Revisit Lmser from a Deep Learning Perspective

Wenjin Huang, Shikui Tu(✉), and Lei Xu(✉)

Department of Computer Science and Engineering,
Centre for Cognitive Machines and Computational Health (CMaCH), SEIEE School,
Shanghai Jiao Tong University, Shanghai, China
{huangwenjing,tushikui,leixu}@sjtu.edu.cn

Abstract. Proposed in 1991, Least Mean Square Error Reconstruction for self-organizing network, shortly Lmser, was a further development of the traditional auto-encoder (AE) by folding the architecture with respect to the central coding layer and thus leading to the features of Duality in Connection Weight (DCW) and Duality in Paired Neurons (DPN), as well as jointly supervised and unsupervised learning which is called Duality in Supervision Paradigm (DSP). However, its advantages were only demonstrated in a one-hidden-layer implementation due to the lack of computing resources and big data at that time. In this paper, we revisit Lmser from the perspective of deep learning, develop Lmser network based on multiple fully-connected layers, and confirm several Lmser functions with experiments on image recognition, reconstruction, association recall, and so on. Experiments demonstrate that Lmser indeed works as indicated in the original paper, and it has promising performance in various applications.

Keywords: Autoencoder · Lmser · Bidirectional deep learning

1 Introduction

Least Mean Square Error Reconstruction (Lmser) self-organizing network was first proposed in 1991 [13,14], and it is a further development of autoencoder (AE) with favorable features. Early efforts on AE can be traced back to 1980s. Three-layer networks, i.e., networks with only one hidden layer, were used to make auto-association to learn inner representations of observed signals [1,2]. In this framework, the network architecture is considered to be symmetric with the hidden layer as the central coding layer Y. The input pattern X is mapped through the encoder part to the central coding layer, while the output $\hat{X}$ is constrained to reconstruct the input via the decoder part to decode the internal representations back to the data space.

The Lmser architecture is obtained from folding AE along the central coding layer Y, and then is improved into a distributed cascading by not only constraining $X \rightarrow Y$ and $Y \rightarrow \hat{X}$ to share the same architecture, but also using the same neurons for the two layers symmetrically paired between the encoder and decoder with respect to the central coding layer, and using the same connection weights for the bidirectional links along the directions of $X \to Y$ and $Y \to \hat{X}$. One neuron takes dual roles in encoder and decoder, and this nature is shortly called Duality in Paired Neurons (DPN), which can be regarded as adding short-cut connections between the paired neurons. Using the same connection weights for the bidirectional links is referred to $A_j = W_j^T$, where W_j is the weight matrix for layer j in the direction $X \to Y$ and A_j is the weight matrix at the corresponding layer in the direction $Y \to \hat{X}$. The matrix equality $A_j = W_j^T$ indicates that each connection weight between a pair of neurons plays a dual role for both directions, and thus this nature is shortly called Duality in Connection Weight (DCW). DCW enables Lmser to approximate identity mapping for each layer simply through $W_j A_j = A_j^T A_j \approx I$ which holds exactly for an orthogonal matrix A_j. Therefore, AE implements a direct approximation of inverse of $X \to Y$ by $Y \to \hat{X}$ in a simple cycle, whereas Lmser improves the direct cascading into a distributed cascading by DPN and DCW.

Due to the above architectural features, Lmser works in two phases, i.e., perception phase and learning phase. In perception phase, the signal propagation from two directions $X \to Y$ and $Y \to \hat{X}$ constitute a dynamic process which will approach equilibrium in a short term [14]. If the reconstruction $\hat{X}$ is not close to the input X, then Lmser enters the learning phase to update the connection weights to reduce the discrepancy between X and $\hat{X}$. Moreover, part of the central (or top) layer Y can be used for label prediction Y_L and thus supervised learning at the top by labeled data and unsupervised learning at the bottom by unlabeled data are made jointly. This nature is shortly called Duality in Supervision Paradigm (DSP). More advances about Lmser are referred to a recent review in [15].

As discussed in [13], Lmser potentially has many functions. However, due to the lack of powerful computing facility and big data at the time of 1990's, Lmser was implemented by computer simulations with only one hidden layer. It was shown that a neuron in Lmser net behaved similar to a feature detector in the cortical field during learning [13]. In recent years, some features of Lmser were also adopted in the literature. For example, stacked restricted Boltzmann machines (RBMs) [5] constrained the weight parameters to be symmetrically shared by the encoder and decoder network, while neurons in U-Net [11] and deep RED-Net [10] shared values by skip connections from the layers in the encoder to the layers in the decoder. However, whether Lmser indeed works on deep network structures and whether it is effective for those potential functions as indicated in [13], are still not systematically explored.

In this paper, we revisit Lmser by implementing it on a multi-layer network, and confirm that it indeed works as indicated in [13,14]. Since the dynamic process in the perception phase of multi-layer Lmser net can not be exactly implemented in practice, we present an effective implementation to approximate the dynamic process by updating neurons in a layer-by-layer way. Meanwhile, instead of using original Lmser learning rule to calculate gradient when updating parameters, we compute the gradients via back propagation, which is easy to implement and works well in practice. Experiments confirm several potential functions of Lmser and demonstrate its promising performance in various applications. Our contributions are summarized as follows:

- We revisit Lmser net by implementing it on multiple layers of neural networks. Our implementation can train the deep Lmser networks stably and effectively.
- Experiments are conducted on image reconstruction, generation, associative recall, and classification, in comparisons with AE as a baseline. The results not only confirm that Lmser works as indicated in the original paper, but also demonstrate its promising performance in various applications.

2 Related Work

2.1 Networks with Symmetrically Weighted Connections

A stack of two-layer restricted Boltzmann machines (RBMs) with symmetrically weighted connections was used in [6] to pretrain an AE that has multiple hidden layers in a one-by-one-layer way. AE consisting of multiple layers may have a slow convergence speed when optimizing its weights because the gradients vanishes as it propagates from the last layer of decoder to early layers in encoder. The pretrain can help to remedy this problem. The stacked RBMs worked well in dimensionality reduction.

2.2 Networks with Symmetrically Skip Connections

One recent typical example network architecture with symmetrically skip connections is the U-Net [11]. It consists of a contracting path as encoder and an expansive path as decoder, and both paths form a U-shaped architecture. The feature map from each of the layer of the contracting path was copied and concatenated with the symmetrically corresponding layer in the expansive path. Such skip connections can be regarded as a type of sharing the encoder neuron values with the correspondingly paired decoder neuron. Experiments in [11] demonstrated that U-Net worked very well for biomedical image segmentation. Such skip connections were also adopted in deep RED-Net [10]. Different from U-Net, it directly adds old feature map with present top-down signal. Feature map must be in the same size, so it only add skip connections for specific layers. It has been shown in [10] that deep RED-Net worked well in super-resolution image restoration. Moreover, with the appearance of ResNet [4] and DenseNet [7], deep neural networks with skip-connections become very popular and showed impressive performance in various applications.

3 A Brief Review of Lmser

The Lmser self-organizing net was proposed in [13,14] based on the principle of Least Mean Square Error Reconstruction (LMSER) of an input pattern. An example of Lmser architecture is demonstrated in Fig. 1(a). When Lmser net consists of multiple layers, each neuron z_k in the k-th layer receives both bottom-up signal y_k from the lower layer and top-down signal u_k from the upper layer, and is activated by their summation.

The Lmser net works in two phases, i.e., perception and learning. In the perception phase, the input pattern X triggers the dynamic process by passing the signals up from the bottom layer, while simultaneously the signals in the upper layers will be passed down to the lower layers. It has been proved that the process will converge into an equilibrium state [13]. The top-down signal to the input layer is regarded as reconstruction of the input. In the learning phase, the parameters are updated by minimizing the mean square error between the input and reconstruction. Given by Eq. (5a) in [14], the loss function for Lmser learning is:

$$J = \frac{1}{2}E(\|\overrightarrow{x} - W_1^T\overrightarrow{z_1}\|^2) \tag{1}$$

where $E(\cdot)$ denotes the expectation, W_1 is the weights of the first layer, $z_1 = s(y_1+u_1)$ is the activity of the first layer neurons which receive both the bottom-up signals y_1 and the top-down signals u_1. As given by Eqs. (6a)–(8b) in [14], the gradients to update the network parameters are restated below:

$$\begin{aligned}
&\text{for} \quad k=1, \quad \varepsilon_{i0} = x_i - u_{i0}, \varepsilon_{i1} = y_{i1} - y_{i1}^r, \quad \frac{\partial J}{\partial w_{pq1}} = \varepsilon_{q0}z_{p1} + s'_{p1}\varepsilon_{p1}x_q \\
&\text{for} \quad k \geq 2, \quad \varepsilon_{ik} = \sum_{j=1}^{n_{(k-1)}} \varepsilon_{j(k-1)}w_{ijk} \\
&\qquad\qquad\qquad \frac{\partial J}{\partial w_{pqk}} \approx s'_{q(k-1)}\varepsilon_{q(k-1)}z_{pk} + s'_{pk}\varepsilon_{ik}z_{q(k-1)}
\end{aligned} \tag{2}$$

4 Methods

4.1 Revisit Lmser and Implement It on Multiple Fully-Connected Layers

In order to further study the features and functions of Lmser, we need to implement it on multiple fully-connected layers. There is one technical challenge that it is hard to exactly implement the dynamic process in the perception phase and it may be time consuming to reach the convergent state. To overcome this problem, we propose an effective and simple way to approximate the dynamic process.

As shown in Fig. 1(b), we update the neurons in a layer-by-layer way. At the very beginning, a signal vector x is placed at the input layer into the network, and then it will trigger the bottom-up signal propagation through the layers one by one. In the first bottom-up pass, there is no input signal for the top layer, so at the i-th layer, the top-down signals from $(i+1)$-th layer to the i-th layer are

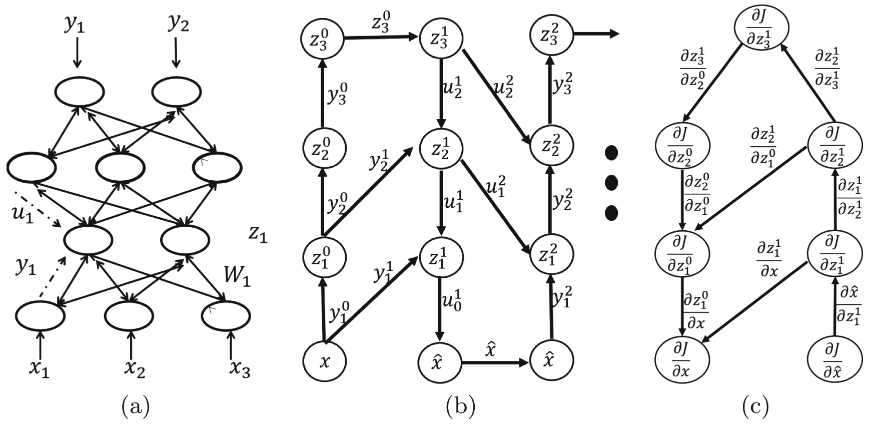

Fig. 1. (a) The architecture of multi-layer Lmser, where connection is bidirectional and symmetric, u_k is the top-down signal, y_k is the bottom-up signal, and each neuron is output as a sigmoid activation, i.e., $z_k = s(u_k + y_k)$, where $u_k = {W_{k+1}}^T z_{k+1}$, $y_k = W_k z_{k-1}$. (b) The reflection implementation, where z_k^t denotes the value of z_k at time t. (c) The calculation of gradients during back propagation.

initialized to be zero. After the first bottom-up pass, all neurons are given activity values. Then, the first top-down pass propagates the signals backwards and the neuron activation at the i-th layer is calculated as a sigmoid of the summation of u_i and y_i. In analogy to light reflections, we call such one bottom-up and one top-down pass as one reflection. In this way, bottom-up signal and top-down signal can be updated alternatively until they become stable. In practice, we use the Rectified Linear Units (ReLU) as the neuron activating function, and we find that instead of reaching the stable state of the updating process, one reflection followed by learning phase works well for the whole Lmser learning.

Another difference in our implementation from the original Lmser net is the learning rule. In the learning phase, for the calculation of gradient, instead of directly using Eq. (2), we compute it via an approximate back propagation. Such implementation enables us to use the available computational platform and back propagation library efficiently, and has been shown to work well in practice.

$$\frac{\partial J}{\partial W_k} = \frac{\partial J}{\partial z_k^1}\frac{\partial z_k^1}{\partial W_k} + \frac{\partial J}{\partial z_{k-1}^1}\frac{\partial z_{k-1}^1}{\partial W_k}$$

$$\frac{\partial z_k^1}{\partial W_k} = \frac{\partial z_k^1}{\partial z_{k-1}^0}\frac{\partial z_{k-1}^0}{\partial W_k} + \frac{\partial z_k^1}{\partial z_{k+1}^1}\frac{\partial z_{k+1}^1}{\partial W_k}, \quad \frac{\partial z_{k-1}^1}{\partial W_k} = \frac{\partial z_{k-1}^1}{\partial z_{k-2}^0}\frac{\partial z_{k-2}^0}{\partial W_k} + \frac{\partial z_{k-1}^1}{\partial z_k^1}\frac{\partial z_k^1}{\partial W_k}$$

$$\frac{\partial J}{\partial W_k} = \left(\frac{\partial J}{\partial z_k^1} + \frac{\partial J}{\partial z_{k-1}^1}\frac{\partial z_{k-1}^1}{\partial z_k^1}\right)\frac{\partial z_k^1}{\partial z_{k+1}^1}\frac{\partial z_{k+1}^1}{\partial W_k} \tag{3}$$

When we set reflection $T = 1$, as shown in Eq. (3), we can decompose $\frac{\partial J}{\partial W_k}$ to sum of multiplications of factors in the form of $\frac{\partial J}{\partial z_k^t}$, $\frac{\partial z_{k-1}^1}{\partial z_k^1}$, and $\frac{\partial z_l^t}{\partial W_k}$, where $\frac{\partial z_l^t}{\partial W_k}$ can be further decomposed or easily computed. Finally, $\frac{\partial J}{\partial W_k}$ is composed of factors in the form of $\frac{\partial J}{\partial z_k^t}$, $\frac{\partial z_{k-1}^1}{\partial z_k^1}$, $\frac{\partial z_k^1}{\partial z_{k-1}^0}$, and $\frac{\partial z_k^0}{\partial z_{k-1}^0}$, which can be calculated effectively in the process of back propagation, as shown in Fig. 1(c).

4.2 Jointly Supervised and Unsupervised Lmser Learning

In Lmser, each input pattern X can be recognized by a label Y_L output at the top layer, which plays the same role as a classifier. Thus, for the input with labels, Lmser can be implemented jointly both in an unsupervised manner to reduce the reconstruction error at the bottom layer and in a supervised way by minimizing the discrepancy between the predicted label Y_L and the true label. The reconstruction error will push the network to do self-organizing, which helps network to learn structure information of input and facilitates concept abstracting and formation at the top domain. The self-organizing learning directed by reconstruction error plays the role of regularization, which help network to prevent over fitting and make classification be more robust such as defense against adversarial attacks.

In perception phase, the implementation of dynamic process is the same as in Sect. 4.1, but with two parts computed at the top layer, i.e., one for predicted labels and the other as input to the decoder for top-down reconstruction. In learning phase, the loss function includes two terms for reconstruction and classification separately:

$$J = \frac{1}{2}E(\|\overrightarrow{x} - W_1^T \overrightarrow{z_1}\|^2) + L(f(x), y) \tag{4}$$

where the additional term $L(f(x), y)$ measures the error between the Lmser predicted $f(x)$ and the given label y.

5 Experiments

In this section, we demonstrate the effectiveness and strengths of the deep Lmser learning by some promising results on image recognition, reconstruction, generation, and associative recall. We use Lmser(un) to denote the Lmser network trained only in the unsupervised manner, use Lmser(un-n) to denote the Lmser(un) by removing DPN, use Lmser(sup) to denote the Lmser network trained jointly in the supervised and unsupervised manner.

5.1 Datasets and Experimental Settings

We evaluate Lmser on two benchmark datasests: MNIST [9] and Fashion-MNIST [12].

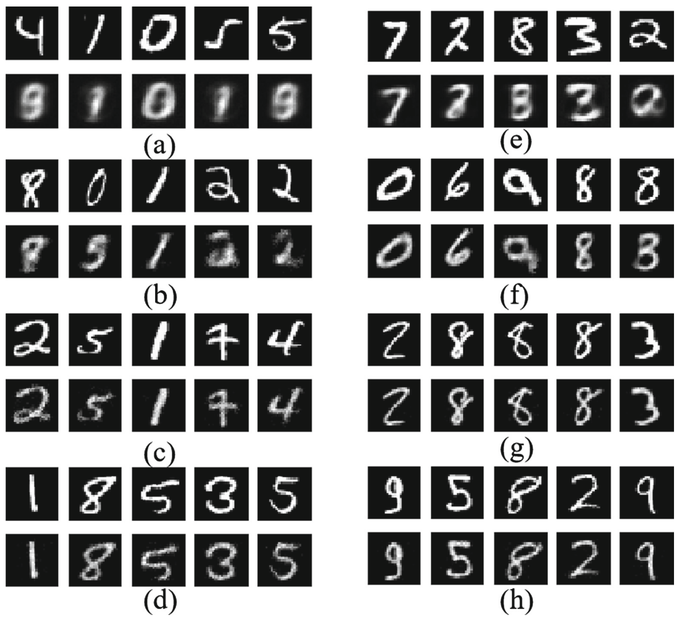

Fig. 2. Examples of reconstructed images on MNIST. (a) AE, $\tau = 500$; (b) Lmser(un-n), $\tau = 500$; (c) Lmser(un), $\tau = 500$; (d) Lmser(sup), $\tau = 500$; (e) AE, $\tau = 5000$; (f) Lmser(un-n), $\tau = 5000$; (g) Lmser(un), $\tau = 5000$; (h) Lmser(sup), $\tau = 5000$. (τ: the number of training iterations.)

- The MNIST constains $60,000$ training images and $10,000$ testing images. The images are handwritten digits from 250 people. Each picture in the data set consists of 28×28 pixels, each of which is represented by a gray value.
- Fashion-MNIST (F-MNIST) is a MNIST-like dataset which shares the same image size and structure of training and testing splits. F-MNIST is served as a replacement for the original MNIST dataset for benchmarking machine learning algorithms.

For MNIST and F-MNIST, we use 4 layers with the numbers of neurons $[10, 100, 300, 784]$ for Lmser based on fully-connected layers. We train every network model with Adam (Adaptive moment estimation) optimiser method [8]. In the training process, the batch size is set to be 50 and the adaptive learning rate is set to be 0.01 with decay rate 0.9999.

5.2 Reconstruction

We evaluate the reconstruction performance on MNIST dataset and F-MNIST dataset. Lmser is a development of AE, thus for comparisons, AE is used as a baseline. Detailed reconstruction errors are summarized in Table 1. Examples of the reconstructed results are given in Fig. 2 for different number of training iterations.

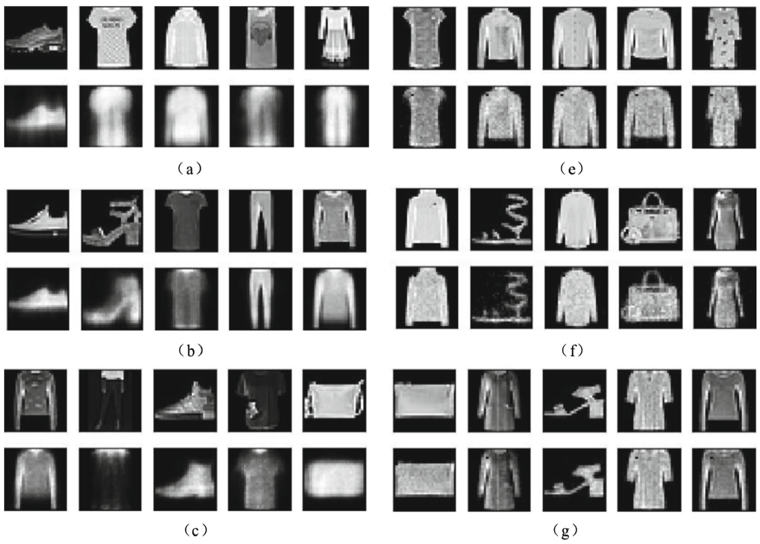

Fig. 3. Examples of reconstructed images on Fashion-MNIST. (a–c) AE, $\tau = 1000, 5000, 10000$; (e–g) Lmser(un), $\tau = 1000, 5000, 10000$. (τ: the number of training iterations)

Table 1. Reconstruction error on MNIST, where τ denotes the number of training iterations.

Model	$\tau = 500$	$\tau = 5000$	$\tau = 20000$
AE	0.071	0.036	0.020
Lmser(un-n)	0.030	0.021	0.016
Lmser(un)	0.0067	0.0018	0.0006
Lmser(sup)	0.0078	0.0025	0.0007

It can be observed that Lmser converges faster with smaller reconstruction errors than AE, and Lmser(un) is better than Lmser(un-n), which suggests that shortcuts between paired neurons play a significant role in reconstruction performance. Lmser(sup) is slightly worse than Lmser(un), but still much better than Lmser(un-n) and AE, indicating that there is a trade-off between the classification and reconstruction. Moreover, we also test the reconstruction performance on F-MNIST. Examples are shown in Fig. 3. Obviously, Lmser converges faster and is better than AE, which is consistent to the observations from Fig. 2.

5.3 Recognition

In this section, we investigate the classification performance of Lmser(sup), which is trained jointly in supervised manner and unsupervised manner. The classification accuracies of Lmser(sup) and a fully-connected feedforward network (FCN)

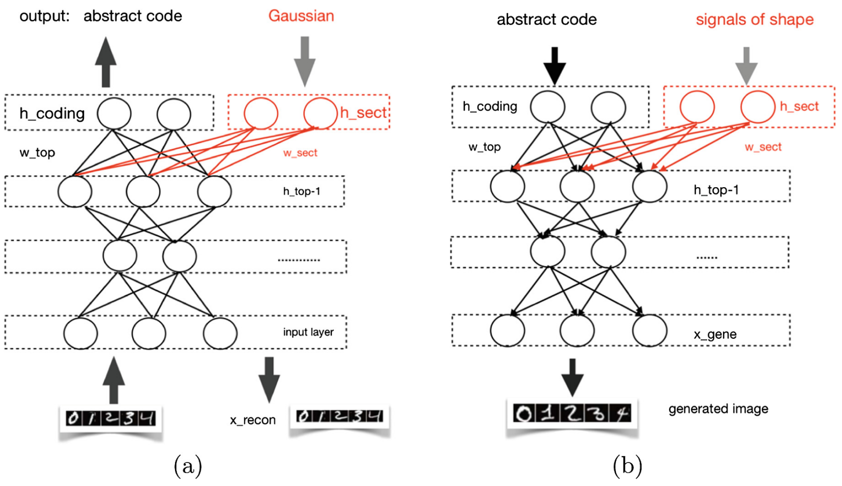

Fig. 4. Structure of supervised Lmser for image generation: (a) training stage; (b) generation stage

are comparably high, both over 98%. When there exist adversarial attacks at intensity level 0.3 from the Fast Gradient Sign Method (FGSM) [3], a well-known adversarial attack method, the classification accuracy of FCN drops down to 2.68%, while Lmser(sup) still gets 31.16%.

In this section, we investigate the performance of deep Lmser learning in image generation by manipulating the latent code to get different styles of handwritten digit numbers and clothes. In addition to the categorical coding units, we add two more hidden coding units h^1_{sect} and h^2_{sect} to the top layer of Lmser(sup) to get different styles of handwritten digits, as shown in Fig. 4. We assume that the two additional coding units are independent and follow Gaussian distributions. In practice, when we train the model Lmser(sup), the means of the two additional coding units are computed by encoder. Before they are fed into the decoder, we perturb them by a zero-mean Gaussian. When generating digital numbers, as shown in Fig. 4(b), the decoder will generate digital numbers according to not only the label by the categorical coding units but also randomly sampled style codes by the two additional hidden coding units. The style codes may also be assigned at specific values. Figure 5(e) show that when assigning the coding unit with different values, the digital number is gradually changing in different styles.

5.4 Generation

For the input patterns without labels, no labels can be used to guide the separation of categorical information and non-categorical styles. Lmser is still able to form a self-organized top coding domain, which preserves the neighbourhood relations and topological similarities. By manipulating one coding unit in the top

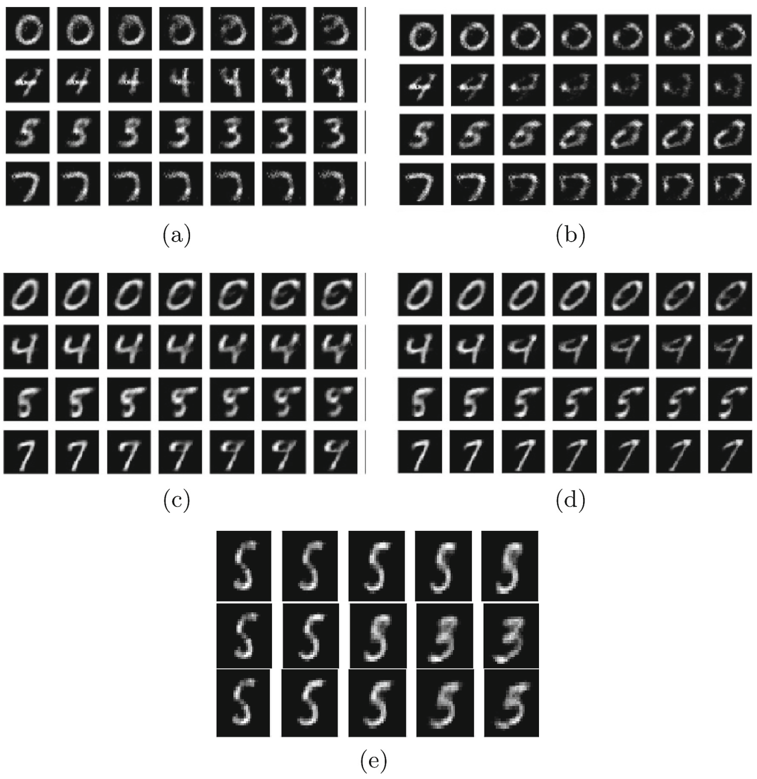

Fig. 5. Generated digital numbers by Lmser(un). There are 10 codes in the latent space, where Z_i represent the i-th coding unit. (a) manipulating Z_3; (b) manipulating Z_0; (c) manipulating Z_5; (d) manipulating Z_9. (e) Generated digital number by Lmser(sup-n) with changing two coding units. Top: manipulating h^1_{sect}; middle: manipulating h^2_{sect}; bottom: manipulating both the two units. It seems that the two units can control different shape changing.

layer with others fixed, we can observe pattern changing smoothly as the coding value varies, which indicates that the coding regions are well self-organized and clustered. With such property, we are able to synthesize images in a controllable way or in a creative way for novel synthesis and reasoning. Figure 5(c) shows that when specifying the coding unit with values from 0 to 10, the digital number is gradually changed to have a circle created over the head, while in Fig. 5(d) another coding unit seems to control the tilt degree of generated images.

5.5 Association

For the incomplete input, Lmser is able to recover the missing part by associative memory from the observed part. This function is related to tasks such as associative recall of recover the whole image with some key parts blocked.

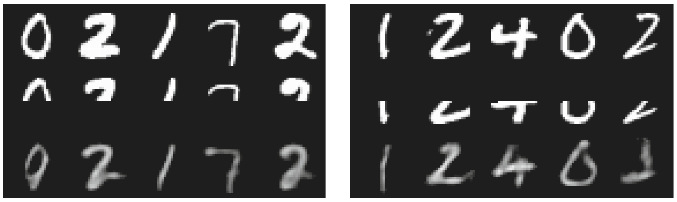

Fig. 6. Examples of associative recall from partial mnist images. (top) the ground-truth images; (middle) partial images with half blocked; (bottom) associative recall by Lmser(un-n).

Specifically, the observed part of the image is fed into the Lmser network, and triggers the bottom-up signals passing to the top layers. Then, the activated neurons passing the top-down signals back to the bottom layer to give a complete image, which actually recovers unobserved part based on what has been learned by the network layers. The sharing connection weights by both directions of the links between the consecutive layers enables Lmser to catch invertible structures under input patterns for restoring these structures under partial input.

We train the Lmser net on the MNIST dataset, and then feed the masked images into the model for the output of reconstructed complete images. In practice, we have found that shortcuts between paired neurons is helpful in reconstruction, but not so beneficial in associative memory, because it can also pass hole to the low layers of decoder when it pass the detailed information from encoder to decoder. Therefore, we only present the results by the Lmser net without using the paired neurons for both encoder and decoder in Fig. 6. Examples of the half blocked images are given in the middle row in Fig. 6, and the corresponding reconstructed complete digits from the associative memory by Lmser are shown in the bottom row, which are similar to the ground-truth images in the top row. The results demonstrate that Lmser is promising for this task of associative recall from partial input.

6 Conclusion

In this paper, we have revisited the Lmser network for the practical implementation of multiple layers, and confirmed that several of its potential functions indeed work effectively via experiments on image recognition, reconstruction, associative memory, and generation. Experiments demonstrate that Lmser not only works as preliminarily discussed in the original paper, but also is promising in various applications. It deserves further investigations on Lmser in the future for its improvements and comparisons with state-of-the-art methods in many real applications.

Acknowledgement. This work was supported by the Zhi-Yuan Chair Professorship Start-up Grant (WF220103010), and Startup Fund (WF220403029) for Youngman Research, from Shanghai Jiao Tong University.

References

1. Ballard, D.H.: Modular learning in neural networks. In: Proceedings of the Sixth National Conference on Artificial Intelligence, AAAI 1987, vol. 1, pp. 279–284 (1987)
2. Bourlard, H., Kamp, Y.: Auto-association by multilayer perceptrons and singular value decomposition. Biol. Cybern. **59**(4–5), 291–294 (1988)
3. Goodfellow, I., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: International Conference on Learning Representations (2015). http://arxiv.org/abs/1412.6572
4. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
5. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Comput. **18**(7), 1527–1554 (2006)
6. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006)
7. Huang, G., Liu, Z., Weinberger, K.Q.: Densely connected convolutional networks. CoRR abs/1608.06993 (2016). http://arxiv.org/abs/1608.06993
8. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. CoRR abs/1412.6980 (2014). http://arxiv.org/abs/1412.6980
9. LeCun, Y., Cortes, C., Burges, C.: MNIST handwritten digit database, vol. 2. AT&T Labs (2010). http://yann.lecun.com/exdb/mnist/
10. Mao, X., Shen, C., Yang, Y.B.: Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In: Advances in Neural Information Processing Systems, pp. 2802–2810 (2016)
11. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
12. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. CoRR abs/1708.07747 (2017)
13. Xu, L.: Least MSE reconstruction for self-organization: (i)&(ii). In: Proceedings of 1991 International Joint Conference on Neural Networks, pp. 2363–2373 (1991)
14. Xu, L.: Least mean square error reconstruction principle for self-organizing neural-nets. Neural Netw. **6**(5), 627–648 (1993)
15. Xu, L.: An overview and perspectives on bidirectional intelligence: Lmser duality, double IA harmony, and causal computation. IEEE/CAA J. Autom. Sin. **6**(4), 865–893 (2019). https://doi.org/10.1109/JAS.2019.1911603