



Deep CNN Based Lmsr and Strengths of Two Built-In Dualities

Wenjing Huang¹ · Shikui Tu¹ · Lei Xu¹

Accepted: 31 August 2020

© Springer Science+Business Media, LLC, part of Springer Nature 2020

Abstract

Least mean square error reconstruction for the self-organizing network (Lmsr) was proposed in 1991, featured by a bidirectional architecture with several built-in natures. In this paper, we developed Lmsr into CNN based Lmsr (CLmsr), highlighted by new findings on strengths of two major built-in natures of Lmsr, namely duality in connection weights (DCW) and duality in paired neurons (DPN). Shown by experimental results on several real benchmark datasets, DCW and DPN bring to us relative strengths in different aspects. While DCW and DPN can both improve the generalization ability of the reconstruction model on small-scale datasets and ease the gradient vanishing problem, DPN plays the main role. Meanwhile, DPN can form shortcut links from the encoder to the decoder to pass detailed information, so it can enhance the performance of image reconstruction and enables CLmsr to outperform some recent state-of-the-art methods in image inpainting with irregularly and densely distributed point-shaped masks.

Keywords Lmsr · Duality in connection weights · Duality in paired neurons

1 Introduction

Studies on bidirectional neural networks can be traced back to auto-association in the 1980s [4]. Typically, three-layer networks were used to make auto-association to learn inner representations of observed signals [1]. Generally, the bottom-up part mapping from the input layer X to the topmost coding layer Y , i.e., $X \rightarrow Y$ is called encoder, while the top-down direction to recover the input from the topmost coding layer is called decoder, i.e., $Y \rightarrow \hat{X}$, where \hat{X} represents a reconstruction of X . The two parts form an autoencoder (AE) network. The underlying principle of AE implements approximately an identical mapping $X \rightarrow \hat{X}$ by a direct cascading of $X \rightarrow Y$ and $Y \rightarrow \hat{X}$, which forms a simple circle.

Another typical example was least mean square error reconstruction (Lmsr) self-organizing network that was first proposed in 1991 [23,24]. Proceeded beyond AE, Lmsr is featured by a bidirectional architecture with several built-in natures, for which readers can

✉ Lei Xu
leixu@sjtu.edu.cn

¹ Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China

refer to Table I in Ref. [25]. Two major natures are duality in connection weights (DCW) and duality in paired neurons (DPN). DCW refers to using the symmetric weights in corresponding layers in encoder and decoder such that the direct cascading by AE is further enhanced. DPN refers to constraining the neurons in corresponding layers between the encoder and decoder to be the same.

The purpose of designing such a symmetrical structure in Lmser is to approximate identity mapping per two consecutive layers simply through $W^T W \approx I$, where $W^T W = I$ holds only if the weight matrix W is orthogonal. In image reconstruction, the network is trained to form an identity mapping, and the two dualities aim to minimize the error between bottom-up input and top-down prediction at every layer. Meanwhile, DCW and DPN can ease the gradient vanishing problem by adding some extra terms to the gradients of parameters.

Although proposed early in 1991, only one-layer Lmser was implemented due to the limitations on both computing power and data amount in the 1990s. It was shown theoretically and experimentally that a neuron in Lmser net behaved like a feature detector in the cortical field during the learning process [24]. However, the advantage of DPN could not be demonstrated in one-layer Lmser, and several potential functions of Lmser in deep network structure were speculated in [23] but have not been investigated.

In this paper, we present an effective and practical implementation of multilayer Lmser and develop it into a convolutional neural network (CNN) based structure (CLmser) for image data. In the perception phase of Lmser, the signals propagate forwardly and backwardly in a dynamic process which is hard to implement in practice. To overcome this problem, we decouple the dynamic process in the perception phase approximately into recurrent iterations of a bottom-up pass with encoder-to-decoder skip connections and a top-down pass with decoder-to-encoder feedback connections. Then the perception phase is implemented by updating the neurons in a layer-by-layer way and alternatively update bottom-up signals and top-down signals for multiple iterations. Meanwhile, as a result of the above decoupling process, we can approximate Lmser parameter learning by backpropagation [20] to suit for our implementation.

Moreover, we investigate the strengths of DCW and DPN in different aspects by building intermediate models and evaluating their performances in different scenarios. When used alone in image reconstruction, both DCW and DPN can help to improve the generalization ability of the reconstruction model on small-scale datasets and accelerate the convergence speed of the network. The reason is that DPN can pass details from encoder to decoder, and DCW is designed to approximate identity mapping per two consecutive layers. When the two dualities are used together in image reconstruction, both of them can play their positive roles while DPN plays the main role. However, in image inpainting, DCW plays a negative role while DPN plays a positive role and enables the network to outperform some recent state-of-the-art methods [16,26,27] in inpainting with irregularly and densely distributed point-shaped masks.

The contributions of this paper are threefold. Firstly, it is the first time that deep Lmser learning is effectively and stably implemented on multiple layers and further developed into CNN based Lmser (CLmser). Secondly, we found that both DCW and DPN can ease the gradient vanishing problem during the process of training deep networks and enables the Lmser network to perform more robust on the small-scale dataset. Thirdly, it is also found that DPN can enhance the performance of image inpainting.

2 Related Work

One major nature of Lmsr is weights sharing, i.e., DCW. Weights sharing was also used in stacked Restricted Boltzmann machines (RBMs) [9]. A stack of two-layer RBM with symmetrically weighted connections was used to pre-train a multilayer AE in a layer-by-layer way. The pretraining can help to remedy the gradient vanishing problem in the AE. The AE unrolled from a stack of pre-trained RBMs worked well in dimensionality reduction [9].

DPN can form shortcut connections between neurons symmetrically placed in the encoder and decoder. Similar shortcut connections were also found in recent works. One typical example is the U-Net [19]. It consists of a contracting path as the encoder and an expansive path as the decoder, and both paths form a U-shaped architecture. The feature map from each layer of the contracting path was copied and concatenated with the symmetrically corresponding layer in the expansive path. Experiments in [19] demonstrated that U-Net worked well for biomedical image segmentation. Such skip connections were also adopted in deep RED-Net [18], which directly adds feature maps in the encoder with corresponding feature maps in the decoder by constraining their feature maps to be the same size. It has been shown in [18] that deep RED-Net worked well in image super-resolution reconstruction. Moreover, U-Net like architectures were used in [11] for image transformations, and in [16] with all convolutional layers replaced with partial convolutional layers for image inpainting on irregular holes. Instead of adding shortcut connections symmetrically between encoder and decoder, shorter connections in ResNet [7] and DenseNet [10] were added between the layers close to the input and those close to the output, which makes parameter learning more accurate and efficient.

Another nature of the Lmsr is the dynamic process in the perception phase for updating neurons according to both top-down signals and bottom-up signals due to DPN. Recently, a deep predictive coding network (PCN) [6] was proposed for object recognition, which also included local recurrent processing to update neurons. PCN includes both feedback and feedforward connections, where the feedback connections carry top-down predictions and the feedforward connections carry bottom-up errors of predictions. The two connections enable PCN to update the adjacent layers locally and recurrently to refine representations towards minimization of layer-wise prediction errors.

3 Preliminaries

As demonstrated in Fig. 1, the Lmsr architecture consists of symmetric neurons and symmetric weights, due to DCW and DPN. As a result of DCW, Lmsr cascades the layers via bidirectional connections between every two consecutive layers to approximate identity mapping simply through $W^T W \approx I$, where $W^T W = I$ holds only if W is an orthogonal matrix. By DPN, each neuron z_k is activated by both the bottom-up signal y_k from the lower layers and the top-down signal u_k from the upper layers. It is noticed that Lmsr net degenerates back to AE by removing DCW and DPN.

The Lmsr net works in two phases, i.e., perception phase and learning phase. In the perception phase, the input 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 [24]. After the process reached its stable state, the top-down signal to the input layer u_0 is regarded as a reconstruction of the input x . The discrepancy between u_0 and x will trigger the learning

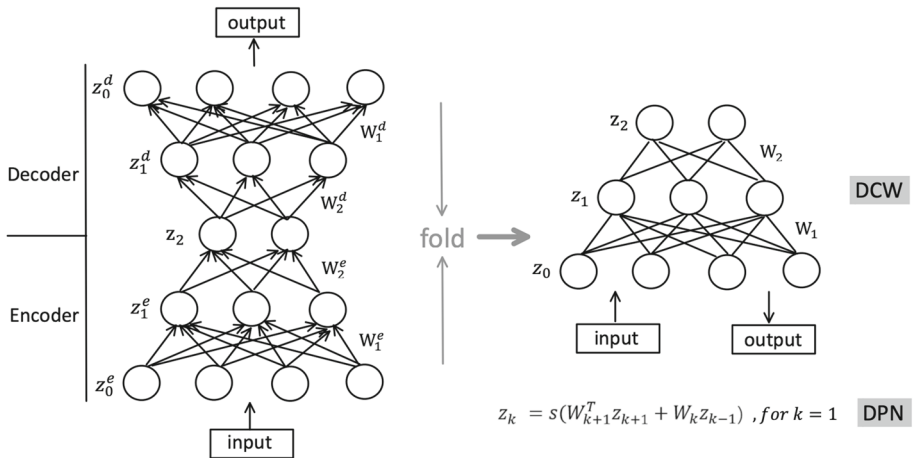


Fig. 1 Visualization of the two dualities: DCW and DPN. By DCW, Lmscr cascades the layers via bidirectional connections between every two consecutive layers. By DPN, each neuron z_k is activated by both the bottom-up signal $y_k = W_k z_{k-1}$ and the top-down signal $u_k = W_{k+1}^T z_{k+1}$

phase to reduce the difference between u_0 and x . In the learning phase, the parameters are updated by minimizing the mean square error between the input and reconstruction. Given by Eq. (5a) in [24], the loss function for Lmscr learning is:

$$J = \frac{1}{2} E \left(\|\vec{x} - W_1^T \vec{z}_1\|^2 \right), \tag{1}$$

where $E(\cdot)$ denotes the expectation, W_1 is the weight matrix in the first layer, $z_1 = s(y_1 + u_1)$ are the activities of the neurons in the first layer which receive both the bottom-up signals y_1 and the top-down signals u_1 . As given by Eqs. (6a)–(8b) in [24], the gradients to update the network parameters were restated below:

$$\begin{aligned}
 -\frac{\partial J}{\partial W_{pqk}} &= \sum_{i=1}^{n_k} \varepsilon_{ik} \frac{\partial z_{ik}}{\partial W_{pqk}} + \sum_{i=1}^{n_{k-1}} \varepsilon_{i(k-1)} \frac{\partial z_{i(k-1)}}{\partial W_{pqk}} \\
 &\approx s'_{pk} \varepsilon_{pk} z_{q(k-1)} + s'_{q(k-1)} \varepsilon_{q(k-1)} z_{pk},
 \end{aligned} \tag{2}$$

where $\varepsilon_{ik} = \sum_{j=1}^{n_{k-1}} \varepsilon_{j(k-1)} W_{ijk}$, W_{ijk} is the weight connecting the i -th unit on $(k-1)$ -th layer and j -th unit on k -th layer. z_{ik} is the activity of i -th unit on k -th layer, n_k is the number of units on the k -th layer, and $s(\cdot)$ is a sigmoid function.

4 Method

4.1 CNN Based Lmscr (CLmscr)

Due to the lack of computing resources and big data in the early 1990s, the Lmscr was implemented with only one hidden layer when it was proposed. Instead of simply considering the original fully-connected layers in [24], we develop Lmscr into deep convolutional Lmscr (CLmscr) to further investigate its dualities and to deal with image data. For the deep implementation of Lmscr, we need to handle the following issues:

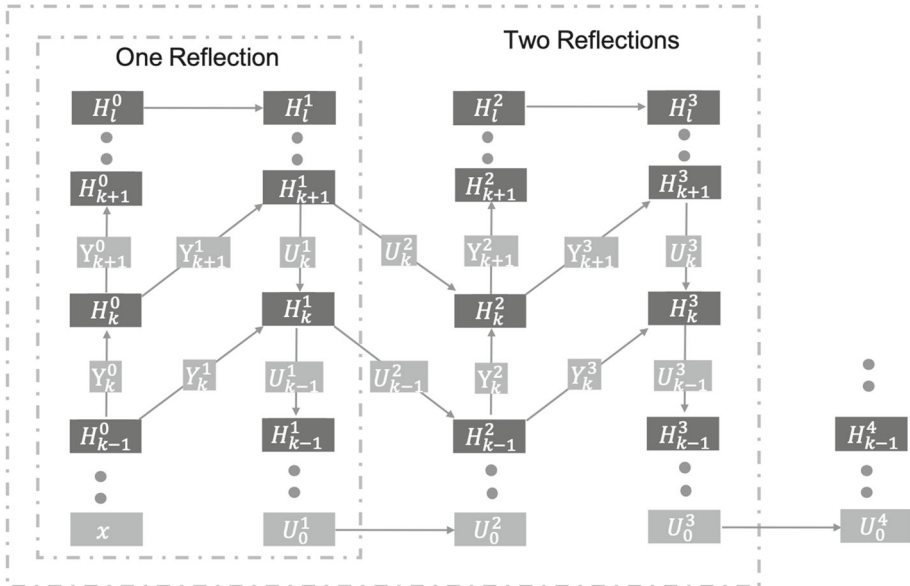


Fig. 2 The approximate deep implementation of Lmsr. Signals flow in the network by multiple bottom-up passes and top-down passes. One bottom-up pass and one top-down pass compose one reflection. The perception phase is implemented by repeating multiple reflections

1. In the perception phase, it is hard to implement the exact dynamic process in multiple layers because neurons are highly coupled and it may be time-consuming to reach convergence.
2. In the learning phase, we need to modify the learning rule of the original Lmsr to suit for our implementation.

To handle the first issue, we propose an effective and simple approximation for the dynamic process. As shown in Fig. 2, we decouple the bidirectional propagation into propagating signals by bottom-up pass and top-down pass alternatively and update the neurons in a layer-by-layer way. We call one bottom-up pass and one top-down as one reflection and repeat multiple reflections to approximate the convergence process of the perception phase.

At the beginning $t = 0$, the image is fed into the bottom layer of the network, which triggers the bottom-up signal propagation layer by layer. In the first bottom-up pass, there is no input signal on the top layer, thus the top-down signals from the higher layers are initialized to be zero and the neurons are initialized as follows:

$$t = 0, \forall i > 0, \text{ there exist: } U_i^t = 0, H_0^t = x, H_i^t = f(Y_i^t) = f(W_i H_{i-1}^t), \quad (3)$$

where H_i^t denotes the values of $n_i \times 1$ neurons in the i -th layer at time t , Y_i^t denotes the $n_i \times 1$ bottom-up signals, and U_i^t denotes the $n_i \times 1$ top-down signals. W_i is the $n_i \times n_{i-1}$ weight matrix, and $f(\cdot)$ is a nonlinear activation function.

After the first bottom-up pass, all neurons have been initialized. According to the scheme in Fig. 2, the top-down signals and bottom-up signals are alternatively updated by the following

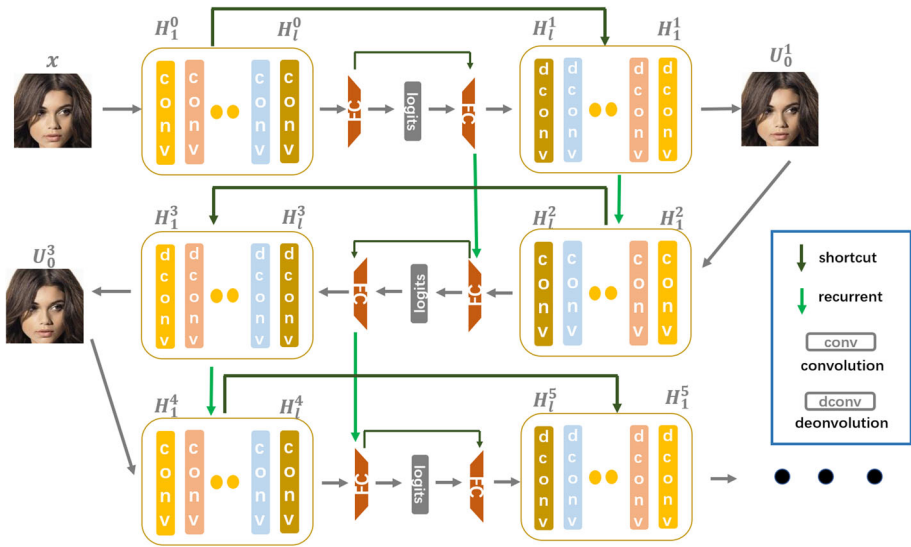


Fig. 3 Architecture of the CNN based Lmser networks (CLmsr). Blocks in the same color share the same weights. The bottom-up signals are computed by convolution while the top-down signals are computed by deconvolution. DCW is implemented by sharing weights between convolution and deconvolution block while DPN is implemented by feedback and shortcut links

rules:

$$t = 2k, k \geq 1, \text{ bottom-up: } H_i^t = f(Y_i^t + U_i^{t-1}), Y_i^t = W_i H_{i-1}^t; \tag{4}$$

$$t = 2k - 1, k \geq 1, \text{ top-down: } H_i^t = f(Y_i^{t-1} + U_i^t), U_i^t = W_{i+1}^\top H_{i+1}^t. \tag{5}$$

Note that in the perception phase of original Lmsr, each neuron receives both top-down and bottom-up signals simultaneously, as given by:

$$H_i^t = f(Y_i^t + U_i^t), Y_i^t = f(W_i H_{i-1}^t), U_i^t = f(W_{i+1}^\top H_{i+1}^t) \tag{6}$$

Comparing our implementation by Eqs. (4)–(5) with the original dynamic process by Eq. (6), it can be noticed that the neuron values H_i are computed by alternatively updating top-down and bottom-up signals. In the bottom-up pass, we fix U_i at the previous state U_i^{t-1} and update Y_i from lower layers, while in the top-down pass, we fix Y_i at the previous state Y_i^{t-1} and update U_i from upper layers. In practice, this is an efficient approximation.

For CNN based Lmsr, the bottom-up signal is calculated by convolution, while the top-down signal is computed via deconvolution. Based on the decoupling process of Fig. 2, the architecture of CLmsr is shown in Fig. 3. To enable DCW, we force the convolution and deconvolution block to share the same weights. The DPN nature is featured by two types of links. One is the skip connection from the encoder to the decoder, and the other is the feedback link from the decoder to the encoder. It should be noted that the information fusion between Y_i^t and U_i^t here is implemented as a simple summation. In general, other fusion operations are possible. For example, in U-Net, the bottom-up signal and top-down signal are concatenated first and then the neuron values are calculated via a convolution operation on the concatenated signals [19].

For the second issue, instead of using Eqs. (6a)–(8b) in [24], we compute the gradients of parameters via backpropagating the errors through the backward path as indicated in the right

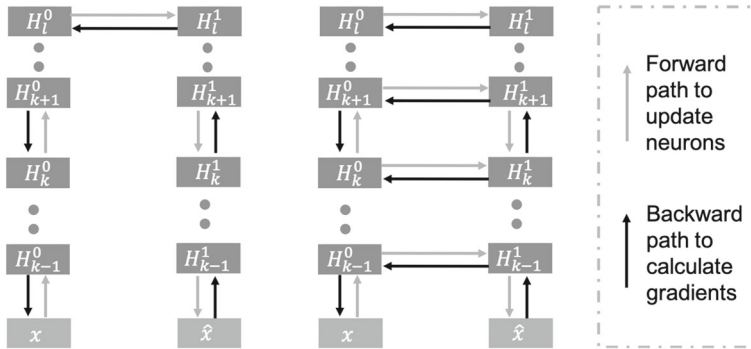


Fig. 4 The error back propagation path for calculating gradients of AE (left) or Lmser (right, demonstrated with one reflection). Compared with AE, Lmser has shortcuts for error propagation (the horizontal black arrows) to avoid gradient vanishing

Table 1 Description of the intermediate models

Model	Description
CLmser	CNN based Lmser
CLmser-w	CLmser without DCW
CLmser-n	CLmser without DPN
CAE	CLmser-w-n, CLmser without DCW nor DPN

subfigure of Fig. 4. The objective of the learning phase is still to minimize the loss function in Eq. (1). At the start of the training process, weight parameters are inaccurate, and thus the execution of the learning phase does not need to wait for the convergence of the perception phase. Experimental results show that implementing the perception phase by repeating one reflection works well in practice.

4.2 Dualities

To investigate the roles of DCW and DPN respectively, we build intermediate models in CLmser by removing DCW (denoted as “-w”) or removing DPN (denoted as “-n”) or removing both which degenerates back to AE. We use CLmsers to represent all the intermediate models. The intermediate models are described in Table 1 and repeated by one reflection in their perception phase, which is the same with AE. The learning rule of the intermediate models for calculating the gradients of parameters can be obtained by error backpropagation [20] through the backward paths defined by the models.

- For AE, as shown in the left subfigure of Fig. 4, when calculating gradients of parameters in the decoder, the reconstruction error propagates backwardly through a bottom-up path; when calculating gradients of parameters in the encoder, the error propagates backwardly through a bottom-up path cascaded by a top-down path. The parameters in the encoder and the decoder are all free and independent. If the network is deep, the process of training for AE may encounter the gradient vanishing problem.
- For Lmser-n, as shown in the right subfigure of Fig. 4, the parameters of the encoder and the decoder are constrained to be same due to the DCW, so that there are two error

propagation paths for calculating the gradients of parameters, which eases the gradient vanishing problem, especially for the weights in the lower layers of the encoder.

- For Lmser-w, as shown in the right subfigure of Fig. 4, shortcut connections are added between the encoder and the decoder, so the errors can be backpropagated through the shortcut links, which avoids the problem of gradient vanishing and leads to faster convergence of parameters.

To further demonstrate the contributions of DCW and DPN to easing gradient vanishing problem, we give the gradient calculation formula for each intermediate model. For all the models, we have:

$$\frac{\partial J}{\partial z_{ik}^d} = \sum_{j=1}^{n_{k-1}} \frac{\partial J}{\partial z_{j(k-1)}^d} s'_{j(k-1)} W_{jik}, \quad \frac{\partial J}{\partial z_{ik}^e} = \sum_{j=1}^{n_{k+1}} \frac{\partial J}{\partial z_{j(k+1)}^e} s'_{j(k+1)} W_{ij(k+1)}, \quad (7)$$

where W_{ijk} is the weight connecting the i -th unit on $(k - 1)$ -th layer and j -th unit on k -th layer, z_{ik}^e and z_{ik}^d represent the activities of i -th unit on k -th layer in the encoder and decoder stage respectively. J is the loss function of the models, which is defined in Eq. (1). $s(\cdot)$ is the activation function and $s'(\cdot)$ is the derivative of activation function.

- For AE, we have:

$$\frac{\partial J}{\partial W_{pqk}^d} = \sum_{i=1}^{n_{k-1}} \frac{\partial J}{\partial z_{i(k-1)}^d} \frac{\partial z_{i(k-1)}^d}{\partial W_{pqk}^d}, \quad -\frac{\partial J}{\partial W_{pqk}^e} = \sum_{i=1}^{n_k} \frac{\partial J}{\partial z_{ik}^e} \frac{\partial z_{ik}^e}{\partial W_{pqk}^e} \quad (8)$$

$$\frac{\partial z_{i(k-1)}^d}{\partial W_{pqk}^d} = s'_{i(k-1)} \delta_{ip} z_{qk}^d, \quad \frac{\partial z_{ik}^e}{\partial W_{pqk}^e} = s'_{ik} \delta_{iq} z_{p(k-1)}^e, \quad (9)$$

where δ_{ip} is dirac delta function, $\delta_{ip} = 1$ under the condition $i = p$, otherwise, $\delta_{ip} = 0$. Consequently, the gradients of parameters in AE can be calculated by the following rule:

$$\frac{\partial J}{\partial W_{pqk}^d} = \frac{\partial J}{\partial z_{p(k-1)}^d} s'_{p(k-1)} z_{qk}^d, \quad -\frac{\partial J}{\partial W_{pqk}^e} = \frac{\partial J}{\partial z_{qk}^e} s'_{qk} z_{p(k-1)}^e, \quad (10)$$

where $\frac{\partial J}{\partial z_{p(k-1)}^d}$ and $\frac{\partial J}{\partial z_{qk}^e}$ can be obtained from Eq. (7).

- For Lmser-n, we have:

$$-\frac{\partial J}{\partial W_{pqk}} = \sum_{i=1}^{n_{k-1}} \frac{\partial J}{\partial z_{i(k-1)}^d} \frac{\partial z_{i(k-1)}^d}{\partial W_{pqk}} \quad (11)$$

$$\frac{\partial z_{i(k-1)}^d}{\partial W_{pqk}^d} = s'_{i(k-1)} \delta_{ip} z_{qk}^d + s'_{i(k-1)} \sum_{j=1}^{n_k} \frac{\partial z_{jk}^d}{\partial W_{pqk}} W_{ijk}, \quad \frac{\partial z_{jk}^d}{\partial W_{pqk}} = s'_{jk} \frac{\partial u_{jk}^d}{\partial W_{pqk}} \quad (12)$$

where u_{jk}^d is the top-down signal to z_{jk}^d . Thus, the gradients in Lmser-n can be computed by:

$$\frac{\partial J}{\partial W_{pqk}} = \frac{\partial J}{\partial z_{p(k-1)}^d} s'_{p(k-1)} z_{qk}^d + \sum_{i=1}^{n_{k-1}} \frac{\partial J}{\partial z_{i(k-1)}^d} s'_{i(k-1)} \sum_{j=1}^{n_k} s'_{jk} \frac{\partial u_{jk}^d}{\partial W_{pqk}} W_{ijk} \quad (13)$$

- For Lmser, the difference compared with Lmser-n is that z_{ik}^d is activated by both top-down signal u_{ik}^d and bottom-up signal y_{ik}^d rather than single top-down signal in Lmser-n. Thus,

the only difference in the learning rule is:

$$\frac{\partial z_{jk}^d}{\partial W_{pqk}} = s'_{jk} \left(\frac{\partial u_{jk}^d}{\partial W_{pqk}} + \frac{\partial y_{jk}^d}{\partial W_{pqk}} \right) = s'_{jk} \left(\frac{\partial u_{jk}^d}{\partial W_{pqk}} + \delta_{jq} z_{p(k-1)}^e \right) \tag{14}$$

Therefore, the general formula for computing the gradients in Lmser is as follows:

$$\begin{aligned} -\frac{\partial J}{\partial W_{pqk}} &= \frac{\partial J}{\partial z_{p(k-1)}^d} s'_{p(k-1)} z_{qk}^d + \sum_{i=1}^{n_{k-1}} \frac{\partial J}{\partial z_{i(k-1)}^d} s'_{i(k-1)} \sum_{j=1}^{n_k} s'_{jk} \frac{\partial u_{jk}^d}{\partial W_{pqk}} W_{ijk} \\ &+ \sum_{i=1}^{n_{k-1}} \frac{\partial J}{\partial z_{i(k-1)}^d} s'_{i(k-1)} s'_{qk} z_{p(k-1)}^e \end{aligned} \tag{15}$$

As shown in Eq. (10), when calculating the gradients for parameters in the encoder in AE framework, $\frac{\partial J}{\partial W_k^e}$ and $\frac{\partial J}{\partial W_k^d}$ are in multiplication forms, which may lead to gradient vanishing in deep networks especially when activation function is not appropriate. It could be observed from Eq. (13) that when DCW is incorporated, a term is added to the gradient to ease the gradient vanishing problem. Furthermore, when DPN is incorporated, more terms are added to the gradient according to Eq. (15), which can improve the convergence speed of the network, as will also be demonstrated by experiments later.

5 Experiments

In this section, we conduct experiments on two tasks: image reconstruction and image inpainting. The detailed settings of network architectures and training procedures are described in Tables 2 and 3 respectively. We conduct the experiments on three dataset including STL10 [5] for image reconstruction and CelebA-HQ [12], Places2 [28] for image inpainting. The details of the datasets are listed below:

- STL10 [5]: A ten class image set with 500 training images and 800 testing images per class. We divide the training images into a training set and a test set in the ratio of 4: 1.
- CelebA-HQ [12]: A high-quality version of the human face dataset generated from CelebA [17]. We randomly split it into 27,000 images and 3000 images for training and testing respectively.
- Places2 [28]: A well-known real-world image dataset. It contains images of 365 scenes collected from the natural world. We train models on 180k subset of the original training set and test models on 10,000 images which are randomly chosen from the original test set.

To investigate the strengths of DCW and DPN in different scenarios, we conduct experiments as follows: (1) in Sect. 5.1, we investigate influences of DCW and DPN on the generalization ability of image reconstruction on the small-scale dataset; (2) in Sect. 5.2, we investigate influences of DCW and DPN on convergence speed of the reconstruction model; (3) in Sect. 5.3, we evaluate influences of DCW and DPN on image inpainting. To investigate the roles of DCW and DPN separately, the intermediate models described in Table 1 are all evaluated in each scenario. Moreover, several recent networks related to DCW and DPN are also listed for comparisons.

After reporting the experimental results, we analyze the effectiveness of DCW and DPN in each scenario. Furthermore, we summarize their strengths in Sect. 5.4 for clarity.

Table 2 Detailed configurations of CLmsers on different datasets

Dataset	d	Kernel	Channel	Stride
STL10	5	3	16, 32, 64, 128, 256	1, 2, 2, 2, 2
C&P	7	3	16, 32, 64, 128, 256, 256, 256	1, 2, 2, 2, 2, 2, 2

Common settings: activation function is ReLU, the number of reflections in perception phase is 1 (d : depth of network; C&P: CelebA-HQ & Places2)

Table 3 Training parameters of CLmsers on different datasets

Parameters	STL10	CelebA-HQ, Places2
Epochs	20	20
Learning rate	0.01	0.001
Optimization method	SGD&Adam	Adam
Batch size	100	16

Table 4 Reconstruction performance of the models on STL10 with varying training sample sizes, i.e., 400, 100, 10 (CL: CLmser; CL-w: CLmser-w; CL-n: CLmser-n)

Sample size (n)	Metric	CL	CL-w	U-Net [19]	CL-n	CAE
400	MSE	0.0060	0.0090	0.0105	0.0576	0.0779
	SSIM	0.8656	0.8284	0.8093	0.3405	0.2508
	PSNR	28.60	26.86	26.21	18.78	17.49
100	MSE	0.0067	0.0101	0.0107	0.0562	0.0782
	SSIM	0.8607	0.8188	0.7971	0.3663	0.2545
	PSNR	28.11	26.41	26.21	18.95	17.47
10	MSE	0.0069	0.0108	0.0106	0.0546	0.0847
	SSIM	0.8590	0.8124	0.8054	0.3566	0.2379
	PSNR	27.99	26.07	26.27	19.04	17.13

Bold indicates that it achieves the best performance for the corresponding metric. For l1 and MSE, the smaller the better; for SSIM and PSNR, the larger the better

5.1 Robustness

We first investigate the robustness of different models on small-scale dataset. Although it is relatively easier to collect big data such as face images nowadays, the data collection is still expensive in some domains, such as biology and health. Therefore, the generalization of a model trained on a small-scale dataset is a preferred feature.

We conduct experiments of image reconstruction on dataset STL10 [5]. To investigate the robustness of models with various sizes of the training set, we train models on datasets with different scales and evaluate their performance on the same test set. We use STL10- n to represent the dataset which contains n samples per class where n is set as 400, 100 and 10, respectively. To evaluate the quality of the restored images, we follow the previous image reconstruction works [16,18] by reporting Peak Signal-to-Noise Ratio (PSNR), Structural SIMilarity (SSIM) index [21], and Mean Square Error (MSE). We also take U-Net [19] into comparison, which has a similar architecture with CLmser-w. Here, U-Net, originally proposed for biomedical image segmentation, is modified for image reconstruction. All experimental results are summarized in Table 4.



Fig. 5 Examples of reconstructed images from models trained on STL10-100: (columns from left to right) the reconstruction by CAE, CLmser-n, CLmser-w, CLmser, the original input image

From Table 4, we can observe that CLmser performs best and the models incorporated with DPN generally perform better than the others as the training sample size decreases from 400 to 10, which indicates that DPN plays an important role in enhancing the performance of the image reconstruction. As discussed in Sect. 4.1, the DPN act as a shortcut from the encoder to the decoder, so the more detailed information in the low levels in the encoder can be directly passed to the corresponding layers of decoder, which can increase the restoration quality. As expected, the performances of U-Net and CLmser-w are similar, because they both include skip connections but without DCW. Examples of the outputs of image reconstruction are provided in Fig. 5.

By comparing the results of CAE and CLmser-n, we find that DCW can also enhance the performance of image reconstruction slightly. Theoretically, the reconstruction capability of CAE should not be worse than that of CLmser-n because DCW places an extra constraint over the weights. However, in this paper, we mainly focus on the generalization ability of models on small-scale datasets. When the samples are insufficient, CAE may meet the overfitting problem and get stuck in an unsatisfying local minimum. However, DCW enforces an approximate identity mapping by symmetric weights layer by layer, which can guide the network to reach a better local minimum. When the sample size increases, the performance of CAE and CLmser-n should be closer, which can be seen from Table 4 by comparing the performance of CLmser-n and CAE when the sample size increases from 10 to 400. To further investigate the influence of DCW on reconstruction capability of models, we conduct more experiments on a known challenging dataset Places2 [28] and find that CLmser-n can still outperform CAE slightly. A possible reason is that for the complete image reconstruction task, where the output pattern is the same as the input pattern, DCW can show its advantage as it enforces an approximate identity mapping layer by layer, which may help the network to approximate the identity mapping. More details can be found in “Appendix B”.

5.2 Convergence Speed

In this section, we analyze the contribution of DCW and DPN to the convergence speed of the network on image reconstruction. We visualize the mean square error between input and reconstruction during the training procedure on STL10-100 in Fig. 6. The loss curves of models trained with stochastic gradient descent (SGD) [3] and Adam method [13] are shown

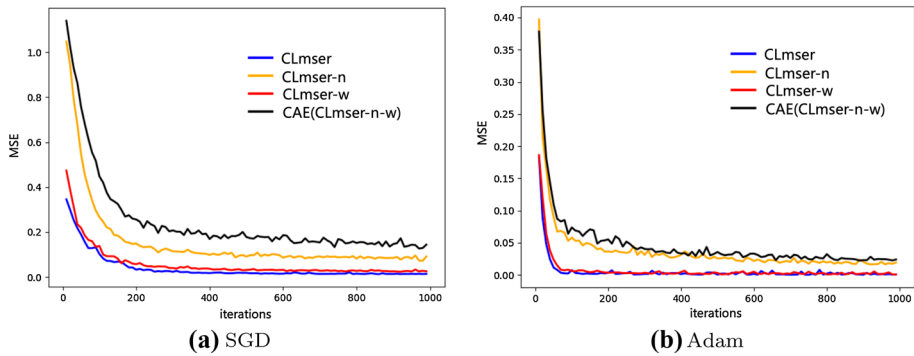


Fig. 6 Loss function visualizations for different models trained with SGD (a) and Adam (b) on STL10-100 dataset

in Fig. 6a, b respectively. We can observe from Fig. 6 that both DPN and DCW can accelerate the convergence speed of the reconstruction model, but DPN plays the main role. This can be explained by Eqs. (13) and (15) that DCW and DPN can add more terms to the gradients so that they can alleviate the gradient vanishing problem and accelerate the convergence speed of the network.

5.3 Image Inpainting

In this section, we applied CLmser on image inpainting [2], the task of filling in corrupted parts in an image with plausible imagination. Some works [16] have studied different types of masks or holes, such as rectangular-shaped masks and irregular masks. In this paper, we mainly focus on images with slight damage, i.e., the corrupted image still holds its main structural information and its identification features. For example, one can still identify the person in the damaged image. We consider the type of irregularly and densely distributed point-shaped mask as illustrated in the left-most column of Fig. 7. The comparisons are conducted on two high quality image datasets including CelebA-HQ [12] and Places2 [28], and we resize the images to 256×256 . We compare our models with some state-of-the-art works as follows,

- PConv [16]: Partial Convolution (PConv), a generative model that replaces the convolutional layer by the partial convolutional layer for filling irregular holes.
- PEN-Net [26]: Pyramid-context ENcoder Network (PEN-Net), a generative model built upon a U-Net structure with a pyramid-context encoder and a multi-scale decoder.
- PICNet [27]: Pluralistic image completion, which introduces a new image completion network with two parallel but linked training pipelines, one is VAE-based reconstructive path and the other is a generative path.

Besides L1 loss, MSE, PSNR, and SSIM, we use Fréchet Inception Distance (FID) [8] as an evaluation metric. As shown in [8], FID can measure the discrepancy between the ground-truth distribution and the generated distribution.

The comparisons of different models on CelebA-HQ and Places2 are summarized in Table 5. The results demonstrate that CLmser-w, which removes DCW from CLmser, outperforms other models on most metrics. More exactly, the DPN can work well for inpainting images with small missing areas. Meanwhile, from the comparison of CLmser and CLmser-w, we can see that when DCW and DPN are used together, DCW plays a slightly negative role

Table 5 The performance of image inpainting on CelebA-HQ (top) and Places2 (bottom) with irregularly and densely distributed point-shaped masks (CL: CLmser; CL-w: CLmser-w; CL-n: CLmser-n)

	Model	$\ell_1(10^{-2})$	MSE(10^{-3})	PSNR	SSIM	FID
CelebA-HQ	CL	0.6657	0.6186	32.13	0.9460	2.686
	CL-w	0.6030	0.5405	32.73	0.9522	2.321
	CL-n	0.6847	0.6659	31.81	0.9465	2.695
	CAE	0.6917	0.6700	31.79	0.9456	2.744
	PConv [16]	0.6632	0.7015	31.60	0.9457	2.953
	PEN-Net [26]	0.9528	1.3100	28.86	0.9126	9.679
	PICNet [27]	0.6700	0.6617	31.84	0.9446	2.216
Places2	CL	1.010	1.384	28.65	0.9279	3.142
	CL-w	0.932	1.281	28.99	0.9341	2.636
	CL-n	1.092	1.608	28.00	0.9214	4.110
	CAE	1.076	1.620	27.96	0.9222	4.085
	PConv [16]	1.188	1.980	27.10	0.9084	5.901
	PEN-Net [26]	1.370	2.557	25.98	0.8930	8.847
	PICNet [27]	1.008	1.426	28.52	0.9292	2.870

Bold indicates that it achieves the best performance for the corresponding metric. For l1 and MSE, the smaller the better; for SSIM and PSNR, the larger the better



Fig. 7 Qualitative comparisons of image inpainting on CelebA-HQ: (columns left to right) masked image, CLmser, CLmser-w, CLmser-n, CAE, PConv, PEN-Net, PICNet, ground truth

Table 6 Per-frame inference time of image inpainting on CelebA-HQ test set (CL: CLmser; CL-w: CLmser-w; CL-n: CLmser-n)

Model	CL	CL-w	CL-n	CAE	PConv [16]	PEN-Net [26]	PICNet [27]
Time(10^{-3})	2.593	2.630	2.595	2.602	2.198	32.18	5.074 – 75.69

Bold indicates that PConv cost the minimum per-frame inference time

in image inpainting. A possible reason is that the goal of the network is not to approximate the identity mapping when the input pattern and output pattern are different, so the design of symmetric weights will affect the inpainting performance of the model. Examples from CelebA-HQ are visualized in Fig. 7, and more qualitative examples on Places2 can be found in Appendix C. Although PEN-Net performs worse than CLmsr-w under irregularly and densely distributed point-shaped mask, PEN-Net can generate realistic samples for damaged images with large missing areas [26].

We also analyze the per-frame inference time of different models. We calculate the average inference time of different models that predict 3000 images on the test set of CelebA-HQ. We repeat the experiments for 5 times to reduce the random errors. In our experiments, all models run on a machine with 2080ti and the size of the input image is 256×256 . Table 6 reports the per-frame inference time of each model.

We can see from Table 6 that the PConv has minimal per-frame inference time, which demonstrates that PConv can work effectively in real-time. CLmsers (CL, CL-w, CI-n, CAE) can also run in real-time due to the similar inference time to PConv. We can also see that PEN-Net takes much more time than CLmsers and PConv. The pyramid-context encoder in PEN-Net improves the quality of encoding by filling regions from high-level feature maps to low-level feature maps through the Attention Transfer Network (ATN) [26], which may cost some extra time. The per-frame inference time of PIC-Net varies with its sampling numbers: 5.074×10^{-3} for 1 sample and 75.69×10^{-3} for 20 samples (PICNet can generate multiple samples for a masked image and choose the best one as the final prediction).

5.4 Summary of DPN and DCW

In Sect. 5.1, we can see that DPN can significantly enhance the performance of image reconstruction by comparing the performance of CLmsr-w and CAE. Moreover, from the results of the CLmsr-n and CAE, we can see that DCW can also slightly improve the generalization ability of the reconstruction model, but it is less effective than DPN. We can get similar observations in Sect. 5.2 that both DPN and DCW can contribute to faster convergence in training and DPN plays the main role. In Sect. 5.4, we can see that when using DPN alone, the model can get the best performance in image inpainting.

In summary, when used alone, both DCW and DPN can enhance the performance of image reconstruction on small-scale datasets and accelerate the convergence speed of the model, but DCW is always less effective than DPN. What's more, DCW plays a slightly negative role in the experiment of image inpainting while DPN plays a positive role. All the experimental results are consistent with the corresponding discussions in Sect. 4.2.

6 Conclusion

We have implemented a deep multilayer version of Lmsr net for the first time and further developed it into CNN based Lmsr. We conducted extensive experiments on benchmark datasets to show that not only the early proposed Lmsr indeed works, but also the two major built-in natures, i.e., DCW and DPN, show their strengths in different aspects. For image reconstruction, both DPN and DCW can ease gradient vanishing problem and make the CLmsr network perform better on small-scale datasets and converge faster in training. However, DPN always plays the main role. For image inpainting, while DCW affects the inpainting performance, DPN can enhance the inpainting performance and enables CLmsr-

Table 7 Image classification on MNIST, Fashion-MNIST and Cifar10

Dataset	MNIST		Fashion-MNIST		Cifar10	
Model	CLmser	CNN	CLmser	CNN	CLmser	CNN
Accuracy	0.9923	0.9906	0.9037	0.9114	0.6518	0.6717
MSE	0.0021	–	0.0054	–	0.0114	–

w to outperform some recent state-of-the-art methods in inpainting with irregularly and densely distributed point-shaped masks.

In this paper, CLmser is applied to image reconstruction and image inpainting. It will be interesting to extend CLmser in other areas, such as image classification and image generation. When Lmser [23] is firstly proposed, it also has the supervised version, which is referred to another duality in Lmser, i.e., Duality in the Supervision Paradigm (DSP) [25]. This duality makes Lmser able to handle image classification tasks. It will be a further research topic. We conduct some simple experiments on MNIST [15], Fashion-MNIST [22] and Cifar10 [14] to show the potential of CLmser on image classification. We implement the DSP by feeding both the image and its label into the network and adding a classification error to the reconstruction error. The performance of classification and reconstruction is reported in Table 7, from which we can observe that DSP enables CLmser to complete the task of image classification and image reconstruction simultaneously.

As for image generation, DPN can pass the details from the encoder to the decoder which may disturb the top-down generated features. How to fuse the top-down information and the detailed information from the encoder also deserves further research.

Acknowledgements This work was supported by National Science and Technology Innovation 2030 Major Project (2018AAA0100700) of the Ministry of Science and Technology of China, and SJTU Medical Engineering Cross-cutting Research Foundation (ZH2018ZDA07), as well as ZhiYuan Chair Professorship Start-up Grant (WF220103010) from Shanghai Jiao Tong University.

A Implementations

For PConv [16], we use an implementation in PyTorch.¹ For PEN-Net [26], we use their official implementation² and follow all their original settings. For PICNet [27], we use their official implementation³ and follow all the settings in their original work except that we start training from the provided pre-trained models and training for 1,000,000 iterations for CelebA-HQ and 2,000,000 iterations for Places2.

B Additional Results on Reconsturction Capability

To further investigate the reconstruction capability of CLmser-n and AE, we conduct experiments on Places2 dataset. We train the two models on 180,346 samples randomly chosen

¹ <https://github.com/naoto0804/pytorch-inpainting-with-partial-conv>.

² <https://github.com/researchmm/PEN-Net-for-Inpainting>.

³ <https://github.com/lyndonzheng/Pluralistic-Inpainting>.

Table 8 Performance of image reconstruction on Places2

Model	ℓ_1	MSE	PSNR	SSIM
CAE	0.0431	0.00471	23.29	0.6842
CLmscr-n	0.0417	0.00453	23.46	0.6942

Bold indicates that it achieves the best performance for the corresponding metric. For ℓ_1 and MSE, the smaller the better; for SSIM and PSNR, the larger the better

from the train set of Places2 and test them on 10,000 images which are randomly chosen from the test set of Places2. The results are listed in Table 8.

The results show that the design of the symmetric weights (DCW) does not affect the reconstruction performance.

C Qualitative Examples

See Fig. 8.

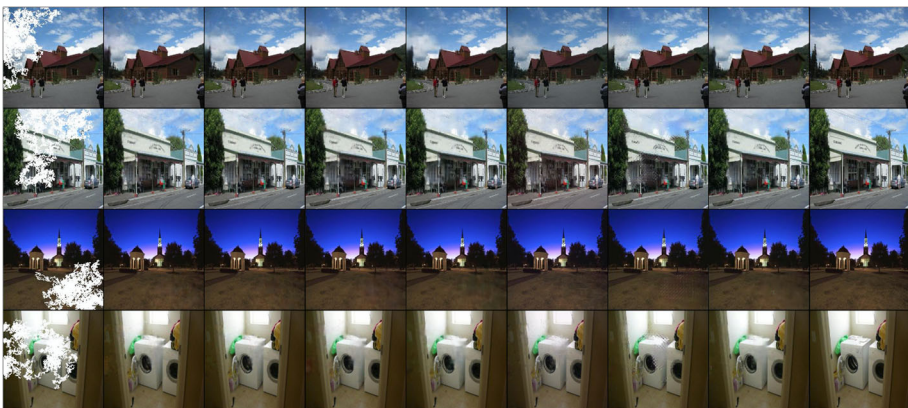


Fig. 8 Qualitative comparisons of image inpainting on Places2: (columns left to right) masked image, CLmscr, CLmscr-w, CLmscr-n, CAE, PConv, PEN-Net, PICNet, ground truth

References

1. Ballard DH (1987) Modular learning in neural networks. In: Proceedings of the sixth national conference on artificial intelligence—volume 1, AAAI'87, pp 279–284
2. Bertalmio M, Sapiro G, Caselles V, Ballester C (2000) Image inpainting. In: Proceedings of the 27th annual conference on computer graphics and interactive techniques, SIGGRAPH '00. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, pp 417–424
3. Bottou L (2010) Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT'2010. Springer, pp 177–186
4. Bourlard H, Kamp Y (1988) Auto-association by multilayer perceptrons and singular value decomposition. *Biol Cybern* 59(4–5):291–294

5. Coates A, Ng A, Lee H (2011) An analysis of single-layer networks in unsupervised feature learning. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics, pp 215–223
6. Han K, Wen H, Zhang Y, Fu D, Culurciello E, Liu Z (2018) Deep predictive coding network with local recurrent processing for object recognition. In: Advances in neural information processing systems, pp 9201–9213
7. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
8. Heusel M, Ramsauer H, Unterthiner T, Nessler B, Hochreiter S (2017) Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: Advances in neural information processing systems, pp 6626–6637
9. Hinton GE, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. *Science* 313(5786):504–507
10. Huang G, Liu Z, Weinberger KQ (2016) Densely connected convolutional networks. *CoRR arXiv:1608.06993*
11. Isola P, Zhu JY, Zhou T, Efros AA (2017) Image-to-image translation with conditional adversarial networks. In: CVPR
12. Karras T, Aila T, Laine S, Lehtinen J (2018) Progressive growing of GANs for improved quality, stability, and variation. In: International conference on learning representations. <https://openreview.net/forum?id=Hk99zCeAb>
13. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. *CoRR arXiv:1412.6980*
14. Krizhevsky A, Hinton G et al (2009) Learning multiple layers of features from tiny images
15. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. *Proc IEEE* 86(11):2278–2324
16. Liu G, Reda FA, Shih KJ, Wang T, Tao A, Catanzaro B (2018) Image inpainting for irregular holes using partial convolutions. *CoRR arXiv:1804.07723*
17. Liu Z, Luo P, Wang X, Tang X (2015) Deep learning face attributes in the wild. In: Proceedings of the IEEE international conference on computer vision, pp. 3730–3738
18. Mao X, Shen C, Yang YB (2016) Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In: Advances in neural information processing systems, pp 2802–2810
19. Ronneberger O, Fischer P, Brox T (2015) U-net: convolutional networks for biomedical image segmentation. In: International conference on medical image computing and computer-assisted intervention. Springer, pp 234–241
20. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. *Nature* 323(6088):533–536
21. Wang Z, Bovik AC, Sheikh HR, Simoncelli EP et al (2004) Image quality assessment: from error visibility to structural similarity. *IEEE Trans Image Process* 13(4):600–612
22. Xiao H, Rasul K, Vollgraf R (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*
23. Xu L (1991) Least MSE reconstruction for self-organization: (i)&(ii). In: Proceedings of 1991 international joint conference on neural networks, pp 2363–2373
24. Xu L (1993) Least mean square error reconstruction principle for self-organizing neural-nets. *Neural Netw* 6(5):627–648
25. Xu L (2019) An overview and perspectives on bidirectional intelligence: Lmser duality, double IA harmony, and causal computation. *IEEE/CAA J Autom Sin* 6(4):865–893
26. Zeng Y, Fu J, Chao H, Guo B (2019) Learning pyramid-context encoder network for high-quality image inpainting. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1486–1494
27. Zheng C, Cham TJ, Cai J (2019) Pluralistic image completion. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1438–1447
28. Zhou B, Lapedriza A, Khosla A, Oliva A, Torralba A (2017) Places: a 10 million image database for scene recognition. *IEEE Trans Pattern Anal Mach Intell* 40(6):1452–1464