DEEP HYBRID NETWORKS WITH GOOD *OUT-OF-SAMPLE* OBJECT RECOGNITION

Muhammad Ghifary, W. Bastiaan Kleijn, Mengjie Zhang

School of Engineering and Computer Science
Victoria University of Wellington
PO Box 600, Wellington, New Zealand
{muhammad.ghifary, bastiaan.kleijn, mengjie.zhang}@ecs.vuw.ac.nz

ABSTRACT

We introduce Deep Hybrid Networks that are robust to the recognition of *out-of-sample* objects, i.e., ones that are drawn from a different probability distribution from the training data distribution. The networks are based on a particular combination of an auto-encoder and stacked Restricted Boltzmann Machines (RBMs). The auto-encoder is used to extract sparse features, which are expected to be noise invariant in the observations. The stacked RBMs then observe the sparse features as inputs to learn the top hierarchical features. The use of RBMs is motivated by the fact that the stacked RBMs typically provide good performance when dealing with *in-sample* observations, as proven in the previous works. To improve the robustness against local noise, we propose a variant of our hybrid network by the usage of a mixture of sparse features and sparse connections in the auto-encoder layer. The experiments show that our proposed deep networks provide good performance in both the *in-sample* and *out-of-sample* situations, particularly when the number of training examples is small.

Index Terms— deep hybrid network, *out-of-sample*, noise robustness, sparse features, object recognition

1. INTRODUCTION

In real world problems, good learning algorithms for object recognition often degrade when facing the *out-of-sample* data, e.g., recognizing objects under unseen environment with noise and/or objects with a distribution different from the learned distribution. Furthermore, this problem becomes harder to solve when the availability of training samples is limited. To be useful in practical applications, object recognition methods should be able to deal with those unexpected situations.

To recognize noisy objects, a typical solution is to incorporate synthetic noise within the training set [1]. However, this may raise another problem if the type of unseen noise is unknown. Another solution is to design a robust model without any explicit noise compensation. It can be built based on handcrafted feature extractors (e.g., LBP [2] and HoG [3]) or based on learning good features (e.g., deep learning [4] and sparse coding [5]). It is likely that such methods effectively denoise the input during the mapping to the features.

In this work, our objective is to create a deep learning approach that learns noise-invariant hierarchical features that can be used for recognition without noise compensation at training time. It is widely known that deep learning now achieves the best performance in many *in-sample* object recognition tasks [6, 7, 8]. It can be considered a new breakthrough in speech recognition (see [9] for a recent overview). However, the robustness of deep learning against the *out-of-sample* data requires further investigation. A recent method by Tang and Eliasmith [10] is an example of deep learning approach that can handle *out-of-sample* problem. In the speech domains, several methods have been proposed to overcome the similar problems [11, 12].

We hypothesize that the robustness of hierarchical features induced by deep learning can be improved by preserving noise-invariant features in the bottom layer. The upper-layer features extracted from the invariant bottom-layer features are then trained using the basic deep learning algorithm of [4]. Based on that hypothesis, we developed a noise-robust deep architecture, which we refer to as Deep Hybrid Network (DHN). It consists of a sparse auto-encoder [13, 14] to train the bottom layer and stacked Restricted Boltzmann Machines (RBMs) [4] to train the upper layers. The sparse features induced by the auto-encoder are expected to be invariant to noisy observations so that the stacked RBMs always observe the lower-layer features as if they receive clean (*in-sample*) inputs. We also investigate the performance of the combination of sparsely-connected weights and sparse features, which produces a variant of the DHN that we refer to as sparse Deep Hybrid Network (sDHN). Our experiments show that the DHN and sDHN provide good performance over a wide range of seen and unseen conditions, particularly when the number of training examples is small.

2. RELATED WORK

In this section, we place our work in the context of the existing literature. Before we start our discussion, we introduce our notation and terminology.

Let us consider a deep neural network with l hidden layers. We will denote the visible vector and the hidden vectors by $\mathbf{v}$, $\mathbf{h}^{(1)}$, ..., $\mathbf{h}^{(l)}$, respectively, and the connection *weights* or *dictionary* between two adjacent layers by the matrices $\mathbf{W}^{(1)}$, ...,$\mathbf{W}^{(l-1)}$. Here we assume that each element in all vectors has a value in the range of $[0, 1]$. We will use two notions of sparsity. The first type of sparsity is that of *sparse features*, i.e., $\mathbf{h}^{(1)}$ with a few non-zero nodes. The second type of sparsity is that of *sparse weights*, i.e., $\mathbf{W}^{(1)}$ with a few non-zero elements formed a particular pattern.

We also define a notion of feature noise invariance in terms of ε-*invariance*. Let us denote a clean data and a noisy data vector by $\mathbf{v}, \tilde{\mathbf{v}} \in \mathbb{R}^{n_v}$, respectively, where $\tilde{\mathbf{v}} = \mathbf{v} + \psi$ with $\psi \in \mathbb{R}^{n_v}$ is an arbitrary unit-length noise vector. Let us also denote a j-th first hidden layer node activation given $\mathbf{v}$ by $h_j^{(1)}$ and the same node activation given $\tilde{\mathbf{v}}$ by $\tilde{h}_j^{(1)}$. The feature $\mathbf{h}^{(1)} \in \mathbb{R}^{n_{h^{(1)}}}$ is said to be ε-*invariant* if it satisfies the following condition

$$\begin{aligned} \forall j = 1, ..., n_{h^{(1)}}, \quad & d(h_j^{(1)}, \tilde{h}_j^{(1)}) \leq \eta\varepsilon \\ \text{subject to} \quad & \|\psi\| \leq \eta, \end{aligned} \tag{1}$$

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

where $d(\cdot,\cdot)$ is any distance measure, η is an upper bound for the noise magnitude, and ε is a small error constant. We suggest that a classifier is likely to be robust to noise with a certain bound in the limit that $\eta \rightarrow 0$ if it has features that satisfy (1) for a particular choice of ε.

Encouraging sparse features is one of the approaches to producing features that are robust to noise [15, 14]. It was initially achieved by sparse coding [5], which can produce sparse features and mimic certain properties of biological visual area V1, e.g., produce an interpretable dictionary like edge detector. It has performed well for image restoration [16, 17] and object classifications [18, 19]. In deep learning, there exist several approaches to imitating sparse coding to produce sparse features by regularization [20, 14]. It has been claimed that sparse features in deep learning may provide good robustness to noise [14] and to some *affine* transformations [21].

Because the sparse feature learning algorithm typically has two components: i) an interpretable dictionary, which induces sparse features, and ii) an encoder function, it raised a question whether the sparse features or the encoder function may lead to a better discriminative model. Coates and Ng [22] suggested that the classification performance might be strongly correlated to the choice of encoder function rather than the interpretability of the learned dictionary; some classifiers produce an excellent performance even using a randomly generated dictionary. However, the sparse features induced by the interpretable dictionary are useful when the availability of labeled examples is limited [22].

Recently, Tang and Eliasmith [10] proposed a robust-to-noise hybrid network referred to as sDBN, which is a Deep Belief Network with sparse weights in the bottom layer. It is trained without any noise compensation and performs well in the recognition of handwritten digits with some types of noise. The sparse weights are the result of locally assigned connections, i.e., assigning each $\mathbf{h}^{(1)}$ node to an $s \times s$ sub-window of visible layer $\mathbf{v}$ represented in a 2D matrix. With local connections, noise in a subset of $\mathbf{v}$ will only affect a particular subset of $\mathbf{h}^{(1)}$. Moreover, [10] also proposed a special kind of denoising method referred to as *probabilistic denoising*. It is applied to sDBN to recover a few nodes in $\mathbf{h}^{(1)}$ suspected as noise. The drawback using this denoising is that it increases the time complexity at both training and test time.

Without the *probabilistic denoising*, we hypothesize that the feature $\mathbf{h}^{(1)}$ in sDBN will be less invariant if it is subjected to distributed noise, e.g., Gaussian white and impulse noise. The performance may also degrade significantly when the number of labeled training data is small. Therefore, we need more general building blocks of deep network to preserve invariance over any type of noise.

3. OUR METHODS

The main motivation of this work is to reduce the change between $h_j^{(1)}$ and $\tilde{h}_j^{(1)}$ as stated in (1), including when the number of training samples is small. We introduce a deep architecture that we refer to as Deep Hybrid Network (DHN). It consists of an auto-encoder [13] on the bottom layer and stacked Restricted Boltmann Machines (RBMs) [4] on the top of the first hidden layer. We also introduce a variant of DHN that incorporates both types of sparsity mentioned in Section 2. We refer the second variant as sparsely-connected Deep Hybrid Network (sDHN).

3.1. Deep Hybrid Network

The Deep Hybrid Network (DHN) is proposed in order to obtain two properties: 1) to retain good generalization achieved by training stacked RBMs, and 2) to preserve invariant features over scattered noise achieved by sparsifying $\mathbf{h}^{(1)}$. Considering the fact that stacked RBMs perform well on *in-sample* recognition problem, we encourage the bottom RBM to observe the *denoised* feature $\mathbf{h}^{(1)}$ rather than to observe the data directly.

The sparse features $\mathbf{h}^{(1)}$ is extracted by the auto-encoder. Before we define the auto-encoder, let us define $\mathbf{v} \in \mathbb{R}^{n_v}$ as a visible vector, $\mathbf{h}^{(1)} \in \mathbb{R}^{n_{h^{(1)}}}$ as a feature vector, $\mathbf{W}^{(1)} \in \mathbb{R}^{n_v \times n_{h^{(1)}}}$ as a fully-connected weight matrix, $f_{\text{encoder}} : \mathbb{R}^{n_v} \rightarrow \mathbb{R}^{n_{h^{(1)}}}$ as an encoder, and $f_{\text{decoder}} : \mathbb{R}^{n_{h^{(1)}}} \rightarrow \mathbb{R}^{n_v}$ as a decoder. The auto-encoder used in DHN is defined by

$$f_{\text{encoder}}(\mathbf{v}) = \sigma((\mathbf{W}^{(1)})^\top \mathbf{v} + \mathbf{b}) = \mathbf{h}^{(1)}, \quad (2)$$

$$f_{\text{decoder}}(\mathbf{h}^{(1)}) = \sigma(\mathbf{W}^{(1)}\mathbf{h}^{(1)} + \mathbf{c}), \quad (3)$$

where $\sigma(\cdot)$ is the logistic function, and $\mathbf{b} \in \mathbb{R}^{n_h^{(1)}}$ and $\mathbf{c} \in \mathbb{R}^{n_v}$ are the hidden and visible biases, respectively.

As is typical in deep neural networks, DHN has two training stages: unsupervised greedy layer wise training referred to as pre-training, and supervised fine-tuning. In the pre-training stage, the auto-encoder is optimized by minimizing the cost function

$$l(\mathbf{W}^{(1)}, \mathbf{b}) = \sum_{l=1}^{N} \| f_{\text{decoder}}^{(l)}(\mathbf{h}^{(1)}) - \mathbf{v}^{(l)} \|_2^2 + \beta KL(\hat{\rho}^{(l)} \| \gamma), \quad (4)$$

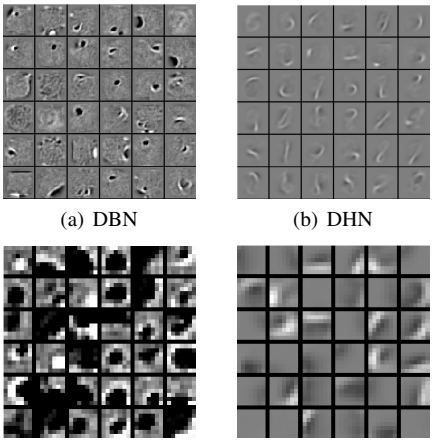
where β is a regularization constant controlling the sparsity, $\hat{\rho}_j = \mathbb{E}[h_j^{(1)}]$ is the average activation of j-th hidden node in $\mathbf{h}^{(1)}$ over all training examples, γ is a uniform vector representing the sparsity target, $KL(\cdot\|\cdot)$ is a variant of Kullback-Leibler divergence defined in the *differentiable sparse coding* work [18], N is the number of training examples.

After pre-training $\mathbf{W}^{(1)}$ and $\mathbf{b}$, each observed instance is encoded into a sparse feature $\mathbf{h}^{(1)}$. From a probabilistic viewpoint, $h_j^{(1)}$ can be considered as $P(h_j^{(1)} = 1|\mathbf{v})$. To provide binary inputs for the stacked RBMs, we sample from $P(h_j^{(1)} = 1|\mathbf{v})$ to obtain a binary value for node $h_j^{(1)}$. The RBMs are pre-trained using k-contrastive divergence (k-CD) learning [23]. After the pre-training stage is finished, we transform DHN into a feed-forward neural network and then fine-tune it using supervised back-propagation.

We expect an additional property produced by sparsifying the features with respect to the advantage of an interpretable dictionary. The property is aimed to retain good generalization performance when using a small number of labeled training examples, as suggested by Coates and Ng [22]. Hence, DHN is also expected to perform well in the recognition of *out-of-sample* data when trained using a small training set.

It is natural to ask why we use the auto-encoder as the bottom building block as sparse features can also be encouraged using RBM-based method, e.g., sparse RBM [20]. However, optimizing the hyperparameters of sparse auto-encoder in (4) such as β, γ is simpler than optimizing the hyperparameters of sparse RBM in terms of obtaining an interpretable dictionary. The sparse RBM naturally has the learning rate and the momentum to be tuned, each of which does not exist in sparse auto-encoder, that makes the optimization harder. Moreover, as simulated by Larochelle et al. [24], auto-encoder learning might be more appropriate than RBM learning in terms of the discovery of good features for a discriminative task. Hence, we suggest that the auto-encoder may lead to more invariant features than the ones discovered by RBM training.

Fig. 1: The MNIST digit images with various types of noise: *clean*, *2-pixel border*, *block*, *impulse-04*, *back-im*



(a) DBN (b) DHN

(c) sDBN (d) sDHN

Fig. 2: The first 36 columns of weights $\mathbf{W}^{(1)}$ after training using 60,000 MNIST examples.

3.2. Sparsely-connected Deep Hybrid Network

Although sparse features in DHN are expected to be invariant, the fully-connected weights specified in the auto-encoder may still cause the features $\mathbf{h}^{(1)}$ to be sensitive to locally occluded image. As described in [10], sparse weights between $\mathbf{v}$ and $\mathbf{h}^{(1)}$ can reduce the effect of noise on local regions. To incorporate both properties of sparse features and sparse weights, we propose a variant of the DHN that we refer to as the sparsely-connected Deep Hybrid Network (sDHN), where the layer connections are similar to sDBN.

To obtain the sparse weights, let us denote a *binary mask* by $\mathbf{M} \in \mathbb{R}^{n_v \times n_{h^{(1)}}}$. Each column of $\mathbf{M}$ represents a connection between each node in $\mathbf{h}^{(1)}$ to an $s \times s$ sub-image in $\mathbf{v}$, where $\mathbf{M}_{ij} = 1$ denotes that a link between v_i and $h_j^{(1)}$ exists. The choice of the $s \times s$ sub-image for each node $h_j^{(1)}$ is random. We then compute the sparse weights $\mathbf{W}_M^{(1)}$ by a *masking* operation

$$\mathbf{W}_M^{(1)} = \mathbf{W}^{(1)} \odot \mathbf{M}, \tag{5}$$

where $\odot$ is the element-wise multiplication. This operation is also applied to make the weight gradient sparse at the learning stage.

Different from sDBN, the features of $\mathbf{h}^{(1)}$ are sparse in sDHN. If $\mathbf{W}_M^{(1)}$ is visualized, the difference between $\mathbf{W}_M^{(1)}$ learned by sDBN and sDHN can be seen from Fig. 2(c)-2(d). The results will be discussed in section 4.3.

Note that we do not employ a special technique called *probabilistic denoising* that can be used at test time [10]. Of course, we could produce more robust model using this technique, but it would increase the computational complexity. In this work, we only focus on the architecture and sparsity with respect to the robustness to noise, while keeping the complexity at inference stage low.

4. EXPERIMENTAL RESULTS

In this section, we investigate the robustness of our proposed methods on handwritten digits compared to other deep networks. More specifically, we evaluated and compared the recognition performance of four deep networks, namely, DBN [4], sDBN [10], DHN, and sDHN, on various types of noise.

4.1. Data Preparation

We used two handwritten digit data sets: MNIST,[1] which consists of 60,000 training images and 10,000 test images of size 28×28, and USPS,[2] which has 7,291 training images and 2,007 test images of size 16×16. Each gray pixel of the images is normalized to a real value of range [0, 1]. We created a rescaled version of the USPS test set used for the cross-domain recognition evaluation (see Section 4.6), which we refer to as r-USPS. Each r-USPS sample is of the size 28×28, the same size as an MNIST sample.

We also created noisy test examples extracted from the MNIST test set. This test set is used to evaluate the performance of the deep networks against noise. The types of noise used in the experiment are *2-pixel border*, block occlusion with random position and size (*block*) in each sample, impulse noise with ratio of 40% (*impulse-04*), and background image (*back-im*) taken randomly from three natural scene patches (see the samples in Fig. 1).

4.2. Training Setup

Each network has the same architecture with three layer of hidden nodes of size 500, 500, 2000, respectively, and 10 target nodes. The size of visible nodes depends on the dimension of the data set (784 for the MNIST and 256 for the USPS). For the sparsely-connected architectures, the connections of size 7×7 were specified by the binary mask $\mathbf{M}$.

All networks were trained by unsupervised greedy layer-wise pre-training and followed by supervised fine-tuning [4]. In every RBM training, we ran 50 epochs of mini-batched learning with learning rate $\alpha = 0.1$, weight decay $\lambda = 10^{-5}$, and 25-CD[3]. In the auto-encoder training, we used L-BFGS back-propagation as the optimizer with sparsity target $\gamma = 0.1$ and regularization constant $\beta = 3$ [25]. Each network was fine-tuned by Conjugate Gradient back-propagation [7]. We ran ten independent experiments for each model to obtain the results in terms of mean and standard deviation.

4.3. Effect of Sparsity

It is useful to study the behavior of the first layer after pre-training using the MNIST training set. We visualize the weight bases (the columns of $\mathbf{W}^{(1)}$) as shown in Fig. 2. We can see that the sparse architectures (DHN and sDHN) have more interpretable weights, which look like digit pen-stroke patterns.

In sDBN and sDHN, the weight representations are more localized in particular parts. This means that each connection between $h_j^{(1)}$ and $\mathbf{v}$ acts as a local filter to map an image patch to a real value. If a 7×7 patch is contaminated by noise, it will disturb only one node in $\mathbf{h}^{(1)}$. Because sDHN produces an interpretable dictionary, it is expected to obtain more invariant node $h_j^{(1)}$ associated with a particular patch than sDBN.

On the other hand, sDBN and sDHN might lose capacity in terms of the number of distinct samples that can be captured. It is based on the claim that the information capacity is proportional to

[1] http://yann.lecun.com/exdb/mnist

[2] http://www-i6.informatik.rwth-aachen.de/ keysers/usps.html

[3] The same number of k of k-CD specified in [10] for a fair comparison

the number of connections in the network [26]. As the connections are more sparse, sDBN and sDHN may fail to store a large number of distinct samples as well as the fully connected networks. In general, it can be expected that the best robustness that a network can achieve decreases with the utilization of its capacity.

4.4. Baseline Performance

We compared the baseline performance of each network on the clean MNIST or USPS data sets. The baseline performance here refers to the recognition of the *in-sample* test set. The comparison is illustrated in Table 1. We can see that the DBN and sDBN outperform our both proposed networks in the case of learning by using the full training set (the first and second row). This suggests that either sparse weights or sparse features do not improve the performance on recognizing the *in-sample* test digits when sufficiently large training examples are available.

In the case of using a smaller training set (third row), the DHN and sDHN, each of which has an interpretable dictionary, perform better than the DBN and sDBN. This is consistent with the hypothesis that the model with an interpretable dictionary obtains better generalization when it is trained using a few labeled examples [22].

Data set (#training)	DBN	sDBN	DHN	sDHN
MNIST (60,000)	**0.96 ± 0.01**	1.11 ± 0.01	1.13 ± 0.09	1.23 ± 0.01
USPS (7,291)	**4.59 ± 0.08**	**4.46 ± 0.13**	4.68 ± 0.00	4.78 ± 0.00
MNIST (1,000)	10.94 ± 0.95	9.22 ± 0.34	**8.19 ± 0.38**	**7.96 ± 0.00**

Table 1: The baseline error rates (%); each deep network was trained and tested using the full set of MNIST. The lowest score (a few have overlapping error confidence intervals) for each setting are written in bold.

4.5. Noisy Test Set Evaluation

We evaluated all the trained networks for the case of *out-of-sample* recognition by testing them on the noisy MNIST test sets. The types of noise used are described in Section 4.1. The complete error rates are shown in Tables 2 and 3.

It is clear that the DBN is unable to retain a good performance for all the noisy test cases. The sDBN performs best in the case of *block* noise but is less robust when dealing with non-local noise (*2-pixel border*, *impulse-04*, and *back-im* noise). In contrast, the DHN and sDHN can cope with non-local noise, especially the impulse and the background image noise. Both also perform better than others in most cases well when the number of training samples is small.

While the sDHN is designed to cope with noise on local image regions, it does not perform as well as sDBN in the recognition of the test set with block occlusion. This performance is likely due to the fact that most of the $\mathbf{h}^{(1)}$ nodes observe the clean patches of the block-occluded images. In this situation, the non-sparse RBM features should provide better performance. In other words, this suggests that sparse features might not helpful for sparsely-connected networks in the recognition of images with mostly clean regions.

Noise	DBN	sDBN	DHN	sDHN
2-pixel border	89.36 ± 0.02	2.55 ± 0.02	2.17 ± 0.05	**1.70 ± 0.02**
block	49.83 ± 0.07	**26.13 ± 0.20**	41.67 ± 0.02	37.80 ± 0.09
impulse-04	88.30 ± 0.02	61.92 ± 0.53	**32.07 ± 0.41**	36.22 ± 0.19
back-im	48.77 ± 0.07	23.78 ± 0.20	**5.38 ± 0.20**	7.46 ± 0.05

Table 2: The error rates (%) on the MNIST test set with various types of noise. Each deep network was trained using 60,000 clean MNIST training examples.

Noise	DBN	sDBN	DHN	sDHN
2-pixel border	89.51 ± 1.36	11.58 ± 0.08	8.57 ± 0.42	**8.13 ± 0.00**
block	67.04 ± 4.34	**28.40 ± 0.80**	39.36 ± 0.30	38.19 ± 0.00
impulse-04	87.26 ± 4.22	56.96 ± 4.72	**26.63 ± 1.02**	**28.18 ± 0.56**
back-im	63.60 ± 8.91	31.38 ± 3.03	**11.06 ± 0.90**	**11.63 ± 0.00**

Table 3: The error rates (%) on the MNIST test set with various types of noise. Each deep network was trained using 1,000 clean MNIST training examples.

4.6. Cross-domain recognition: MNIST vs USPS

We also investigated the cross-domain recognition performance of all networks. This type of evaluation is often conducted to evaluate the *transfer learning* model, i.e., the model is tested on images drawn from different probability distribution from the training images distribution. In this case, we chose the MNIST as the training set and the r-USPS described in Section 4.1 as the test set.

The cross-domain recognition results are shown in Table 4. We found that sDHN performs best for this case. Therefore, it may be useful for solving the transfer learning problem. Furthermore, the DHN and sDHN also perform better than the DBN and sDBN on learning from 1,000 examples, similar outcome to the experiment described in Section 4.4. It again confirms that the models with an interpretable dictionary generalize better than ones without an interpretable dictionary, when the training set is small. In other words, the models might not achieve good results if they fail to obtain an interpretable dictionary learned from a small training set.

#MNIST training	DBN	sDBN	DHN	sDHN
60,000	9.85 ± 0.02	11.21 ± 0.26	10.34 ± 0.26	**8.88 ± 0.19**
1,000	32.21± 0.33	34.18 ± 0.19	29.57 ± 1.54	**24.44 ± 2.05**

Table 4: The error rates (%) of deep networks trained using the MNIST training set, but tested on the r-USPS test set (2,007 images)

5. CONCLUSIONS

In this work, we proposed the Deep Hybrid Network and evaluated its robustness in dealing with *out-of-sample* object recognition. The DHN consists of an auto-encoder as the bottom building block to extract sparse invariant features and stacked RBMs on the top of the auto-encoder. This particular combination encourages the RBMs to always observe *clean* inputs, which has been proven to generalize well as shown in many previous works. Trained on handwritten digit examples without any noise compensation, the DHN obtains significantly better results than the baseline method (DBN) in all *out-of-sample* cases, without losing significant performance in the *in-sample* situation. Compared with the sparse weight network referred to as sDBN, DHN in general performs better than sDBN in the case of non-local noise.

We also combined the DHN with sparse weights to form the sDHN. This combination has improved the performance of the DHN against the *block* noise, but it can not compete with the sDBN on that case. Besides the recognition of block-occluded images, both DHN and sDHN perform better than other networks on learning from a small number of examples. Furthermore, sDHN performs best in the cross-domain recognition case, i.e., the setting of which the training and test images of the same object category are taken from different data sets. From experiments, we confirmed that our sparse feature networks that have an interpretable dictionary perform better than dense feature networks when they are trained from a small number of examples.

6. REFERENCES

[1] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *In Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[2] D.-C. He and L. Wang, "Texture unit, texture spectrum, and texture analysis," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 28, pp. 509–512, 1990.

[3] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 886–893.

[4] G. E. Hinton and S. Osindero, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[5] B. A. Olshausen and D. J. Field, "Sparse coding of sensory inputs," in *Current Opinion in Neurobiology*. Science Direct, 2004, vol. 14, no. 4, pp. 481–487.

[6] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proceedings of the 30th International Conference on Machine Learning*, 2013, pp. 1319–1327.

[7] R. Salakhutdinov and G. E. Hinton, "Deep Boltzmann machines," in *the 12th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009, pp. 448–455.

[8] W. Y. Zhou, S. Zhu, A. Y. Ng, and K. Yu, "Deep learning of invariant features via simulated fixations in video," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 25, 2012, pp. 3212–3220.

[9] L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. L. Seltzer, G. Zweig, X. He, J. Williams, Y. Gong, and A. Acero, "Recent advances in deep learning for speech research at microsoft," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013, pp. 8604–8608.

[10] Y. Tang and C. Eliasmith, "Deep networks for robust visual recognition," in *ICML*, 2010, pp. 1055–1062.

[11] A. Mushtaq and C.-H. Lee, "An MCMC approach to joint estimation of clean speech and noise for robust speech recognition," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2013, pp. 7107–7111.

[12] M. L. Seltzer, D. Yu, and Y. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013, pp. 7398–7402.

[13] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 19, 2007, p. 153.

[14] M. Ranzato, Y.-L. Boureau, and Y. LeCun, "Sparse feature learning for deep belief networks," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 20, 2008, pp. 1185–1192.

[15] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Review*, pp. 129–159, 2001.

[16] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," in *IEEE Transactions on Image Processing*, vol. 15, 2006, pp. 3736–3745.

[17] J. Mairal, M. Elad, and G. Sapiro, "Sparse representation for color image restoration," in *IEEE Transactions on Image Processing*, vol. 17, no. 1, 2008, pp. 53–69.

[18] D. M. Bradley and J. A. Bagnell, "Differentiable sparse coding," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 21, 2009, pp. 113–120.

[19] J. Mairal, F. Bach, and J. Ponce, "Task-driven dictionary learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, 2012.

[20] H. Lee, C. Ekanadham, and A. Y. Ng, "Sparse deep belief net model for visual area V2," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 20, 2007, pp. 873–880.

[21] J. Goodfellow, Q. V. Le, A. M. Saxe, H. Lee, and A. Y. Ng, "Measuring invariance in deep networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2009, vol. 22, pp. 646–654.

[22] A. Coates and A. Y. Ng, "The importance of encoding versus training with sparse coding and vector quantization," in *Proceedings of the 28th International Conference on Machine Learning*, 2011, pp. 912–928.

[23] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Computation*, vol. 14, pp. 1771–1800, 2002.

[24] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin, "Exploring strategies for training deep neural networks," *Journal of Machine Learning Research*, vol. 1, pp. 1–40, 2009.

[25] Q. V. Le, J. Ngiam, A. C. A. Lahiri, B. Prochnow, and A. Y. Ng, "On optimization methods for deep learning," in *Proceedings of the 28th International Conference on Machine Learning (ICML)*, L. Getoor and T. Scheffer, Eds., 2011, pp. 265–272.

[26] E. Fiesler, H. J. Caulfield, and A. Choudry, "Some theoretical upperbounds on the capacity of neural networks," in *Proceedings of the First Workshop on Neural Networks*, vol. 2, 1990, pp. 51–58.