configuration 784-1000-200-10.

Architecture		Mean (±SD)	p-value	Usage time (±SD) (ms)
SAE (n'_1=1000, n'_2=200)	Λ_b	0.9604 (± 0.0014)	0.0151	324.1(±39)
	Λ_a	0.9667 (±0.0015)	0.0172	290.2(±16)
agglo-SAE (τ=7)	$n'_{1,new}$	874 (±13)	-	-
	$n'_{2,new}$	95(±2)	-	-
	Λ_b	0.9422 (± 0.0039)	-	162.3 (±43)
	Λ_a	**0.9826** (±3.9e-4)	-	165.0 (±35.6)
DpAE (20% dropout)	Λ_b	**0.9612** (± 0.0024)	0.3743	193.4 (±37.8)
	Λ_a	0.9768 (± 0.0013)	0.3738	209.2 (± 41.4)

In an attempt to investigate the effect of the proposed algorithm on overfitting, the number of hidden units of the first and second layer were increased to 500 and 500, respectively as in Table 5 and 1000-1000 as in Table 6. One of the key observations in Tables 5-6 is that as the AE's hidden layer size grows, the performance of SAE deteriorates. It was also observed that the use of dropout did not help in improving the performance. It is remarked that during the experiments, 20%, 30%, and 50% dropout fractions were tested and the one with the best output performance was reported. However, in these experiments, agglo-SAE outperforms both SAE and DpAE before and after fine-tuning. The performance of agglo-SAE could be traced to its ability to eliminate filtering redundancy and it is worth mentioning that such elimination might help in reducing overfitting. Table 7 reports the classification results on the small NORB data set and demonstrates that the network with agglo-SAE outperforms the two other networks before and after fine-tuning. It can be noticed that more than 50% of the filters in first layer and 75% in the second layer were redundant and removed with improved classification performance. Lastly, it was also observed that the results obtained in most of the experiments using Algorithm 2 are close to the results obtained using Algorithm 1. The conclusion drawn from this observation is that Algorithm 2 implements faster search of hyperparameter τ, hence it is more practical in the training phase.

Table 5: Classification performance on MNIST dataset using initial network configuration 784-500-500-10.

Architecture		Mean (±SD)	p-value	Usage time (±SD) (ms)
SAE (n'_1=500, n'_2=500)	Λ_b	0.9595 (±0.0012)	<0.0001	240.6 (±10)
	Λ_a	0.9642 (± 0.0012)	0.0025	223.6 (±16)
agglo-SAE (τ=7)	$n'_{1,new}$	483 (±3)	-	-
	$n'_{2,new}$	392 (±3)	-	-
	Λ_b	**0.9708** (± 0.0013)	-	220.8 (±9.8)
	Λ_a	**0.9785** (± 0.0046)	-	211.8 (± 20)
DpAE (20% dropout)	Λ_b	0.9640 (±0.0074)	0.1777	253.7 (± 12.2)
	Λ_a	0.9704 (± 0.0112)	0.1777	227.1 (±19.8)

Table 6: Classification performance on MNIST dataset using initial network configuration 784-1000-1000-10.

Architecture		Mean (±SD)	p-value	Usage time (±SD) (ms)
SAE (n'_1=1000, n'_2=1000)	Λ_b	0.9647 (± 0.0012)	0.0004	480.6 (±18.1)
	Λ_a	0.9689 (±9.6e-4)	<0.0001	476.1 (±57.9)
agglo-SAE (τ=7)	$n'_{1,new}$	844 (±6)	-	-
	$n'_{2,new}$	272 (±10)	-	-
	Λ_b	**0.9730** (±8.9e-4)	-	298.8 (± 8.2)
	Λ_a	**0.9812** (±9.0e-4)	-	275.9 (±17)
DpAE (20% dropout)	Λ_b	0.9464 (±0.0513)	0.3124	403.4 (±59.5)
	Λ_a	0.8483 (± 0.2701)	0.3336	378.2 (±57.7)

5. Conclusion

This paper proposes new techniques for data representation in the context of DL for stacked AEs by leveraging on the ability to agglomerate regularized sparse RFs and also by enhancing the feature generation process at the output layer via controlled feature compression. The performance of the proposed method in terms of decomposing data into parts and non-redundant feature extraction was compared for the conventional SAE with constrained AEs of type DpAE,

Table 7: Classification performance on NORB dataset.

Architecture		Mean (±SD)	p-value	Usage time (±SD) (ms)
SAE (n_1'=500, n_2'=500)	Λ_b	0.8085 (± 0.0065)	0.3516	385.7 (±38.8)
	Λ_a	0.8143 (±0.0065)	0.0313	387.8 (±34.6)
agglo-SAE (τ=14)	$n'_{1,new}$	242 (±14)	-	-
	$n'_{2,new}$	123 (±8)	-	-
	Λ_b	**0.8129** (±0.0102)	-	159.1 (±27.0)
	Λ_a	**0.8303** (±0.0087)	-	170.7 (±31.8)
DpAE (20% dropout)	Λ_b	0.5042 (±0.1377)	0.0062	443.0 (±66.1)
	Λ_a	0.5090 (±0.1208)	<0.001	475.2 (± 57.3)

NCAE, and L_1/L_2-NCAE. The proposed technique uses agglomerative clustering which starts off by allocating each original filter to a separate cluster and merges two most similar clusters as long as the average *similarity* between their members is above a set threshold τ. This concept is illustrated using the NORB normalized-uniform object data set, MNIST handwritten digits data and Yale Face Dataset. The results show that a large number of originally generated RFs are overlapping across these three data sets, creating unnecessary amount of filtering redundancy. By using the proposed methods, such redundancy can be controlled, eliminated and AEs are enabled to extract fewer and more distinctive features.

Acknowledgement

This work was sponsored in part by the National Science Foundation under Grant 1641042.

References

Ayinde, B. and J. Zurada (2016). Clustering of receptive fields in autoencoders. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pp. 1310–1317. IEEE.

Ayinde, B. O., E. Hosseini-Asl, and J. M. Zurada (2016). Visualizing and understanding non-negativity constrained sparse autoencoder in deep learning. In *Rutkowski L., Korytkowski M.,*

Scherer R., Tadeusiewicz R., Zadeh L., Zurada J. (eds) Artificial Intelligence and Soft Computing. ICAISC 2016. Lecture Notes in Computer Science, vol 9692, pp. 3–14. Springer.

Belhumeur, P. N., J. P. Hespanha, and D. J. Kriegman (1997). Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 19*(7), 711–720.

Bengio, S., L. Deng, H. Larochelle, H. Lee, and R. Salakhutdinov (2013). Guest editors introduction: Special section on learning deep architectures. *IEEE Transactions on Pattern Analysis and Machine Intelligence 35*(8), 1795–1797.

Bengio, Y. (2009). Learning deep architectures for ai. *Foundations and trends® in Machine Learning 2*(1), 1–127.

Bengio, Y., A. Courville, and P. Vincent (2013). Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 35*(8), 1798–1828.

Bengio, Y. and Y. LeCun (2007). Scaling learning algorithms towards ai. *Large-Scale Kernel Machines 34*(1), 1–41.

Byrd, R. H., P. Lu, J. Nocedal, and C. Zhu (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing 16*(5), 1190–1208.

Cai, D., X. He, Y. Hu, J. Han, and T. Huang (2007). Learning a spatially smooth subspace for face recognition. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition Machine Learning (CVPR'07)*.

Chorowski, J. and J. M. Zurada (2015). Learning understandable neural networks with nonnegative weight constraints. *Neural Networks and Learning Systems, IEEE Transactions on 26*(1), 62–69.

Deng, L. (2014). A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing 3*, e2.

der Maaten, L. V. and G. Hinton (2008). Visualizing data using t-sne. *Journal of Machine Learning Research 9*(11).

Ding, C. and X. He (2002). Cluster merging and splitting in hierarchical clustering algorithms. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pp. 139–146. IEEE.

Han, D. (2013). Comparison of commonly used image interpolation methods. *ICCSEE, Hangzhou, China*.

He, Y., K. Kavukcuoglu, Y. Wang, A. Szlam, and Y. Qi (2013). Unsupervised feature learning by deep sparse coding. *arXiv preprint arXiv:1312.5783*.

Hinton, G., S. Osindero, and Y. W. Teh (2006). A fast learning algorithm for deep belief nets. *Neural Computation 18*(7), 1527–1554.

Hinton, G. and R. Salakhutdinov (2006). Reducing the dimensionality of data with neural networks. *Science 313*(5786), 504–507.

Hinton, G. E., N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Hosseini-Asl, E., J. M. Zurada, and O. Nasraoui (2016). Deep learning of part-based representation of data using sparse autoencoders with nonnegativity constraints. *Neural Networks and Learning Systems, IEEE Transactions on 27*(12), 2486–2498.

Kang, X., S. Li, and J. Hu (2012). Fusing soft-decision-adaptive and bicubic methods for image interpolation. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pp. 1043–1046. IEEE.

LeCun, Y., Y. Bengio, and G. Hinton (2015). Deep learning. *Nature 521*, 436–444.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*(11), 2278–2324.

Lee, H., C. Ekanadham, and A. Ng (2007). Sparse deep belief net model for visual area v2. *Advances in Neural Information Processing Systems 7*, 873–830.

Leibe, B., A. Leonardis, and B. Schiele (2004). Combined object categorization and segmentation with an implicit shape model. In *Workshop on Statistical Learning in Computer Vision, ECCV*, Volume 2, pp. 7.

Lemme, A., R. Reinhart, and J. Steil (2012). Online learning and generalization of parts-based image representations by non-negative sparse autoencoders. *Neural Networks 33*, 194–203.

Li, J., H. Chang, and J. Yang (2015). Sparse deep stacking network for image classification. *arXiv preprint arXiv:1501.00777*.

Manickam, S., S. D. Roth, and T. Bushman (2000). Intelligent and optimal normalized correlation for high-speed pattern matching. *Datacube Technical Paper*.

Nair, V. and G. E. Hinton (2009). 3d object recognition with deep belief nets. *Advances in Neural Information Processing Systems*, 1339–1347.

Ng, A. (2011). Sparse autoencoder. In *CS294A Lecture notes*, URL https://web.stanford.edu/class/cs294a/sparseAutoencoder_2011new.pdf. Stanford University.

Ngiam, J., Z. Chen, S. A. Bhaskar, P. W. Koh, and A. Y. Ng (2011). Sparse filtering. In *Advances in Neural Information Processing Systems*, pp. 1125–1133.

Ngiam, J., A. Coates, A. Lahiri, B. Prochnow, Q. V. Le, and A. Y. Ng (2011). On optimization methods for deep learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 265–272.

Olshausen, B. A. and D. J. Field (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature 381*(6583), 607–609.

Poultney, C., S. Chopra, and Y. Cun (2006). Efficient learning of sparse representations with an energy-based model. *Advances in Neural Information Processing Systems*, 1137–1144.

Schulz, H., K. Cho, T. Raiko, and S. Behnke (2015). Two-layer contractive encodings for learning stable nonlinear features. *Neural Networks 64*, 4–11.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research 15*(1), 1929–1958.

Walter, B., K. Bala, M. Kulkarni, and K. Pingali (2008). Fast agglomerative clustering for rendering. In *IEEE Symposium on Interactive Ray Tracing*, pp. 81–86. IEEE.

Xu, J., L. Xiang, R. Hang, and J. Wu (2014). Stacked sparse autoencoder (ssae) based framework for nuclei patch classification on breast cancer histopathology. In *Biomedical Imaging (ISBI), 2014 IEEE 11th International Symposium on*, pp. 999–1002. IEEE.