Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 13 (2012) 120 – 127

Procedia Computer Science

Proceedings of the International Neural Network Society Winter Conference (INNS-WC 2012)

A Folded Neural Network Autoencoder for Dimensionality Reduction

Jing Wang[a], Haibo He[a,*], Danil V. Prokhorov[b]

[a]*Department of Electrical, Computer and Biomedical Engineering, University of Rhode Island, Kingston, RI, 02881, USA*
[b]*Toyota Research Institute NA, TTC, Ann Arbor, MI, 48105, USA*

Abstract

Dimensionality reduction has been a long-standing research topic in academia and industry for two major reasons. First, in almost every domain, ranging from biology, social science, economics, to military data processing applications, the increasingly large volume of data is challenging the existing computing capability and raising the computing cost. Second, the notion of "intrinsic structure" allows us to remove some redundant dimensions from high-dimensional observations and reduce it into low-dimensional features without significant information loss. Autoencoder, as a powerful tool for dimensionality reduction has been intensively applied in image reconstruction, missing data recovery and classification. In this paper, we propose a new structure, folded autoencoder based on symmetric structure of conventional autoencoder, for dimensionality reduction. The new structure reduces the number of weights to be tuned and thus reduces the computational cost. Simulation results over MNIST data benchmark validate the effectiveness of this structure.

Keywords: folded autoencoder, neural network, dimensionality reduction, image reconstruction,

1. Introduction

Dealing with huge amount of high-dimensional data, such as global climate pattern, human gene expression microarray and even customer reviews on large B2C websites, scientists confront the problem of the dimensionality reduction everyday and therefore hope to discover efficient methods to transform the high-dimensional data into a more compact and meaningfully expression in low-dimensional space. Conventional techniques for dimensionality reduction including principle component analysis (PCA) and multidimensional scaling (MDS) are widely used in these domains for their implementing easiness and computing efficiency. Once the true patterns embedded in data are found, the data could be compressed without too much loss. However, in many cases, the non-linear property of the data will render both PCA and MDS ineffective. As indicated in [1], both techniques fail to detect the true degree of freedom of complex natural observations, such as human face under different viewing conditions. Therefore, a global geometric framework for dimensionality reduction: Isomap is proposed to overcome this limitation through combination of PCA and MDS[1]. To avoid to solving large dynamic programming

*Corresponding author. Tel.: 1-401-874-5844; Fax: 1-401-782-6422.
Email address: he@ele.uri.edu (Haibo He)

problems in dimensionality reduction, locally linear embedding (LLE), using the local symmetries of data, is proposed [2]. LLE is also guaranteed to converge to the global optima. Though, Isomap and LLE are more powerful to find the intrinsic structure of data than conventional techniques, the primary limitation for both methods is that they could only learn the one-way mapping from high-dimensional space to low-dimensional space. Autoencoder, first introduced in [3], is capable to learn the the two-way mapping relationships between high-dimensional space and low-dimensional space. Hence autoencoder could either convert high-dimensional input to low-dimensional codes(encoding) or synthesize the input from corresponding low-dimensional codes(decoding). Moreover, autoencoder can be trained using backpropagation algorithm, which is very simple to implement and can be easily expanded to train autoencoder with arbitrary hidden layers. For the advantages aforementioned, autoencoder is widely used to solve dimensionality reduction problems in various domains.

A simple application of autoencoder in dimensionality reduction can be found in [4], in which an autoencoder containing 3 hidden layers is used to extract the features from CT images. A framework based on ensemble learning is proposed in [5] for 3D object recognition. The framework is termed as multi-sensor model and each autoencoder contained in it will learn the target images from many viewpoints and combine multiple results for final decision. Lange[6] further explores the application of autoencoder by integrating autoencoder to a reinforcement learning system. Provided by the image from digital camera, high-dimensional information from environment is learned by autoencoder and reduced to low-dimensional feature. Based on these "concise" description of environment, learning system will make an agent perform corresponding actions to complete a specific task. Another important application for autoencoder is missing data recovery. A fault-tolerant indirect adaptive neurocontroller (FTIANC)[7] is proposed to control a static synchronous series compensator (SSSC) and sensor evaluation and (missing sensor) restoration scheme (SERS) is integrated into the controller. When some crucial time-varying sensor measurements are corrupted or unavailable, SERS will reconstruct the corrupted or missing sensor measurements and guarantee that FTIANC is still able to effectively control the SSSC.

In practical application of dimensionality reduction, such as image compression, the computing time is always a major concern. For conventional autoencoder the computing time would increase linearly as the dimension of input data increases. Based on results presented in[8] and motivated by the symmetric structure of conventional autoencoder, we propose a new structure for autoencoder to reduce the computing time. Specifically, in proposed structure, the number of weights to be tuned is reduced by power of two compared with conventional structure for autoencoder. We hope this improvement could make a concrete step toward the real-time application of autoencoder. Empirical study between proposed structure and conventional structure for dimensionality reduction, assisted with squared reconstruction error, are also presented in this article.

The rest of article is organized as follows. In Section2, we briefly introduce the mechanism of conventional autoencoder and show the effectiveness of Restricted Boltzmann Machine (RBM) in weights initialization. At the end of this section, we will discuss the symmetric property of weights in conventional autoencoder, upon which our new structure for autoencoder is built. In Section3, we present the implementation-level framework and algorithms for proposed structure. We discussed in detail about the proposed structure for dimensionality reduction. In Section4, we present the reconstruction results both visually and numerically, through reconstruction images and squared reconstruction error curve. Finally, Section5 concludes the work we have done and discusses several research directions in the future.

2. Unfolded autoencoder

Autoencoder is a multi-layer perceptron (MLP) that has symmetric structure and designed to learn an approximation to the identity function, so as to the output is as similar to the input as possible. As illustrated in Fig. 1, an autoencoder is composed of two networks, the "encoder" network which is used to transform the input from high-dimensional space into features or codes in low-dimensional space and a symmetric "decoder" network which is to used reconstruct the input from corresponding codes[8]. Two networks are trained jointly by tuning the weights of decoder network first and encoder network next. The goal is to minimize the deference between the reconstruction/output and origin/input.

Usually, the weights are initialized randomly through sampling a given distribution and backpropagation is chosen as learning algorithm. Therefore, the selection of initial weights are crucial for an autoencoder to find a better

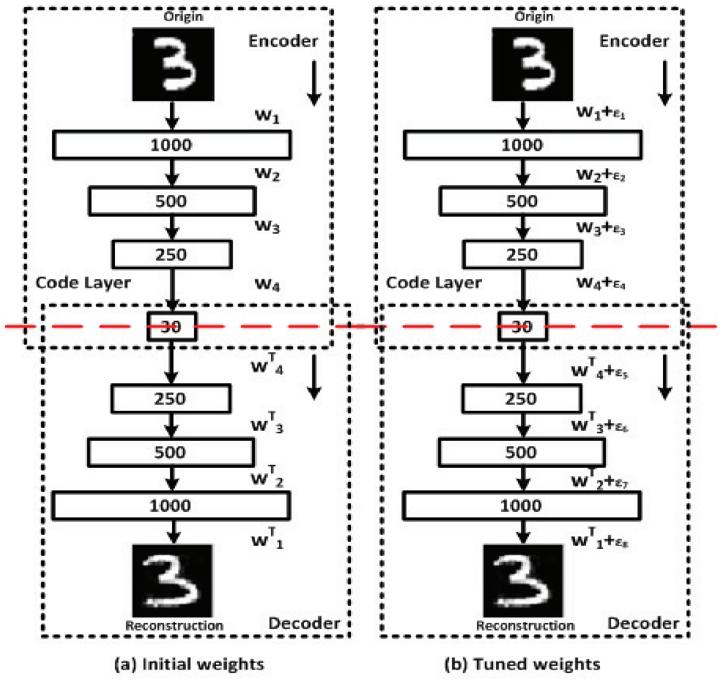
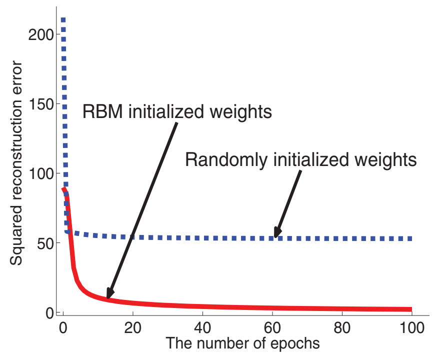


Fig. 1: Autoencoder Neural network[8]

solution. If the initial weights are close enough to a good solution, backpropagation works well even for autoencoder that contains multiple hidden layers. However, if the initial weights make the real output far away from the the desired output, backpropagation are more likely to converge to a poor local minimal. A lot of algorithms have been designed and hopefully to find a better initial weights. Here, we will only focus on one of these algorithms for weights initialization - Restricted Boltzmann Machine (RBM). RBM is powerful tool that can learn and represent complicated distributions and is successfully used in autoencoder for dimensionality reduction of image[8]. Unlike randomly initialized weights, RBM calculates the initial weights and bias according to the underlying structure of training data. Starting from the weights initialized by RBM, backpropagation is much more likely to find a local optima that is closer to the global optima. A "mini" version of MNIST data set[9], is used to show the effectiveness of RBM and the mini MNIST data set contains 200 samples randomly drawn from original MNIST data set. The weights are initialized through randomly sampling and RBM, respectively. Two autoencoders are trained over 140 samples for 100 epochs and tested over another 60 samples. The reconstruction results as well as the squared reconstruction error are depicted in Fig. 2. Compare second row and third row of the reconstruction result and the squared reconstruction error, it is clear that weights generated by RBM significantly improve the reconstruction result. For all the results presented in this paper, bias has been removed from both conventional autoencoder and proposed autoencoder structures and the reason will be discussed in detail in section3.

As illustrated in Fig.1(a), the weights initialized by RBM are transposed identical in pairs. The weights tuning process is equivalent to adding a correction term ($\epsilon_1 \sim \epsilon_8$) to each weight, which is illustrated in Fig.1(b). Therefore, we would like to know how would the weights change after they are tuned over a relatively long learning epochs, would the property of symmetry still hold to some extent? The autoencoder used previously is trained and tested over the same data set and the maximum learning epoch is increased to 1000. The symmetric weights pairs that we compared are depicted in Fig. 3. To measure the difference between these weights pairs, here we define the mean squared error (MSE) as the square of the substraction of one weight w_l from its symmetric counterpart w_{L-l} over each component of the weight, where L is the number of hidden layers.

$$MSE = \frac{(w_{L-l} - w_l)^2}{the\ number\ of\ element\ in\ w_l},\ l = 0, \ldots, (L-1)/2 \tag{1}$$



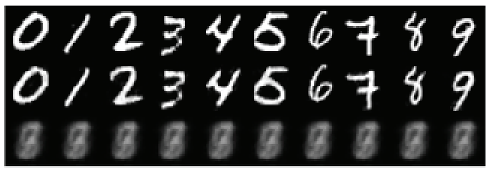


Fig. 2: RBM initialized weights versus randomly initialized weights

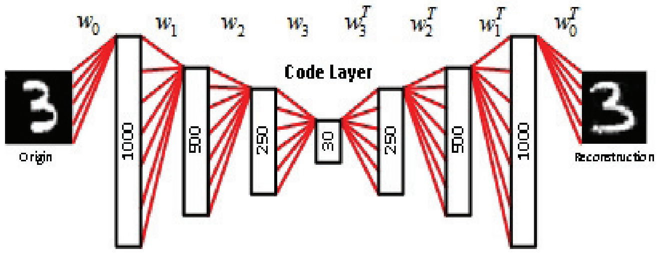


Fig. 3: Symmetric initial weights pairs for a “784-1000-500-250-30” autoencoder

The maximum and minimum value of each weight and MSE between each symmetric weights pair can be found in Table1 [1]. We can see that for each weights pair, MSE is small enough compared to the range of the symmetric weights pair. Or we can say that the weights of autoencoder retain a asymptotically symmetric structure even after being tuned for a relatively long period. Therefore, we would like to conjecture: whether autoencoder could be implemented by using only half of its symmetric structure and tuning half of its weights? Based on this conjecture, a new structure for autoencoder is proposed and will be discussed in detail in next Section.

Table 1: MSE comparison between symmetric weights pairs

Weight pairs	MSE	Min/Max
$w0$	7.69×10^{-5}	−0.6112/0.6145
$w7$	–	−0.6157/0.6192
$w1$	1.53×10^{-5}	−0.4771/0.4823
$w6$	–	−0.4776/0.4853
$w2$	1.14×10^{-3}	−0.5666/0.6089
$w5$	–	−0.6025/0.6476
$w3$	6.74×10^{-3}	−1.6674/1.6335
$w4$	–	−1.7081/1.5845

[1]For convenience of discussion, the index starts from 0 in this article.

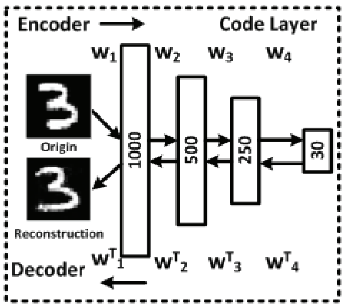


Fig. 4: System-level framework of folded autoencoder

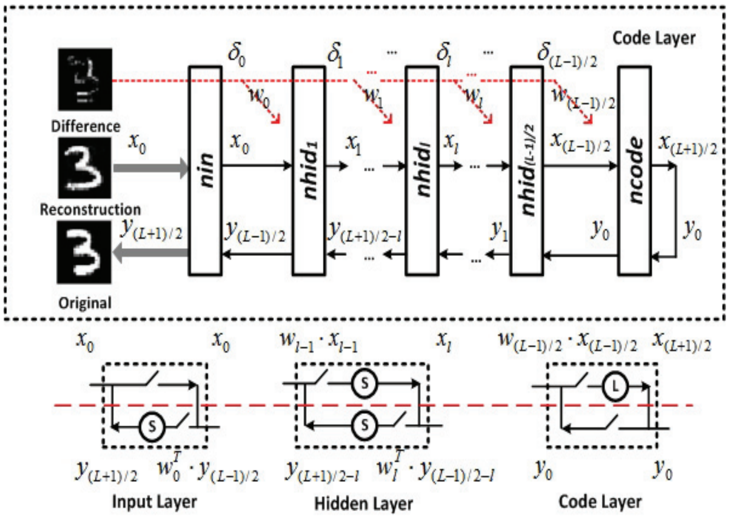


Fig. 5: Implementation-level framework for folded autoencoder

3. Folded autoencoder

In this section, we present a new structure for autoencoder, which is based on the symmetric property of conventional autoencoder. The system-level framework of the new structure is illustrated in Fig.4. Compare with the autoencoder illustrated in Fig.1, we can see the new structure is generated by folding the right side of the conventional structure to the left side. For convenience of discussion, we would like to name the new structure as folded autoencoder and conventional structure as unfolded autoencoder accordingly.

A folded autoencoder containing $(L - 1)/2$ hidden layers is equivalent to an unfolded autoencoder containing L hidden layers and the primary difference of computing cost between two structures lies in the number of weights to be tuned, which is reduced by half in folded autoencoder. RBM is used to initialize the weights for both structures. But bias is removed from the new structure because it will break the symmetry and reduce the performance of the new structure. For a fair comparison, bias is also removed from unfolded autoencoder. Interested readers could refer to [8] and [10] for methods of training unfolded autoencoder using RBM and backpropagation, as well as derivation of RBM.

The implementation-level framework of folded autoencoder is illustrated in Fig.5. The black solid line represents the direction that data propagates and red dashed line represents the direction that error propagates. When autoencoder works in "encoding" mode, original image data propagates from input layer to code layer, where original image is reduced to low-dimensional codes. Whereas in "decoding" mode data "flows" in the reverse direction: low-dimensional codes are expanded layer by layer and eventually mapped to a high-dimensional space with the same dimensionality of original data, where reconstructed image is obtained. In folded autoencoder, data should

be allowed to pass through each neuron from either direction and thus the structure of neuron is modified accordingly. As shown in the bottom of Fig.5, for a single neuron, the "upper route" will be turned on and the "lower route" would be turned off, when autoencoder works in encoding mode and ***vise versa***.
Consider an unfolded autoencoder that has L hidden layers and $L + 1$ weights, $\mathbf{W} = \{w_1, \ldots, w_{L+1}\}$, to be tuned. For an equivalent folded autoencoder, as depicted in Fig.5, only half of the weights, $\mathbf{W} = \{w_1, \ldots, w_{(L+1)/2}\}$, are needed to be tuned. The algorithm for implementing a folded autoencoder neural network that has $(L-1)/2$ hidden layers is given below.

```
[Algorithm I: The folded autoencoder]
Initialize weights
$\mathbf{W} = \{w_0, \ldots, w_{(L-1)/2}\} \leftarrow \mathbf{RBM}$
$\mathbf{x_1} = \{x_1, \ldots, x_m, \ldots, x_M\}, x_m \in \Re^n$

while($epoch < 1000$ and $MSE > 10^{-8}$)
for $m = 1 : M$

Generate reconstruction image
$x_{l,m} = \sigma(w_{l-1} \cdot x_{l-1,m})$
where, $l = 1, \ldots, (L-1)/2$
$x_{(L+1)/2,m} = w_{(L-1)/2,m} \cdot x_{(L-1)/2,m}$
$y_{0,m} = x_{(L+1)/2,m}$
$y_{(L+3)/2-l,m} = \sigma(w_{l-1}^T \cdot y_{(L+1)/2-l,m})$
where, $l = (L+1)/2, \ldots, 1$

Tune weights using backpropagation
$err_m = x_{0,m} - y_{(L+1)/2,m}$
$\delta_{0,m} = err_m \cdot (1 - y_{(L+1)/2,m}) \cdot y_{(L+1)/2,m}$
$\delta_{l,m} = w_{l-1} \cdot \delta_{l-1,m} \cdot (1 - y_{(L+1)/2-l,m}) \cdot y_{(L+1)/2-l,m}$
$\Delta w_l = \eta \cdot \delta_l \cdot y_{(L-1)/2-l,m}$
$w_l \leftarrow w_l + \Delta w_l$
where, $l = 1, \ldots, (L-1)/2$
end
$epoch \leftarrow epoch + 1$
end
```

where, the subscripts, l and m denote the indices of hidden layer and current sample in training set. Activation functions applied in all neurons are sigmoid function which is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

Readers should be careful with input layer and code layer and treat them differently with other hidden layers. First, the activation function for code layer neurons is linear. Second, the input layer is "invisible" to original input data which means input data could directly pass through the input layer without involving any arithmetic operation. Similarly, code layer is also "invisible" to extracted codes. Folded autoencoder with arbitrary layers could be trained by routine mentioned above. Note that unfolded autoencoder is symmetric about the code layer, therefore L should always an odd number.

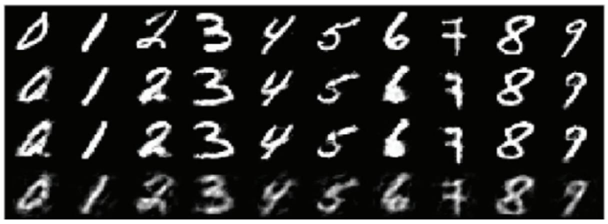


Fig. 6: Top to bottom: Original images; reconstruction image by folded autoencoder; reconstruction image by unfolded autoencoder; reconstruction image by standard PCA using 30 principal components

4. Simulation Analysis and Discussions

To validate the performance of proposed structure for dimensionality reduction, proposed structure for autoencoder is trained and tested over MNIST data set. Both folded and unfolded autoencoders with the structure of "$784-1000-500-250-30$" are used to extract the "codes" from MNIST data set. MNIST data set is a collection of $70,000$ handwritten digits by different writers. Considering the high computing cost of backpropagation and here our purpose is limited to compare the performance between two structures for autoencoder. A mini version of MNIST data set is used in this experiment. This mini MNIST data set is randomly drawn from MNIST data set and contains 200 images, with 20 exemplars for each digit.

In training period, RBM and backpropagation are employed to generate identical initial weights and tune the weights for both folded and unfolded autoencoders, respectively. Being randomly permutated, 70% samples are used for training and remaining 30% samples are used for test. The training process is considered to be complete when current epoch exceeds the maximum number of epochs or the squared construction error is less than the preset threshold. The maximum epoch and maximum squared reconstruction error are set to 1000 and 10^{-8} in this experiment.

Table 2: MSE comparison among different dimensionality reduction techniques

Method	Training MSE	Test MSE
Folded autoencoder	0.0351	35.1535
Unfolded autoencoder	0.0323	32.0443
PCA	11.6745	19.2997

The reconstruction results for folded/unfolded autoencoder are illustrated in Fig.6, a sample drawn from original test set is plotted in the first row and corresponding reconstructions by folded and unfolded autoencoders are plotted in the second and third row. To fully evaluate the performance of neural network methods and conventional methods for dimensionality reduction, we compare both folded and unfolded autoencoders with the PCA introduced in[11]. Similarly, 70% samples are used for training and remaining 30% samples are used for test. For a fair comparison, 30 principal components are used to represent the test data set and the image reconstructed from 30 principal components are plotted in the last row in Fig.6. Compare the reconstruction images from the second row to the last row, we can see that proposed autoencoder matches conventional autoencoder and both neural network based method outperform PCA. Finally, the MSEs for three dimensionality reduction techniques are listed in Table2. Notice that, the inconsistency between the MSEs and reconstruction effects from the perspective of human vision, which could be explained from the point of view that MSE is not a good measure to evaluate the similarity between two images[12].

We also record the training time that required for one round implementation of folded autoencoder and unfolded autoencoder over mini MNIST data set for 1000 epochs. Compared with unfolded autoencoder, folded autoencoder largely reduces the training time by 17.2%.

5. Conclusion and future work

In this paper, we start from autoencoder, a powerful tool for dimensionality reduction and propose a new structure - folded autoencoder for image reconstruction. The implementation-level structure as well as the algorithms are presented and analyzed in great detail. We compare the performance of new structure with conventional structure for autoencoder and MNIST data set is used in comparison. Simulation results validate that the new structure not only could reduce the computing time but improve the generalization performance. There are several interesting future research directions along this topic. First, our work is primarily focused on the development of new framework and algorithms for dimensionality reduction as well as empirical study in this paper. The proposed framework should be further investigated from the aspect of theoretical analysis, such as learning capability, computing complexity and robustness. Second, though the simulation in this paper show some promising results, experiment on large scale, more complex data sets as well as statistical tests are necessary to fully evaluate the capability of new framework. Third, our discussion is only limited to autoencoder in this paper. There are many new frameworks and techniques have been developed and successfully applied in dimensionality reduction, such as convolutional network, cognitive neural network and incremental learning. Therefore, it would be very interesting to see how the proposed framework performs when it is integrate to these frameworks and techniques.

6. Acknowledgement

This work was supported in part by the National Science Foundation (NSF) under grant ECCS 1053717, Defense Advanced Research Projects Agency (DARPA) Mathematics of Sensing, Exploitation and Execution (MSEE) under grant FA8650-11-1-7148, and the K. C. Wong Education Foundation. The authors are also very grateful to the Toyota Research Institute NA for the suggestions and comments for this work.

References

[1] J. B. Tenenbaum, V. de Silva, J. C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323.

[2] S. Roweis, L. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.

[3] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Neurocomputing: foundations of research, 1988, Ch. Learning internal representations by error propagation, pp. 673–695.

[4] A. Steudel, S. Ortmann, M. Glesner, Medical image compression with neural nets, in: Proceedings of the 3rd International Symposium on Uncertainty Modelling and Analysis, ISUMA '95, IEEE Computer Society, Washington, DC, USA, 1995, pp. 571–576.

[5] Y. Yaginuma, T. Kimoto, H. Yamakawa, Multi-sensor fusion model for constructing internal representation using autoencoder neural networks, in: Neural Networks, 1996., IEEE International Conference on, Vol. 3, 1996, pp. 1646–1651 vol.3. doi:10.1109/ICNN.1996.549147.

[6] S. Lange, M. Riedmiller, Deep auto-encoder neural networks in reinforcement learning, in: IJCNN, 2010.

[7] Q. Wei, R. G. Harley, G. K. Venayagamoorthy, Fault-tolerant indirect adaptive neurocontrol for a static synchronous series compensator in a power network with missing sensor measurements, Neural Networks, IEEE Transactions on 19 (7) (2008) 1179–1195.

[8] G. E. Hinton, R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.

[9] Y. LeCun, C. Cortes, THE MNIST DATABASE of handwritten digits.
URL `http://yann.lecun.com/exdb/mnist/index.html`

[10] P. Smolensky, Parallel distributed processing: explorations in the microstructure of cognition, vol. 1, 1986, Ch. Information processing in dynamical systems: foundations of harmony theory, pp. 194–281.

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

[12] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, Image quality assessment: from error visibility to structural similarity, Image Processing, IEEE Transactions on 13 (4) (2004) 600 –612.