Robust and Reliable Feature Extractor Training by Using Unsupervised Pre-training with Self-Organization Map

You-Min Lee and Jong-Hwan Kim

Department of Electrical Engineering, KAIST
291 Daehak-ro, Yuseong-gu, Daejeon 305-701, Republic of Korea
{ymlee, johkim} @ rit.kaist.ac.kr

Abstract. Recent research has shown that deep neural network is very powerful for object recognition task. However, training the deep neural network with more than two hidden layers is not easy even now because of regularization problem. To overcome such a regularization problem, some techniques like dropout and de-noising were developed. The philosophy behind de-noising is to extract more robust features from the training data. For that purpose, randomly corrupted input data are used for training an auto-encoder or Restricted Boltzmann machine (RBM). In this paper, we propose unsupervised pre-training with a Self-Organization Map (SOM) to increase robustness and reliability of feature extraction. The basic idea is that instead of random corruption, our proposed algorithm works as a feature extractor so that corrupted input maintains the main skeleton or structure of original data. As a result, our proposed algorithm can extract more robust features related to input data.

1. Introduction

In the image recognition task, deep neural network has shown good performance than shallow one, since the deep network can represent hierarchical feature of objects. However, until early 2000, there has not been an efficient training algorithm for deep architecture. The conventional neural network algorithm, the supervised error back-propagation (EBP) [4] does not suffice for training deep architecture because stochastic gradient decent (SGD) effect would be very weak at frontal hidden layer and the overall complexity of deep neural network is so high. However, after greedy layer wise pre-training was introduced, training the deep neural network with more robust and efficient could be possible [8][9]. Among many greedy layer wise methods, Auto-encoder (AE) has been a popular one because it can be implemented easily and training time is shorter than another famous algorithm, Restricted Boltzmann machine (RBM) [7]. Also, there are variants of the AE that were proved better than the conventional one. De-nosing Auto-encoder (dAE) was proposed to extract more robust features than the conventional AE [6]. The dAE uses corrupted data for auto-encoder input instead of original data; therefore, the dAE can detect more various and robust features than the conventional AE. In the dAE, the corruption method relies on randomness such that, for example, it randomly discards 30% pixels on the image. Therefore, it is

expect that the obtained corrupted input by using the dAE may lose the main skeleton or structure of the original data. Therefore, the dAE may cause neural network to learn features less related to given input data.
To deal with such a weak point, this paper proposes to use the Self-Organization Map (SOM) [3] for corrupt input data to pre-train the neural network. The SOM has been used for classifying or clustering unlabeled data in an unsupervised manner. However, in this paper, the training algorithm does not use the SOM for classifying. Instead, the SOM is used for detecting the main skeleton or structure of input data. If the greed size of the SOM is smaller than the number of data pixels, the SOM can corrupt input data without much loss of its main structure. For the dAE, if input destruction rate goes up more than 80%, input data lose most of the main information and as a result, it cannot extract good features. However, for the AE with SOM, even if the greed size is 3x3, which is only 5% of average number of input data pixels, it can detect meaningful features.
The paper is organized as follows. Section 2 describes the basic theory and related works. In Section 3, the performance of the proposed AE with the SOM is demonstrated through the comparison with the original AE and the dAE. Section 4 presents the experimental results and concluding remarks follow in Section 5.

2. Related Work

2.1 The Auto-encoder and Stacked Auto-encoder

To train the deep architecture efficiently, a generative model has been developed for many years. Two popular algorithms for the generative model are Restricted Boltzmann machine (RBM) and Auto-encoder (AE). The structure of AE is one hidden layer neural network and can be trained by error back-propagation (EBP) algorithm. As the purpose of using AE is to reconstruct input data at the output layer, the desired target output of the AE should be input data itself. Since there is one hidden layer between input and output layers, we can regard weights located between input and hidden layers as an encoder and weights located between hidden and output layers as a decoder. The EBP learning rule is applied to optimize weights in auto-encoder.

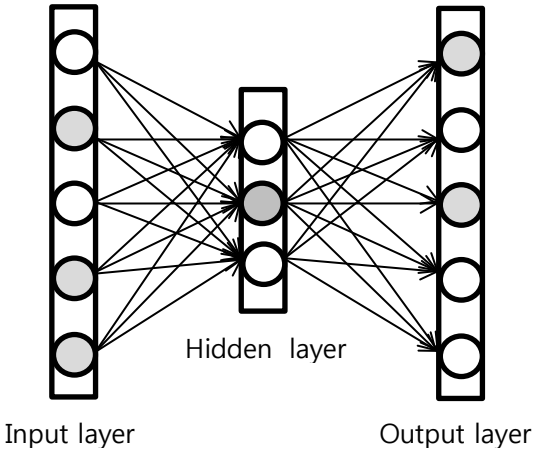


Fig. 1. The architecture of the auto-encoder.

Through the forward propagation, the network output is obtained as

$$y = f(W_2 \cdot f(W_1 \cdot x + b_1) + b_2). \quad (1)$$

where W_1 and W_2 are weight vectors located between input and hidden layers, and hidden and output layers, respectively. Also, b_1 and b_2 are bias terms. The function f is a neural activation function. Neural activation functions mainly used are sigmoid and linear rectified unit (ReLU). Since the neural network output should be input data itself, the desired target output d is the same as x. Therefore, the training rule is to minimize reconstruction error $E = \frac{1}{2}\|x - y\|^2$ and find parameters to minimize a reconstruction error and it can be represented as.

$$W^*, b^* = \arg min_{W,b} E(x, y). \quad (2)$$

It should be noted that training the AE neural network can be regarded as increasing the conditional probability of getting x at the output layer when the input data is x, which is $P(x|x)$.

Since the number of neurons of hidden layer is smaller than that of input and output neurons, the hidden layer acts as a kernel, which can extract informative features of data. However, previous research showed that even if the number of hidden nodes is larger than that of input and output nodes, network can still extract good features [1].

To include auto-encoder training rule in the deep neural network, the greedy layer wise training method can be used. The training step is as follow [6][10]:

1) Train the auto-encoder that uses original input data as its input. And then, use weights located between auto-encoder input layer and hidden layer as the first layer of the deep neural network.

2) Using the output of the first hidden node of the deep neural network as input, train the second auto-encoder. And then, use weight located between auto-encoder input layer and hidden layer as the second layer of the deep neural network.

3) Using the output of the k^{th} hidden node of the deep neural network as input, train the $(k+1)^{th}$ auto-encoder. And then, use weights located between auto-encoder input layer and hidden layer as the $(k+1)^{th}$ layer of the deep neural network.

4) Repeat 3) until all the deep neural network layers are trained.

5) For classification, insert single regression or the softmax layer on the output node and do the fine-tuning.

2.2 The De-noising Auto-encoder

To extract robust features, de-noising auto-encoder (dAE) was developed [6]. The difference between the conventional AE and dAE is that the AE uses original input

itself as auto-encoder input and the dAE uses corrupted version of original input as auto-encoder input. Because of corruption, the dAE tries to recover original data from distorted data and such a constraint forces the dAE to extract more robust features. There are various corruption methods. The most popular one is to use randomness; set input pixel value to 0 at some probability. Depending on corruption rate, the features of dAE can be different. The dAE learning rule is to minimize the reconstruction error $\mathrm{E} = \frac{1}{2}\|x - y\|^2$. The dAE output y is represented as:

$$\mathrm{y} = \mathrm{f}(W_2 \cdot \mathrm{f}(W_1 \cdot \tilde{x} + b_1) + b_2) \tag{3}$$

where $\tilde{x}$, W_* and b_* are corrupted input, weight vectors and bias terms, respectively. The dAE architecture is shown in Fig. 2.

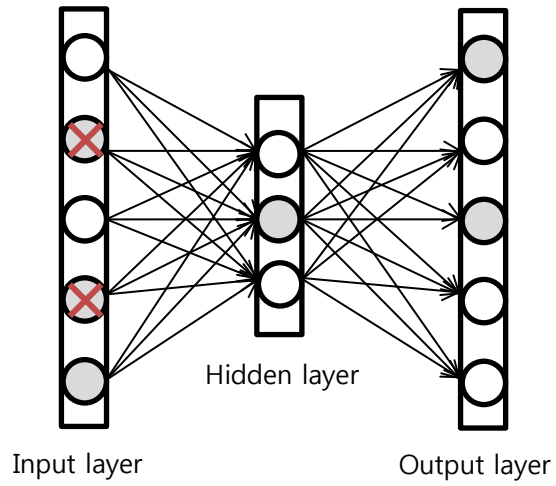


Fig. 2. The architecture of the De-noising auto-encoder.

2.3 Self-organization map

The Self-organization map (SOM) is unsupervised learning based neural network, which has been used for data clustering and classification [3]. The SOM learning rule is to update weight that is closer to input x than other weights. Therefore, winner takes all. Weights are updated as follow:

$$W_{k+1} = W_k + \eta \cdot h_{ij} \cdot (x - W_k) \tag{4}$$

where η is a learning-rate and h_{ik} is a neighborhood equation, respectively. The parameter h_{ik} forces weight that is located near the winner weight to update strongly and forces weight, which is located far from the winner weight to update weakly. The parameter h_{ik} is represented as

$$h_{ij} = \exp(-\frac{{d_{ij}}^2}{2\sigma^2}) \tag{5}$$

where parameter d_{ij} represents the distance between two weights and the

parameter σ is a standard deviation. After training, the SOM network learns the distribution of input data; such a property can be used to corrupt input data without much loss of the important information.

3. Models

3.1 Architecture

The Self-organization map (SOM) should be included between original input and the auto-encoder (AE) input layers to corrupt input data. The AE with SOM architecture is shown in Fig. 3. The sigma and eta value used in the SOM was 200 and 0.1, respectively.

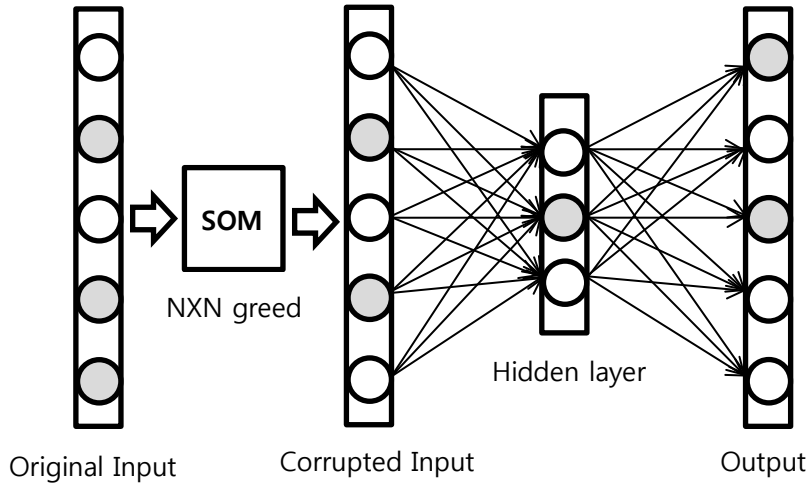
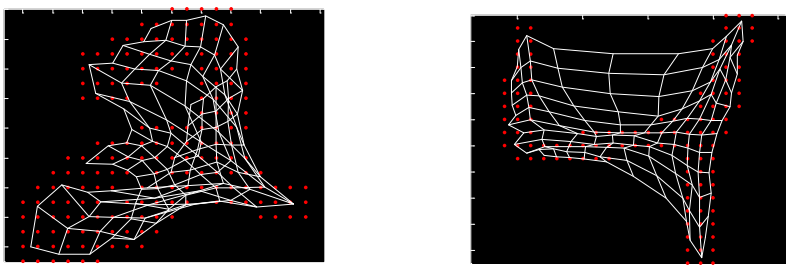


Fig. 3. The AE with SOM architecture.

By adjusting the SOM greed size, we can change the corruption rate. If the greed size is small, the corruption rate is big. On the other hand, if the greed size is big, the corruption rate is small. Before corrupt the input data, the SOM constructs a self-organization map. Here are some examples. The input grey image is a 28x28 size digit from the Mixed National Institute of Standards and Technology (MNIST).



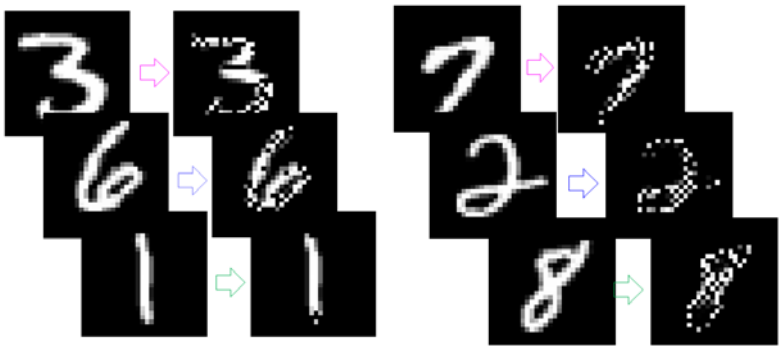
(a) Number 2. (b) Number 4.

Fig. 4. (a) shows the SOM result when input data is number 2 and (b) shows the SOM result when input data is 4.

In Fig. 4, red points represent pixel location of the original image and white map shows self-organized map after training the SOM. As shown in Fig. 4, the SOM greed is well distributed on the original input data pixels. However, some points in the greed are out of the input data region. Therefore, it is required to remove those pixels. The removing is implemented as follow: after training the SOM, the white pixels located out of the red pixel region are moved to the nearest red pixel point to reconstruct the valid input format for the AE.

3.2 Input Corruption Using the SOM

Before the experiments, the comparison between the dAE corrupted input and the SOM corrupted input is needed. SOM corrupted input examples are provided in this section.

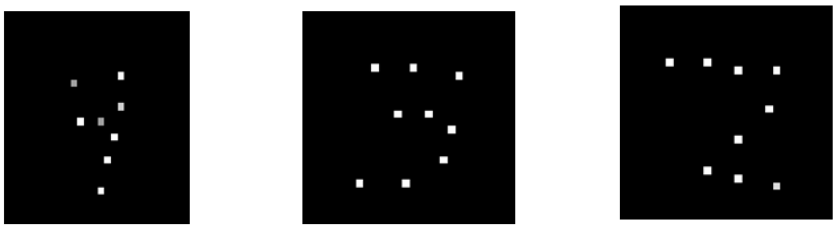


(a) Greed size 11x11 (b) Greed size 8x8

Fig. 5. (a) The SOM output when the greed size is 11x11. (b) The SOM output when the greed size is 8x8.

Fig. 6. The dAE corrupted input result according to different corruption rate.

In case that the dAE corrupted rate is small, it is hard to distinguish the difference between the dAE corrupted input and the SOM corrupted input with a large greed size. However, if the corruption rate is a large, there is a clear difference between the dAE corrupted input and the SOM corrupted input with a small greed size.



(a) The SOM corrupted input when the greed size is 3x3.

(b) The dAE corrupted input when the corruption rate is 90%.

Fig. 7. The comparison between the dAE corrupted input and the SOM corrupted input. Original input numbers are 4, 3 and 2, respectively. We can see that even though the SOM corrupted input has a few pixel points compared to the dAE corrupted input, it maintains the main skeleton of input data. For the dAE corrupted input, it is almost impossible to answer what the original number is.

4. Experiments

Experiments were conducted using a deep neural network with three hidden layer, each layer has 1000 nodes and the number of input nodes is 28x28. At the output layer, 10 nodes softmax classifier was used for digits classification. To compare the accuracy of the AE, dAE and AE with SOM, we trained weights located between input and first hidden layers with different methods. For the conventional AE, no input distortion was used to train auto-encoder. For the dAE, various distortion rates were tested for the first hidden layer and fixed distortion rate for second hidden layer and third one to 20% and 30%, respectively. For the AE with SOM, various greed sizes were tested for the first hidden layer and fixed distortion rate for second hidden layer and the third one to 20% and 30%, respectively. Because of limited time, we use only 10,000 training data. The error rate was obtained from 10,000 test data. The number of pre-training epoch was 12 for each layer and 20 epoch was used for the fine tuning. Learning rate were 0.1 and 0.3 for the pre-training and fine-tuning, respectively. At the end of experiments using 60,000 training data, we compared the accuracy between the dAE and the AE with SOM. In these experiments, parameters like the number of nodes and learning rate are chosen based on trial and error.

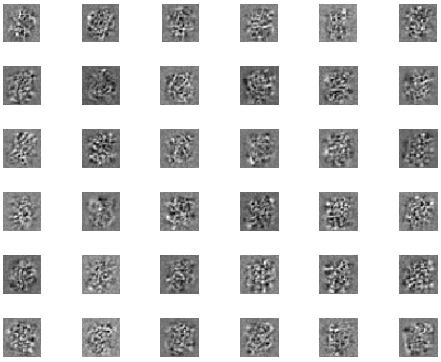
The programing environment was MATLAB 2013a and to increase learning speed, GPU was used. By using GPU operator, the program learning time was three times faster than that without GPU operator. For each task, elapsed time was recorded. However, different computers were used to test the dAE and the AE with SOM. For the dAE, 3.2GH CPU, GeForce 560, and 8GH RAM computer was used. For the SOM, 2.8GH CPU, GeForce 440 and 4GH RAM computer was used. Therefore, the SOM elapsed time is much longer than the dAE. The reason using two different computers is because of very long learning time.

Method (10,000 training data)	Before Fine tuning	After Fine tuning	Elapsed time(sec)
AE	7.66%	3.97%	5266
dAE(**10%**)	4.44%	2.96%	4909
dAE(**30%**)	4.44%	2.73%	4878
dAE(**50%**)	4.44%	3.09%	4984
dAE(**80%**)	5.27%	3.9%	4998
AE with SOM(13X13) Distortion rate **10%**	4.72%	3.15%	7800
AE with SOM(10X10) Distortion rate **35%**	4.5%	3.02%	7743
AE with SOM (7X7) Distortion rate **70%**	4.35%	2.93%	7723
AE with SOM (3X3) Distortion rate **95%**	5.09%	4%	7800

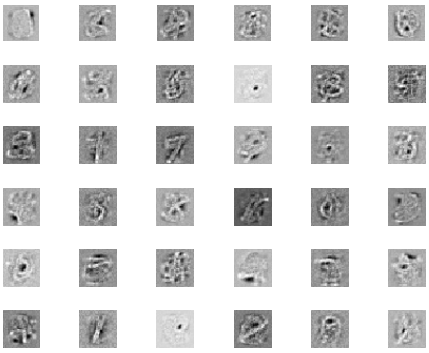
Table 1. The error rate was checked at before fine tuning and after fine tuning. Elapsed time was checked for the AE and the dAE using 3.2GH CPU, GeForce 560 and 8G RAM computer. Elapsed time was checked for the AE with SOM using 2.8GH CPU, GeForce 440 and 4G RAM computer. At glance, the learning time of the AE with SOM looks like taking very longer than that of the dAE. However actually there is very little difference between them.

As shown table 1, the best error rate for the dAE and the AE with SOM was better than the conventional AE. Which justifies our hypothesis; the dAE and the AE with SOM can extract more robust features than the conventional AE. For the dAE and the SOM, if the input data corruption rate was small, there was not a clear visual difference between the dAE and the SOM features obtained. However, if the input data corruption rate was high, the SOM showed much better performance than the dAE. These results imply that when corrupted data is used for an auto-encoder input, the main skeleton of input data should be maintained to extract meaningful features. When distortion rate was small, the dAE as well as the SOM corrupted input data can maintain their main structure; therefore, there was not a big difference. Even though their best error rate was similar, there were differences between features learned. When we saw the dAE features, most of them look like local features. For the SOM, however, their features look like the mixture of local filter and the conventional auto-encoder features. At this moment, we cannot interpret what do that means, but it may be worth thinking about it. In terms of learning time, the AE with SOM takes longer than that of the dAE because there is a SOM network in the input to first hidden layer. However, the elapsed time is almost same. For 60,000 training data, the SOM network tasks only about 5 minute. This is very small compared to the total elapsed time which is for 60,000 training cases, about 6 hours.

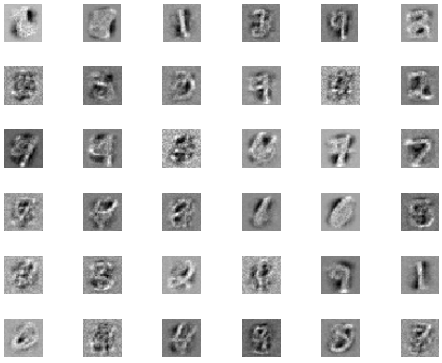
Fig. 8 shows each feature obtained at the first hidden layer with different methods.



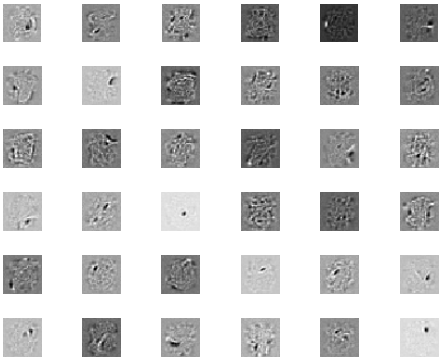
(a) The features learned from the conventional auto-encoder.



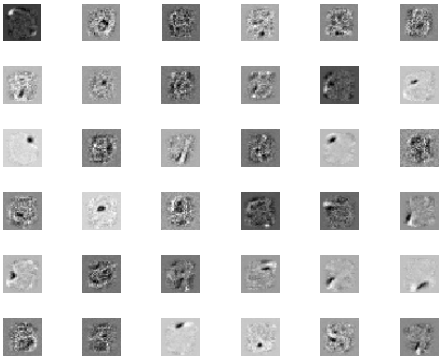
(b) The features learned from the auto-encoder with 30% distortion.



(c) The features learned from the auto-encoder with 80% distortion.



(d) The features learned from the auto-encoder with 10% distortion.



(e) The features learned from the auto-encoder with SOM greed size of 7x7.

Fig. 8. Obtained features from the different methods.

Table 2 shows the best performance achieved by training 60,000 data. Before fine tuning, the AE with SOM's error rate was 2.99% that is almost 0.5% lower than that of the dAE with any distortion rate. After fine tuning, the dAE error rate was 1.43% that is the best of all experiments in this paper.

Method (60,000 training data)	Before Fine tuning	After Fine tuning	Elapsed time(sec)
dAE(**10%**)	3.54%	**1.43%**	22,600
AE with SOM(7X7) Distortion rate **70%**	**2.99%**	1.61%	23,000

Table 2. The error rate test when the number of training data is 60,000. When test the dAE, distortion rate for each layer was well tuned according to other researcher's paper. The greed size and the distortion rate for each layer were randomly chosen for the AE with SOM.

5. Conclusion

To train the deep neural network, we need good representations for input data. If we can insert good representations about the world in the deep neural network, we may not have to worry about pre-training the network and fine-tuning procedure. To obtain robust features that represent the world more efficiently, a generative model was proposed. Also, to reduce the regularization problem, some techniques such as the de-noising auto-encoder and the dropout were proposed. The de-noising auto-encoder shows how we can get good representations related to the environment data. However, the random method may not guarantee reliable performance. The AE with SOM concept was started with the thinking “If the data’s main skeleton is maintained, human can recover original data even if the original data was largely corrupted.” With this in mind, this paper tried to verify that 1) The AE with SOM shows reliable performance than the dAE in terms of error rate standard deviation. 2) The AE with SOM’s performance does not much depend on its distortion rate. For 2), the AE with SOM shows that even for high distortion, it can extract meaningful features. Therefore, we can justify that distorted auto-encoder input data should maintain the main skeleton or structure. For 1), we need more time for experiments. This remains a topic for future research.

Ackowledgment

This work was supported by the Technology Innovation Program, 10045252, Development of robot task intelligence technology that can perform task more than 80% in inexperience situation through autonomous knowledge acquisition and adaptational knowledge application, funded by the Ministry of Trade, industry & Energy (MOTIE, Korea).

This work was also supported by the Software Computing Technology Development Program, 14-824-09-012, Technology Development of Virtual Creatures with Digital Emotional DNA of Users, funded by the Ministry of Science, ICT and Future Planning.

Reference

[1] Bengio, Yoshua. "Learning deep architectures for AI." *Foundations and trends® in Machine Learning* 2.1 (2009): 1-127.

[2] Aarts, Emile HL, and Jan HM Korst. "Boltzmann machines and their applications." *PARLE Parallel Architectures and Languages Europe*. Springer Berlin Heidelberg, 1987.

[3] Kohonen, Teuvo. "The self-organizing map." *Proceedings of the IEEE* 78.9 (1990): 1464-1480.

[4] Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors." *Cognitive modeling* (1988).

[5] Rojas, Raúl. "Unsupervised Learning and Clustering Algorithms." *Neural Networks*. Springer Berlin Heidelberg, 1996. 99-121.

[6] Vincent, Pascal, et al. "Extracting and composing robust features with denoising

autoencoders." *Proceedings of the 25th international conference on Machine learning*. ACM, 2008.

[7] Hinton, Geoffrey. "A practical guide to training restricted Boltzmann machines."*Momentum* 9.1 (2010): 926.

[8] Le, Quoc V. "Building high-level features using large scale unsupervised learning." *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013.

[9] Lee, Honglak, et al. "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations." Proceedings of the 26th Annual International Conference on Machine Learning. ACM, 2009.

[10] Hinton, Geoffrey E. "Deep belief networks." *Scholarpedia* 4.5 (2009): 5947.