Broad Autoencoder Features Learning for Classification Problem

Ting Wang, South China University of Technology, Guangzhou, China

Wing W. Y. Ng, South China University of Technology, Guangzhou, China

Wendi Li, South China University of Technology, Guangzhou, China

Sam Kwong, Hong Kong City University, Kowloon, Hong Kong

ABSTRACT

Activation functions such as tanh and sigmoid functions are widely used in deep neural networks (DNNs) and pattern classification problems. To take advantage of different activation functions, this work proposes the broad autoencoder features (BAF). The BAF consists of four parallel-connected stacked autoencoders (SAEs), and each of them uses a different activation function, including sigmoid, tanh, relu, and softplus. The final learned features can merge by various nonlinear mappings from original input features with such a broad setting. It not only helps to excavate more information from the original input features through utilizing different activation functions, but also provides information diversity and increases the number of input nodes for classifier by parallel-connected strategy. Experimental results show that the BAF yields better-learned features and classification performances.

KEYWORDS

Feature Learning, Pattern Classification, Stacked Autoencoders

1. INTRODUCTION

With the quick advancement and deployment of information technologies, there are gigantic volumes of information in various organizations accessible on the Internet, for example, video, image, and medical data. The need of mining helpful data from these huge information poses a great challenge to the AI people group. Typical AI techniques requiring hand-crafted features cannot discover hidden information from data and may experience the ill effects of either data misfortune or overfitting. Conversely, DNNs have been effectively applied in a wide range of AI applications and delivered incredible outcomes in computational intelligence, e.g., video processing, image classification, speech recognition, and computer visual recognition.

Until recently, machine learning techniques can partition into generative and discriminative strategies. At present, the most commonly utilized DNN are generative models, e.g., the Deep Belief Networks (Hinton et al., 2006), the Restricted Boltzmann Machines (Salakhutdinov et al., 2007), and the Deep Boltzmann Machines (Salakhutdinov et al., 2009). These techniques prepared the log-probability gradient prepared to utilize MCMC-based strategies that turn out to be progressively uncertain as preparing advances. It is because examples from the Markov Chains cannot blend between

DOI: 10.4018/IJCINI.20211001.0a23

This article published as an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>) which permits unrestricted use, distribution, and production in any medium, provided the author of the original work and original publication source are properly credited.

models sufficiently quickly. Moreover, generative models, e.g. the Autoencoder (AE) (Bengio et al., 2009), the Variational Autoencoder (Kingma and Welling, 2014; Mescheder et al., 2017; Tan et al., 2018), and the Important Weighted Autoencoders (Burda et al., 2016) have created to utilize direct back-proliferation for preparing and maintaining a strategic distance from challenges yielded by the MCMC preparation. Each of these strategies considered as the projection that yields a considerable classification result by anticipating tests from the original feature space into a projected space with a better class-separability for pattern classification problems (Wasikowski et al., 2010). Among them, the AE (Bengio et al., 2009) is an unsupervised feature learning technique that aims to recover the representation to be roughly equivalent to the original sources of the inputs. The number of hidden units is normally bigger than the number of feature dimensions for feature representation learning. The projection at the hidden layer of the AE yields a helpful representation of the original sources of inputs (Bengio et al., 2009).

In practice, the original data may be missing or obscured. For example, the blocked part of an object in an image or the missing word in text data is usually missing or obscured. Therefore, the Dual Autoencoders Features method (DAF) (Ng et al., 2016) combines two activation functions to deal with this problem. With different activation functions, the DAF uses two Stacked AEs (SAE) to learn features by extracting different characteristics from the training data. However, the DAF only uses *sigmoid* and *tanh* functions, leaving other activation functions aside such as *relu* and *softplus*. It is better to use *sigmoid* when the characteristic difference is more complicated or the difference is not particularly large. But it suffers from the vanishing gradient problem during the backpropagation which makes it difficult to complete the training of a deep neural network. The effect of using *tanh* is incredibly good when the characteristic difference is obvious and the characteristic effect will be continuously expanded during the cycle. Moreover, the convergence speed of SGD obtained by using *relu* will be much faster than *sigmoid/tanh* while *softplus* can be regarded as the smoothed version of *relu*. These motivate us to utilize advantages of different activation functions, and we propose the Broad Autoencoders Features (BAF) extend the superiority of the DAF. There are four SAEs with different activation functions used to learn the mapping from the input space to the representation space to capture different characteristics in the original data. The learned features are parallel concentrated to form the BAF that provides information diversity and increases the number of input nodes for Softmax classifier. Different activation functions have different advantages and disadvantages, the BAF does not automatically choose the activation function to optimize the model performance.

The paper is organized as the following: Section II gives a brief review of the related works. Section III proposes the BAF. Sections IV and V give experimental results and the conclusions, respectively.

2. RELATED WORK

The Autoencoder (AE) is a type of neural network that can use to dimensionality reduction (Bengio et al., 2009). The AE learns low-dimensional features of the original inputs by unsupervised learning (Baldi, 2012). The representation capability of learned features can be further improved by a deep architecture obtained by stacking several AEs. In general, an AE consists of an input layer, an encoding layer, and a decoding layer. The input data x is mapped by a deterministic mapping to a hidden representation $f(x)$ at the encoding layer:

$$f(x) = S_e(W_e x + b_e) \quad (1)$$

where $S_e(\cdot)$, b_e , and W_e denote the activation function of the encoding layer, the bias vector, and the weight matrix, respectively. The decoding layer maps $f(x)$ onto a reconstruction $g(f(x))$:

$$g(f(x)) = S_d(W_d f(x) + b_d) \quad (2)$$

where $S_d(\cdot)$, b_d , and W_d denote the activation function of the decoding layer, the bias vector, and the weight matrix, respectively. The AE's goal is to minimize the reconstruction error between inputs x and outputs $g(f(x))$, which is done by finding a set of optimal parameters $\theta = \{W_e, b_e, W_d, b_d\}$. Hence, the optimization process of an AE is defined as follows:

$$\arg_{\theta}^{\min} \frac{1}{2} \sum_{i=1}^N x^{(i)} - g(f(x^{(i)}))_2 \quad (3)$$

AE's feature learning has been widely used in different application fields, such as very deep AE for content-based image retrieval by learned features from images (Krizhevsky & Hinton, 2011). Heydarzadeh *et al.* propose a robust and accurate algorithm for in-bed posture classification using AE.

It is regularly accepted that the number of hidden neurons is smaller than the number of input features so that the AE performs the compression process from the original inputs to the hidden layers (encoding). If a random Gaussian sequence is fed to AE as the input, the meaningful features will fail to extract from the original inputs. Under these circumstances, a large number of hidden neurons are needed for the AE (the number may be larger than the number of input neurons) to achieve feature extraction. It may cause failure to compress the input features. In another aspect, the AE can excavate correlated relationships among input features. Therefore, the Sparse Autoencoder (SAE) (Ng, 2011) adds a sparsity constraint to hidden neurons and discovers the latent structured data. SAE has been applied to detect human body in depth images (Su *et al.*, 2015), obtain learning-based representation (Tang *et al.*, 2016; Wong *et al.*, 2018), and achieve soft sensor modeling (Yuan *et al.*, 2018).

The AE is pre-trained to determine an initial value of the weight parameters W ahead of the training process. But AE gets easily overfitted due to a series of problems in the initial model. For example, the noises in the training data, the large scale of the training data, and the complexity of the AE model. To prevent the overfitting, the input data (input layer of the AE) is often added with noise, so that the learned W can enhance the AE model's generalization capability. The Denoising Autoencoder (Vincent *et al.*, 2010) (DAE) extends the classical AE that learns a more stability projection towards the infection data. The input data is randomly dropped in some input dimensions of the randomly selected training samples to improve the AE's robustness (Vincent *et al.*, 2010). The feature learning process with the Stacked Denoising Autoencoder (SDAE) has been widely used in pose-invariant speech recognition (Weninger *et al.*, 2014) and face recognition (Kang *et al.*, 2013). There are other applications on the DAE such as electrocardiogram active classification (Al Rahhal *et al.*, 2016) and saliency detection (Han *et al.*, 2016). Jiang *et al.* propose a novel feature representation learning method for wind turbine gearbox fault diagnosis which uses the stacked multilevel denoising autoencoders.

A good feature representation should be both a good reconstruction of the input data and robust to partly disturbance of the inputs. The traditional AE, the stacked AE, and the sparse AE meet the first standard while the DAE and the SDAE are consistent with the second standard. The robustness is more important in many cases. To learn robust features, the AE should be robust to small perturbation added to the training samples. Rifai *et al.* propose Contractive Autoencoder (CAE) which trains AE by minimizing the reconstruction error plus a new penalty. The Frobenius norm for the inputs' Jacobian matrix is computed as the penalty term. Moreover, a method of using stacked CAE is applied for hierarchical feature extraction (Masci *et al.*, 2011).

3. BROAD AUTOENCODER FEATURES

The Broad Autoencoders Features (BAF) combines feature learned from four stacked autoencoders (SAEs) which each uses a different activation function. The four activation functions are: $relu(x) = 0$ for $x < 0$ and x otherwise, $sigmoid(x) = 1 / (1 + e^x)$, $tanh(x) = 2 / (1 + e^{-2x})$, and $softplus(x) = \log_e(1 + e^x)$. Figure 1 shows the workflow of the BAF. The greedy layer-wise training algorithm without the fine-tuning step is applied to train each 2-layer SAE with different activation function independently.

As shown in Figure 1, each of the 2-layer SAEs consists of two AEs. In each, the first AE learns to encode inputs using the encoding layer by minimizing the reconstruction error between outputs of its decoding layer and the original inputs from the training dataset. Then the second AE uses the outputs from the encoding layer of the first AE as input to learn the encoding via a minimization of the reconstruction error between the outputs of the encoding layer of the first AE and the decoding layer of the second AE. The outputs of the encoding layer of the second AE are used as the learned features. Such that, the four 2-layer SAEs using different activation functions learn four different sets of learned features which are then concatenated to form the BAF. Figure 2 shows the detailed algorithm of the feature learning of the BAF.

When a sample arrives, it is encoded by using encoding layers of AEs in the *sigmoid* SAE, the *tanh* SAE, the *relu* SAE, and the *softplus* SAE, respectively, to form four sets of learned features. Then, these four sets of learned features are concatenated to form the BAF for this sample. Figure 3 shows the detailed algorithm of the feature encoding procedures of the BAF.

3.1. Training of the BAF

At the beginning of training, all weights are randomly initialized, and biases are set to zeros. Then, weights are updated using the standard backpropagation algorithm. Samples are then fed to the AE in the forward pass to generate output values of the decoding (i.e. output) layer. These values are then used to compute the gradients of both weights and the bias for updating in the decoding layer as follows:

$$\frac{\partial J}{\partial W_d} = \delta_d \left(g(f(x)) \right)^T \quad (4)$$

$$\frac{\partial J}{\partial b_d} = \delta_d \quad (5)$$

where $\delta_d = \partial 0.5 \times (g(f(x)) - x)^2 / \partial W_d f(x) + b_d$. The gradients for both weights and the bias in the encoding layer are computed as follows:

$$\frac{\partial J}{\partial W_e} = \delta_e \left(f(x) \right)^T \quad (6)$$

Figure 1. Workflow of the BAF

$$\frac{\partial J}{\partial b_e} = \delta_e \quad (7)$$

where $\delta_e = W_e \delta_d f'$ and f' denotes the partial derivative of $f(x)$. Weights and biases of AEs in the BAF are updated repeatedly until converge using (4) – (7) by the Limited-memory BFGS (L-BFGS) (Byrd et al., 1995) method.

3.2. Computational Complexity of the BAF

Give N training samples and m hidden neurons in each AE of the BAF. The computation of gradients for both weights and bias need to perform matrix multiplications which has a complexity of $O(m^d)$ where d is a constant varying by different matrix multiplication algorithms. Therefore, the computational complexity of the BAF is $O(4m^d) = O(m^d)$.

4. EXPERIMENTAL RESULTS

In experiments, we use four benchmark datasets to compare the performance of the BAF, DAF, and AFs. Each AF is a BAF with only one type of activation function. For instance, the Sigmoid-AF uses *sigmoid* function for all SAEs. Four benchmark datasets include the MNIST dataset, the Fashion_MNIST dataset, the COIL20 dataset, and the ORL dataset. There are 60,000 grayscale handwritten

Figure 2. The algorithm of the feature learning of the BAF

Algorithm 1 Feature learning of the BAF

Require: $n \times N$ feature matrix D , where n and N denote the numbers of features and samples, respectively. m_j denotes the number of neurons on the encoding layer of the j^{th} AE, respectively.

Ensure: Four SAEs with the *sigmoid*, the *tanh*, the *relu* and the *softplus* activation functions, respectively.

- 1: Scale each input feature to the range of $[0, 1]$.
 - 2: Set $\varphi = D$, and $m_0 = n$.
 - 3: Set $F = \{\text{sigmoid}, \text{tanh}, \text{relu}, \text{softplus}\}$.
 - 4: For $k=1$ to 4 do
 - For $j=1$ to 4 do
 - 3.1: Train an AE with the $F\{k\}$ activation function using φ and (6) with m_j neurons on the encoding layer and $m_{(j-1)}$ neurons on both the input and the decoding layer.
 - 3.2: Compute the outputs of the encoding layer for all training sample to form a $m_j \times N$ matrix H_j .
 - 3.3: Set $\varphi = H_j$.
 - End for
 - 5: End for
-

Figure 3. The algorithm of the feature encoding of the BAF

Algorithm 2 Feature encoding of the BAF

Require: $n \times N_X$ feature matrix X , where n and N_X denote the numbers of features and samples.

Ensure: $4m_l \times N_X$ learned feature matrix H_{BAF} .

- 1: Four SAEs corresponding to the *sigmoid*, the *tanh*, the *relu* and the *softplus* activation functions are obtained from Algorithm 1 respectively.
 - 2: Compute values of the encoding layer for each SAE using X to form the feature matrix H_{sigmoid} , H_{tanh} , H_{relu} and H_{softplus} , respectively.
 - 3: Concatenate H_{sigmoid} , H_{tanh} , H_{relu} and H_{softplus} by rows to get the learned feature matrix H_{BAF} .
-

digital images in MNIST's training set, and 10,000 images in its testing set. Each image is a black-and-white image which ranges from 0 to 9. In Fashion_MNIST, there are also 60,000 training images and 10,000 test images, which are in 10 fashion categories. Each image is a 28×28 grayscale image. The dataset can be regarded as an evolutionary version of the MNIST dataset. The category labels are T-shirts/ blouses, pants, pullovers, dresses, coats, sandals, shirts, canvas shoes, packages, short boots. The COIL20 dataset contains 20 objects, including 1440 images from varying angles, which are at an interval of 5° . There are 72 images of the size of 32×32 the pixels. The ORL dataset contains 40 subjects, including 400 images. These images are different facial expressions, such as open/closed eyes, glasses/no glasses, and smiling/no smiling facial expressions. Table 1 shows the characteristics of the datasets mentioned above. In Table 1, the second and the third columns show the numbers of features and classes. The fourth and fifth columns show the number of training and test set. Figure 4 are some examples from the four datasets. The experimental results are judged by the mean and the standard deviation values of the accuracy on the test set.

Table 1. Characteristics of the four datasets

Datasets	Characteristic			
	#Classes	#Features	#Training	#Testing
MNIST	10	784	60,000	10,000
Fashion-MNIST	10	784	60,000	10,000
COIL20	20	1024	1150	290
ORL	40	1024	320	80

As mentioned earlier, BAF combines four SAE with four different activation functions in parallel, each SAE has 2 layers. They are *sigmoid*, *tanh*, *relu*, and *softplus* functions. The DAF (Ng et al., 2016), which is used for imbalanced pattern classification problems, is similar to the setting of the BAF. The Sigmoid-AF, the Tanh-AF, the Relu-AF, and the Softplus-AF use the same architecture with BAF, while each of them uses one type of activation function respectively. All the four AFs, the BAF, and the DAF use Softmax as the classifier. In experiment comparison, default values are used for parameters of these methods.

All the four AFs, the BAF, and the DAF use the same number of hidden neurons in each SAE's first and second layers (i.e. $m_1 = m_2$). The experiments run for 10 independent times. In the experiments, we adopt 400 as the number of the first layer's hidden units. For the following layers' hidden unit numbers, they are from 100 to 300 at the step of 100 for all SAEs. Tables 2-5 report the average and the standard deviations over the 10 experiments' ACC, along with the number of second hidden layer units. We can see that the proposed BAF performs best compared with the other methods in the tables.

Moreover, to further prove BAF's superiority, we adopt one-tailed *t*-tests that aim to verify whether BAF's performance is better than the other methods. For instance, when we compare the performance of BAF with DAF, it can be expressed as VAF vs. DAF. The statistically significant level is set to 0.01 in our experiments. Table 6 presents the *p*-values results of the pairwise one-tailed tests on the models' accuracies. It is obvious that the all pairwise one-tailed *t*-tests are less than 0.01. It shows that the proposed BAF performs significantly better than the other methods.

Table 7 shows the overall average and standard deviation results of the four datasets accuracies, which yield that BAF has the best average performance. The BAF has the highest average performance in all datasets according to Figure 5. Moreover, Figure 6 presents the average classification accuracy

Figure 4. Example images of these datasets

(a) MNIST dataset[↵]

(b) Fashion_MNIST dataset[↵]

(c) COIL20 dataset[↵]

(d) ORL dataset[↵]

Table 2. Average $\pm$ standard deviation of ACC of different methods on MNIST dataset

Methods	Number of neurons on the second hidden layer		
	100	200	300
DAF	0.8986 $\pm$ 0.0008	0.9336 $\pm$ 0.0017	0.9061 $\pm$ 0.0026
Sigmoid-AF	0.9307 $\pm$ 0.0019	0.9316 $\pm$ 0.0025	0.9543$\pm$0.0014
Tanh-AF	0.9277 $\pm$ 0.0016	0.9331 $\pm$ 0.0008	0.9356 $\pm$ 0.0008
Relu-AF	0.9214 $\pm$ 0.0023	0.9266 $\pm$ 0.0024	0.9292 $\pm$ 0.0020
Softplus-AF	0.7829 $\pm$ 0.0787	0.8702 $\pm$ 0.1036	0.9220 $\pm$ 0.0446
BAF	0.9407$\pm$0.0012	0.9487$\pm$0.0022	0.9539 $\pm$ 0.0023

Table 3. Average $\pm$ standard deviation of ACC of different methods on FANSHION-MNIST dataset

Methods	Number of neurons on the second hidden layer		
	100	200	300
DAF	0.8193 $\pm$ 0.0017	0.8243 $\pm$ 0.0013	0.8254 $\pm$ 0.0016
Sigmoid-AF	0.8158 $\pm$ 0.0031	0.8280 $\pm$ 0.0023	0.8258 $\pm$ 0.0058
Tanh-AF	0.8518 $\pm$ 0.0017	0.8583 $\pm$ 0.0011	0.8596 $\pm$ 0.0013
Relu-AF	0.8481 $\pm$ 0.0010	0.8529 $\pm$ 0.0023	0.8560 $\pm$ 0.0020
Softplus-AF	0.7316 $\pm$ 0.0960	0.8154 $\pm$ 0.0368	0.8272 $\pm$ 0.0513
BAF	0.8539$\pm$0.0016	0.8612$\pm$0.0036	0.8624$\pm$0.0046

Table 4. Average $\pm$ standard deviation of ACC of different methods on COIL20 dataset

Methods	Number of neurons on the second hidden layer		
	100	200	300
DAF	0.7673 $\pm$ 0.0076	0.8805 $\pm$ 0.0173	0.9239 $\pm$ 0.0141
Sigmoid-AF	0.8343 $\pm$ 0.0162	0.8340 $\pm$ 0.0137	0.8243 $\pm$ 0.0069
Tanh-AF	0.9406$\pm$0.0126	0.8333 $\pm$ 0.0137	0.8256 $\pm$ 0.0065
Relu-AF	0.8329 $\pm$ 0.0156	0.8340 $\pm$ 0.0133	0.8260 $\pm$ 0.0072
Softplus-AF	0.8354 $\pm$ 0.0161	0.8343 $\pm$ 0.0134	0.8253 $\pm$ 0.0071
BAF	0.8659 $\pm$ 0.0510	0.9753$\pm$0.0079	0.9771$\pm$0.0098

Table 5. Average $\pm$ standard deviation of ACC of different methods on ORL dataset

Methods	Number of neurons on the second hidden layer		
	100	200	300
DAF	0.7512 $\pm$ 0.0376	0.7762 $\pm$ 0.0519	0.7800 $\pm$ 0.0312
Sigmoid-AF	0.7575 $\pm$ 0.03596	0.8087 $\pm$ 0.0185	0.8262 $\pm$ 0.0423
Tanh-AF	0.7687 $\pm$ 0.0297	0.8600 $\pm$ 0.0242	0.8725 $\pm$ 0.0235
Relu-AF	0.7662 $\pm$ 0.0335	0.8175 $\pm$ 0.0187	0.8187 $\pm$ 0.0415
Softplus-AF	0.7412 $\pm$ 0.0349	0.7562 $\pm$ 0.0388	0.7425 $\pm$ 0.0363
BAF	0.8462$\pm$0.0256	0.8637$\pm$0.0258	0.8812$\pm$0.0232

Figure 5. Boxplots of overall performance of the BAF and other methods on four datasets

(a) Number of neurons on the second hidden layer (100)^c

(b) Number of neurons on the second hidden layer (200)^c

(c) Number of neurons on the second hidden layer (300)^c

Table 6. The p -values of the pairwise one-tailed tests on ACC

	P -values		
	100	200	300
BAF vs. DAF	1.27004E-19	2.47263E-09	4.19913E-18
BAF vs. Sigmoid-AF	3.16259E-07	8.60066E-10	0.00030609
BAF vs. Tanh-AF	1.0857E-05	4.60563E-06	1.66303E-06
BAF vs. Relu-AF	2.0775E-07	1.54036E-07	5.70163E-07
BAF vs. Softplus-AF	8.00467E-07	0.0002938	0.000190216

Table 7. Overall results of ACC cross four datasets

Methods	Number of neurons on the second hidden layer		
	100	200	300
DAF	0.8091 $\pm$ 0.0663	0.88536 $\pm$ 0.0682	0.8588 $\pm$ 0.0678
Sigmoid-AF	0.8345 $\pm$ 0.0719	0.8505 $\pm$ 0.0551	0.8576 $\pm$ 0.0644
Tanh-AF	0.8722 $\pm$ 0.0793	0.8711 $\pm$ 0.0430	0.8733 $\pm$ 0.0459
Relu-AF	0.8421 $\pm$ 0.0639	0.8712 $\pm$ 0.0399	0.8706 $\pm$ 0.0433
Softplus-AF	0.7727 $\pm$ 0.0473	0.8190 $\pm$ 0.0476	0.8292 $\pm$ 0.0733
BAF	0.8766$\pm$0.0434	0.9122$\pm$0.0585	0.9187$\pm$0.0554

Figure 6. Bars of overall performance of the BAF and other methods on four datasets

of all the methods, in which BAF yields the highest average results. The acceptable reason that BAF outperforms DAF and other AFs is that more effective features are learned by AEs using different activation functions.

5. CONCLUSION

In this paper, we propose the BAF to deal with the image pattern classification problems. Stacked AEs are used in the BAF, and four different activation functions are used to assert a more robust feature learning process compared with the DAF and other four SAEs which use one type of activation function. Experimental results show that the BAF yields higher statistically significant accuracies than the feature-learning based DAF and other four methods for different datasets. Classification accuracies yielded in experiments show that the BAF with *sigmoid*, *tanh*, *relu*, and *softplus* stacked AEs provides a significantly better feature representation. The architecture of the network will be further studied to improve the performance of the BAF in the future work. Automatic selection of activation functions and network architecture of BAF is an important future work of us.

ACKNOWLEDGEMENT

This research was funded by National Natural Science Foundation of China under Grant 61876066 and Guangdong Province Science and Technology Plan Project (Collaborative Innovation and Platform Environment Construction) 2019A050510006.

REFERENCES

- Al Rahhal, M. M. (2016). Deep Learning Approach for Active Classification of Electrocardiogram Signals. *Information Sciences*. <https://dl.acm.org/doi/10.1016/j.ins.2016.01.082>
- Baldi, P. (2012). Autoencoders, unsupervised learning, and deep architectures. *International Conference on Unsupervised and Transfer Learning Workshop*, 27, 37–50. <https://dl.acm.org/doi/10.5555/3045796.3045801>
- Bengio, Y. (2009). Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning*, 2, 1–127. <http://www.iro.umontreal.ca/~lisa/pointeurs/TR1312.pdf>
- Burda, Y., Grosse, R., & Salakhutdinov, R. (2016). Importance Weighted Autoencoders. *International Conference on Learning Representations*. <https://arxiv.org/abs/1509.00519>
- Byrd, Lu, Nocedal, & Zhu. (1995). A limited Memory Algorithm for Bound Constrained Optimization. *SIAM Journal on Scientific Computing*, 61, 1190–1208. <https://locus.siam.org/doi/abs/10.1137/0916069>
- Diederik, P. K., & Welling, M. (2014). Auto-Encoding Variational Bayes. *International Conference on Learning Representations*. <https://arxiv.org/pdf/1312.6114.pdf>
- Han, J., Zhang, D., Wen, S., Guo, L., Liu, T., & Li, X. (2016). Two-Stage Learning to Predict Human Eye Fixations via SDAEs. *IEEE Transactions on Systems, Man, and Cybernetics*, 46, 487–498. <https://ieeexplore.ieee.org/document/7051244>
- Heydarzadeh, M., Nourani, M., & Ostadabbas, S. (2016). In-bed Posture Classification Using Deep Autoencoders. In *International Conference of the IEEE Engineering in Medicine and Biology Society* (pp. 3839–3842). <https://ieeexplore.ieee.org/document/7591565>
- Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). Article. *Neural Computation*, 18, 1527–1554. <https://ieeexplore.ieee.org/document/6796673>
- Jiang, G., He, H., Xie, P., & Tang, Y. (2017). Stacked Multilevel-Denoising Autoencoders: A New Representation Learning Approach for Wind Turbine Gearbox Fault Diagnosis. *IEEE Transactions on Instrumentation & Measurement*, 99, 1–12. <https://ieeexplore.ieee.org/abstract/document/7932980>
- Kang, Y., Lee, K., Eun, J., Park, S. E., & Choi, S. (2013). Stacked Denoising Autoencoders for Face Pose Normalization. In *International Conference on Neural Information Processing, Volume 3* (pp. 241–248). https://link.springer.com/chapter/10.1007/978-3-642-42051-1_31
- Krizhevsky, A., & Hinton, G. E. (2011). Using Very Deep Autoencoders for Content-based Image Retrieval. *European Symposium on Artificial Neural Networks*. <https://www.cs.toronto.edu/~hinton/absps/esann-deep-final.pdf>
- Man Wong, C., Man Vong, C., Wong, P. K., & Cao, J. (2018). Kernel-based Multilayer Extreme Learning Machines for Representation Learning. *IEEE Transactions on Neural Networks and Learning Systems*, 29, 757–762. <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7801860>
- Masci, J., Meier, U., Ciresan, D., & Schmidhuber, J. (2011). Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction. In *International Conference on Artificial Neural Networks, Volume 6791* (pp. 52–59). https://link.springer.com/chapter/10.1007%2F978-3-642-21735-7_7
- Mescheder, L., Nowozin, S., & Geiger, A. (2017). *Adversarial Variational Bayes: Unifying Variational Autoencoders and Generative Adversarial Networks* (Vol. 70). International Conference on Machine Learning. <https://arxiv.org/abs/1701.04722>
- Mnih, A., & Hinton, G. E. (2007). Restricted Boltzmann Machines for Collaborative Filtering Ruslan Salakhutdinov. International Conference on Machine Learning. <https://dblp.org/rec/bib/conf/icml/2007>
- Ng, A. (2011). *Sparse Autoencoder CS294A Lecture Notes*. <https://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf>
- Rifai, S., Vincent, P., Muller, X., Glorot, X., & Bengio, Y. (2011). Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. In *International Conference on Machine Learning* (pp. 833–840). <https://dl.acm.org/doi/10.5555/3104482.3104587>

Salakhutdinov, R., & Hinton, G. (2009). Deep Boltzmann Machines. In *International Conference on Artificial Intelligence and Statistics* (pp. 448-455). <http://proceedings.mlr.press/v5/salakhutdinov09a.html>

Su, S., Liu, Z., Xu, S., Li, S., & Ji, R. (2015). *Sparse Auto-Encoder based Feature Learning for Human Body Detection in Depth Image* *Journal Signal Processing* (Vol. 27). <https://dl.acm.org/doi/10.1016/j.sigpro.2014.11.003>

Tan, Q., Gao, L., Lai, Y., & Xia, S. (2018). *Variational Autoencoders for Deforming 3D Mesh Models*. *Computer Vision and Pattern Recognition*. <https://ieeexplore.ieee.org/document/8578710/>

Tang, J., Deng, C., & Huang, G. (2016). Extreme Learning Machine for Multilayer Perceptron. *IEEE Transactions on Neural Networks and Learning Systems*, 27, 809–821. <https://ieeexplore.ieee.org/document/7103337?arnumber=7103337&contentType=Early%20Access%20Articles>

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P. (2010). Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *Journal of Machine Learning Research*, 11, 3371–3408. <https://dl.acm.org/citation.cfm?id=1953039&dl=ACM&coll=DL>

Wasikowski, M., & Chen, X. (2010). Combating the Small Sample Class Imbalance Problem Using Feature Selection. *IEEE Transactions on Knowledge and Data Engineering*, 22, 1388–1400. <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5276797>

Weninger, F., Watanabe, S., Tachioka, Y., & Schuller, B. (2014). *Deep Recurrent De-Noising Auto-Encoder and Blind De-Reverberation for Reverberated Speech Recognition* *International Conference on Acoustics*. *Speech and Signal Processing*. <https://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6854478>

Wing, Ng, Zeng, Zhang, Yeung, & Pedrycz. (2016). Dual Autoencoders Features for Imbalance Classification Problem. *Pattern Recognition*, 60, 875-889. <https://www.sciencedirect.com/science/article/abs/pii/S0031320316301303#>

Yuan, X., Huang, B., Wang, Y., Yang, C., & Gui, W. (2018). Deep Learning-based Feature Representation and Its Application for Soft Sensor Modeling with Variable-Wise Weighted SAE. *IEEE Transactions on Industrial Informatics*, 14, 3235–3243. <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7801860>

Ting Wang (S'19) received the M.Sc. in computer science from Northeast Normal University, China, in 2017. She is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, South China University of Technology, China. Her current research interests include learning methods and generalization capabilities for deep neural networks, and their applications in real-world problems, such as smart healthcare and smart grid, etc.

Wing W. Y. Ng (Prof.) (S'02, M'05, SM'15) received his B.Sc. and Ph.D. degrees from Hong Kong Polytechnic University in 2001 and 2006, respectively. He is a Professor with the School of Computer Science and Engineering, South China University of Technology, China. His major research directions include neural networks, deep learning, smart grid, smart healthcare, smart manufacturing, and nonstationary information retrieval. Prof. Ng is currently the deputy director of the Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information and an associate editor of the International Journal of Machine Learning and Cybernetics. He is the principle investigator of four China National Nature Science Foundation projects and a Program for New Century Excellent Talents in University from China Ministry of Education. Dr. Ng served as the Board of Governor of IEEE Systems, Man and Cybernetics Society in 2011 - 2013. He is the corresponding author of this article, wingng@ieee.org.

Wendi Li received the B.E. degree with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China. His current research interests include machine learning, evolutionary computation, and computer vision.

Sam Kwong (M'93-SM'04-F'14) received the B.Sc. degree in electrical engineering from the State University of New York, Buffalo, NY, USA, in 1983 and the M.Sc. degree in electrical engineering from the University of Waterloo, Waterloo, ON, Canada, in 1985, and the Ph.D. degree from the Fernuniversitaet, Hagen, Germany, in 1996. From 1985 to 1987, he was a Diagnostic Engineer with the Control Data Canada, Mississauga, ON, Canada. He later joined Bell Northern Research Canada, Ottawa, ON, Canada, as a Member of Scientific Staff, and the City University of Hong Kong (CityU), Hong Kong, as a Lecturer with the Department of Electronic Engineering in 1990. He is currently a Chair Professor with the Department of Computer Science. His research interests include video coding, pattern recognition, and evolutionary algorithms. Prof. Kwong was elevated to an IEEE fellow for his contributions on optimization techniques for cybernetics and video coding in 2014. He is also appointed as an IEEE Distinguished Lecturer of the IEEE SMC Society in 2017. He is currently the Vice-President of Cybernetics with the IEEE Systems, Man and Cybernetics. He also serves as an Associate Editor of the IEEE Transactions on Evolutionary Computation, the IEEE Transactions on Industrial Electronics, and the IEEE Transactions on Industrial Informatics, and the Journal of Information Science.