Computer Vision and Image Understanding 200 (2020) 103029

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Two-stream deep sparse network for accurate and efficient image restoration

Shuhui Wang [a], Ling Hu [a,b], Liang Li [a,*], Weigang Zhang [c], Qingming Huang [a,b,d]

[a] Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
[b] School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100049, China
[c] Harbin Institute of Technology, Weihai, 264200, China
[d] Key Laboratory of Big Data Mining and Knowledge Management, University of Chinese Academy of Sciences, Beijing 101408, China

ARTICLE INFO

Keywords:
Two-stream sparse network
Image restoration
Image super-resolution
Image denoising

ABSTRACT

Deep convolutional neural network (CNN) has achieved great success in image restoration. However, previous methods ignore the complementarity between low-level and high-level features, thereby leading to limited image reconstruction quality. In this paper, we propose a two-stream sparse network (TSSN) to explicitly learn shallow and deep features to enforce their respective contribution to image restoration. The shallow stream learns shallow features (*e.g.*, texture edges), and the deep stream learns deep features (*e.g.*, salient semantics). In each stream, sparse residual block (SRB) is proposed to efficiently aggregate hierarchical features by constructing sparse connections among layers in the local block. Spatial-wise and channel-wise attention are used to fuse the shallow and deep stream which recalibrates features by weight assignment in both spatial and channel dimensions. A novel loss function called Softmax-L_1 loss is proposed to increase penalties of pixels that have large L_1 loss (*i.e.*, hard pixels). TSSN is evaluated with three representative IR applications, *i.e.*, single image super-resolution, image denoising and JPEG deblocking. Extensive experiments demonstrate that TSSN outperforms most of state-of-the-art methods on benchmark datasets on both quantitative metric and visual quality.

1. Introduction

Image restoration (IR) is an important technique in computer vision, which is widely applied in many areas such as medical imaging (Huang et al., 2017b), satellite imaging (Xiao et al., 2018) and security monitoring (Yoshida et al., 2012), *etc*. IR aims to generate a high-quality (HQ) image from its degraded low-quality (LQ) image, which is generally an ill-posed problem since degradation is an irreversible process. To solve this problem, many methods have been proposed for three classic image restoration tasks, namely, single image super-resolution (SISR) (Yang et al., 2010), image denoising (Dabov et al., 2007b), and JPEG deblocking (Dong et al., 2015). Among these tasks, SISR aims to reconstruct high-resolution (HR) image from low-resolution (LR) image, image denoising is to remove noise from noisy image (*i.e.*, generally additive white Gaussian noise with a standard deviation σ), and JPEG deblocking is to remove the blocking artifact caused by the compression process of JPEG.

Due to the remarkable learning ability, deep convolutional neural network (CNN) is widely used in IR tasks (Dong et al., 2014, 2015). It has been proved by previous study that increasing network depth is helpful in improving the reconstruction quality of HQ images. However, simply stacking more layers leads to increase of computation complexity, requirement of excellent training skills and ignorance of complementarity between different feature levels.

Mainstream IR methods are commonly single-stream feed-forward CNN architecture where only features of the final convolutional layer are directly used in the reconstruction stage. Different receptive fields are considered to learn more complex image-to-image mappings using multi-layer features (Seif and Androutsos, 2018; Li et al., 2018; Zhang et al., 2017b). However, in these works, multi-layer features are only learned in local blocks without considering the global structure information for accurate reconstruction. As demonstrated in previous study (Zeiler and Fergus, 2014; Zeiler et al., 2011; Wu et al., 2017), in deep neural networks, the bottom layers capture more detailed features such as lines and corners, while the top layers learn more semantic features such as shapes, frames and boundaries. In IR problem, direct reconstruction of HQ image from original LQ image may easily suffer from entanglement among different texture information and over-smoothness in visual appearance. Therefore, compared to single stream IR pipeline, it would be better to use two-stream structure to separately learn the semantic features and detailed features to emphasize each part respectively.

Moreover, the hierarchical features generated by very deep network have been widely used by techniques such as skip connections or dense

* Corresponding author.
E-mail addresses: wangshuhui@ict.ac.cn (S. Wang), liang.li@ict.ac.cn (L. Li), wgzhang@hit.edu.cn (W. Zhang), qmhuang@ucas.ac.cn (Q. Huang).

https://doi.org/10.1016/j.cviu.2020.103029
Received 31 December 2019; Received in revised form 18 June 2020; Accepted 23 June 2020
Available online 29 June 2020

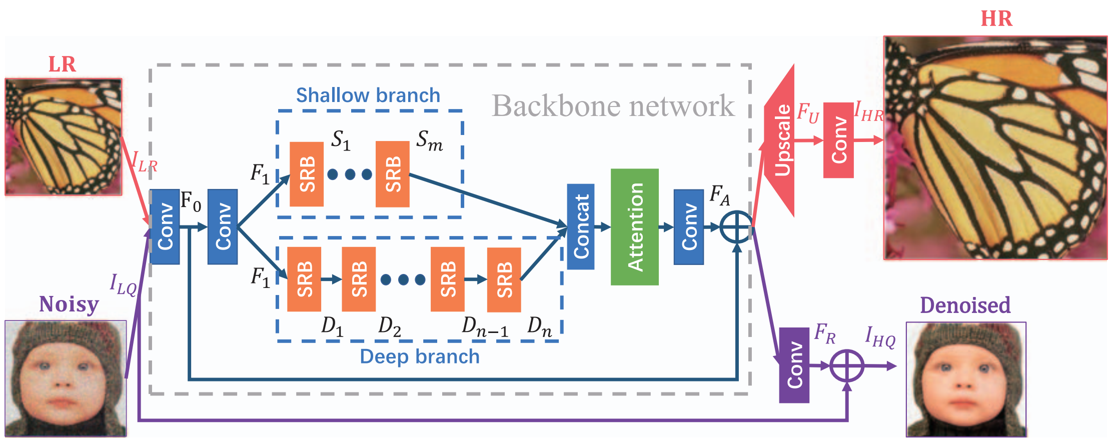


Fig. 1. Overall architecture of TSSN. The backbone network is shown in gray box, in which the top branch learns shallow features with m SRBs and the bottom branch learns deep features with n SRBs ($m < n$). The red flow is designed for image super-resolution and purple flow for image denoising and JPEG deblocking tasks. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

connections. Tai et al. (2017b) proposed a memory block to explicitly discover persistent memory through different feature levels. Tong et al. (2017) proposed dense block to combine low-level features and high-level features to boost the reconstruction performance. Zhang et al. (2018b) proposed residual dense block which cooperates dense connection with local residual learning to extract local features. However, existing techniques usually suffer from potential risk of over-fitting, parameter redundancy and large memory consumption. In this paper, we adopt sparse connection with consistent image restoration quality but large improvements on speed and computational consumption.

To address the above mentioned issues, we propose Two-Stream Sparse Network (TSSN), a new network architecture that outperforms most of state-of-the-art methods on both quality of reconstructed images and computational efficiency. Specifically, the two-stream structure consists of a shallow branch and a deep branch. The shallow branch learns features describing rich textures, and the deep branch learns features with more semantic saliency. Moreover, motivated by Zhu et al. (2018), Sparse Residual Block (SRB), is proposed to efficiently extract rich local features in each branch. The sparse connection in SRB sparsely aggregates features from part of previous layers. To better facilitate network training, residual connections are utilized locally across the sparse connections within each SRB, and globally across multiple SRBs. We also employ spatial-wise and channel-wise attention (Hu et al., 2018b,a) to fuse the shallow and deep features separately learned by the two-stream structure. Attention mechanism recalibrates features by weight assignment, thus it enables the network to learn more useful information for HQ image reconstruction. Furthermore, we propose a new Softmax-L_1 loss which assigns L_1 loss with adaptive coefficients to emphasize hard pixels, which facilitates TSSN to possess better convergence behavior and reconstruction quality.

Our method has the following advantages. First, by explicitly combining the two complementary streams, the reconstruction of HQ image tends to be more stable and robust to information entanglement. Second, with fewer connections, the SRB is more efficient since it can stack more layers under the same computation budget as dense connection. By imposing attention on the two-stream network, our method shows distinguished image reconstruction accuracy by elegant feature recalibration, especially on images with regular texture, such as the building images with large amount of grids and stripes.

This work is a comprehensive extension of our previous work (Hu et al., 2019). Instead of only using channel attention to aggregate features of different branches in Hu et al. (2019), we combine spatial attention and channel attention to fuse the learned shallow features and deep features, and better performance is achieved. We design a new loss, *i.e.*, the softmax-L_1 loss for image restoration tasks. While in Hu et al. (2019), we only use L_2 loss. We compare our method with more state-of-the-art algorithms of SISR and perform comparison up to x8 setting. We add more detail analyses to our model, such as more comprehensive ablation study, number of network parameter and running time. We apply our network to other image restoration tasks, *i.e.*, image denoising and JPEG deblocking. Extensive experiments, including quantitative and qualitative parts, demonstrate that TSSN outperforms existing approaches in these IR tasks. The source code and pre-trained models of our method can be downloaded at https://github.com/LaineyHu/TSSN.

2. Related work

Traditional IR methods generally lack the ability to produce HQ images with high-frequency and realistic textures. To address this limitation, deep convolutional neural networks are widely used in the state-of-the-art methods in IR task. So we mainly focus on recent deep learning based methods that are closely related to our study and techniques applied in our work.

Deep-learning-based SISR. In recent years, numerous deep learning based SISR methods have been proposed and greatly outperform traditional SISR methods. Dong et al. (2014) first proposed to learn CNN-based mappings from LR image to HR image in an end-to-end manner.

Making network deeper and wider. In order to enhance the performance, researchers seek to build deeper and wider networks (Kim et al., 2016a,b; Tai et al., 2017a). Kim et al. (2016a) proposed a 20-layer CNN model and applied residual connection to make the very deep network easier to train. Meanwhile, Kim et al. (2016b) introduced recursive layer to control the number of model parameters while increasing the depth up to 16 recursions. Tai et al. (2017a) proposed an even deeper CNN model (up to 52 convolutional layers) where recursive learning and residual learning are adopted. Nevertheless, all of these methods tend to construct deeper and more complex network structures by consuming more resources.

Making full use of hierarchical features. To fully exploit different levels of features, Lai et al. (2017) proposed LapSRN, which progressively reconstructs the sub-band residuals of HR images in a pyramid manner. Tong et al. (2017) introduced dense skip connections in SRResNet, where feature maps of each layer are propagated into all subsequent layers, providing an effective way to combine the low-level features and high-level features. Based on SRResNet, Lim et al. (2017) analyzed and removed unnecessary modules to simplify the network architecture, and win the champion of the NTIRE2017 SR Challenge. Tai et al. (2017b) introduced a memory block, consisting of a recursive unit and a gate unit, which explicitly mines persistent memory through an adaptive learning process. Zhang et al. (2018b) combined residual learning and dense connection in the residual dense network (RDN) to exploit hierarchical features from all the convolutional layers. Li et al. (2018) utilized different sizes of convolution kernels to adaptively detect the image features in different scales. All of these approaches tend to exploit different levels of features locally but do not explicitly learn the useful features globally.

Specifically, skip connections have been considered to alleviate the ill-posedness of IR problems by hierarchical feature combination (Zhang et al., 2018b). The sparsely aggregated network in Zhu et al. (2018) is designed for classification, which demonstrates better efficiency and effectiveness compared to DenseNet (Huang et al., 2017a). The Residual dense network (Zhang et al., 2018b) is a direct extension from classification to image super-resolution task, which is still a single-stream network. In our TSSN, inspired by the sparse connection in Zhu et al. (2018), we propose sparse residual block, which tends to be a better counterpart that the residual dense block (RDB) in Zhang et al. (2018b) in terms of both model size and accuracy. Moreover, we construct the whole TSSN network by organizing the SRBs in a two-stream structure, while the RDBs are organized as a single-stream structure in Zhang et al. (2018b).

In general, we need to enforce that our TSSN method is a two-stream network structure, while the networks of Zhu et al. (2018) and Zhang et al. (2018b) are single-stream structure. The two-stream structure can explicitly preserve both visual details and semantic information, which has not been well studied for image restoration tasks.

Image denoising and JPEG deblocking. The CNN structure was also applied in JPEG deblocking task (Dong et al., 2015), and it was found that networks designed for SISR are also applicable to JPEG deblocking. Mao et al. (2016) built very deep networks up to 30 layers with symmetric skip connections across encoder–decoder network. Zhang et al. (2017a) utilized residual learning and batch normalization to speed up the training process and boost the denoising performance. Zhang et al. (2017b) integrated CNN denoisers as a prior into model-based optimization method. Tai et al. (2017b) proposed MemNet with up to 80 layers, which improves performance at the cost of increased time. Most of the above methods ignore the complementarity among hierarchical features. To address this problem, we propose TSSN to explicitly learn shallow and deep features via a two-stream structure and design SRB to efficiently construct deeper networks.

Loss functions for IR. As the driving force for effective network learning, loss functions for image restoration also attract research attention. Since PSNR (*i.e.*, standard metric for IR) is highly correlated with pixel-wise difference and minimizing pixel loss directly maximizes PSNR, the pixel loss has become the most widely used loss function in this field. Other loss functions (*e.g.*, perceptual loss (Johnson et al., 2016), adversarial loss (Ledig et al., 2017), *etc.*) suffer the problem with lower PSNR and are therefore less used.

Compared with L_1 loss, L_2 loss penalizes larger errors but is more tolerant to small errors. Zhao et al. (2017) investigated the difference between L_1 and L_2 loss, demonstrating that L_1 loss is believed to be more robust against outliers, thus facilitating networks to converge faster and produce better results. In practice, L_1 loss shows improved accuracy and better convergence over L_2 loss (Ahn et al., 2018; Lim et al., 2017; Zhao et al., 2017). However, L_1 loss cannot well handle those hard pixels with larger reconstruction errors, leading to the lack of high-frequency details in the reconstructed images.

Our work. All of the above methods are single stream networks, lacking the ability to explicitly learn different levels of features. In comparison, we propose TSSN to separately learn shallow and deep features via a two-stream structure and then fuse them by attention, which has been rarely investigated inIR tasks. In each stream, SRB is designed to efficiently aggregate local features and construct deeper networks. In addition, Softmax-L_1 loss is adapted from L_1 loss to handle hard pixels and facilitate networks to produce better results.

3. Proposed methods

3.1. Network structure

We demonstrate overall network structure for image restoration in Fig. 1, in which the backbone network is shown in gray box. For image super-resolution task, we design a upscale module for reconstruction stage as shown in the red flow of Fig. 1. For other image restoration tasks such as image denoising and JPEG deblocking, network structure is adapted by removing the upscale module and adding a residual connection as shown in the purple flow of Fig. 1.

TSSN for SISR. The overall structure is shown in the red flow of Fig. 1. First of all, we use two convolutional layers to extract visual features. Then we separately learn shallow and deep features via a two-stream structure, after which attention mechanism is used to fuse those features. Through a global residual connection, we construct a refined LR input to the upscale module. The upscaled output is finally polished by a convolution layer. Details are demonstrated as follows.

We denote LR image as I_{LR}. After two convolutional layers, we extract features F_0 and F_1,

$$F_0 = Conv_1(I_{LR}), F_1 = Conv_2(F_0) \tag{1}$$

where $Conv_i$ denotes convolution operation of layer i. Then F_1 is fed into two different branches: the shallow branch has m SRBs to extract features with more texture details while the deep one has n SRBs that extracts features with more semantic saliency ($m < n$). The specific number (*i.e.*, m and n) of SRBs in each branch is set empirically. In shallow branch,

$$S_i = SRB_S(S_{i-1}),\ i = 1, 2, \dots, m, \tag{2}$$

where $S_0 = F_1$ and SRB_S denotes the SRB extractor in shallow branch. In deep branch,

$$D_i = SRB_D(D_{i-1}),\ i = 1, 2, \dots, n, \tag{3}$$

where $D_0 = F_1$ and SRB_D denotes the SRB extractor in deep branch. We employ GE (Hu et al., 2018a) and SE (Hu et al., 2018b) attention to fuse the shallow and deep features extracted from two branches as

$$F_A = H_{Att}([S_m, D_n]), \tag{4}$$

where $[S_m, D_n]$ denotes concatenation operation and H_{Att} denotes GE and SE attention fusion. We learn the LR-to-HR mapping in residual manner by adding features of the first convolution layer and the attention-fused features to construct more informative feature maps, which are then fed into the upscale module to reconstruct HR results. We use pixel shuffle method proposed by Shi et al. (2016) to reconstruct the result. Upscale module is formulated as

$$F_U = H_{Upscale}(F_0 + F_A), \tag{5}$$

where $H_{Upscale}$ denotes the pixel shuffle operation. Finally we use a convolution operation to polish the final result,

$$I_{HR} = Conv_{end}(F_U), \tag{6}$$

where $Conv_{end}$ denotes the last convolutional layer.

TSSN for image denoising and JPEG deblocking. The overall structure is shown in the purple flow of Fig. 1, which is slightly adapted from the SR network structure which removes the upscale module and adds another global residual connection cross the whole network. Details are demonstrated as follows.

We denote LQ image as I_{LQ}. We extract features F_0,

$$F_0 = Conv_1(I_{LQ}), \tag{7}$$

where we replace the input I_{LR} with I_{LQ} in Eq. (1). After the same operations as from Eq. (1) to Eq. (4) in SR method, we get attention

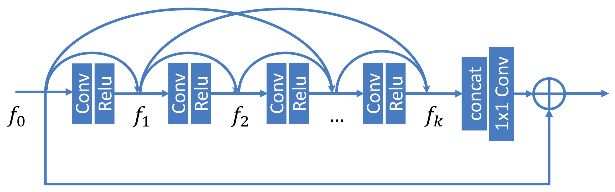


Fig. 2. Our proposed sparse residual block.

fused features F_A. Then we feed the residual results of F_A and F_0 into the last convolutional layer as

$$F_R = Conv_{end}(F_A + F_0). \tag{8}$$

For image denoising and JPEG deblocking, we adds another global residual connection to propagate original information to the reconstruct result as

$$I_{HQ} = I_{LQ} + F_R, \tag{9}$$

where I_{HQ} is the final high-quality reconstructed result.

Details for network structure. In our proposed TSSN, we set 4 SRBs in shallow branch and 7 SRBs in deep branch for the best use of limited computational resources. In one SRB, there are 16 convolutional layers with sparse connections. We set 3×3 as the kernel size of all convolutional layers except for the channel reduction layer which is set as 1×1. For convolutional layers with kernel size of 3×3, we use zero padding to keep the size of output fixed. All the numbers of filters are set to be 64. For SISR, we use pixel shuffle method again to upscale the coarse resolution features to fine ones, which is removed for image denoising and JPEG deblocking. The output of final convolutional layer is set to be 3 channels for color images and 1 channel for gray images.

3.2. Sparse residual block

For the sake of fusing multi-layer information and better network training, current works tend to use dense connection among layers to aggregate hierarchical features, but such operation brings problems with over-fitting and huge computation burden. Instead, sparse connection (Zhu et al., 2018) has comparable performance to dense connection but with fewer parameters. Motivated by this, we propose SRB in which we adopt sparse connection to aggregate local features as shown in Fig. 2. Different from dense connection, sparse connection sparsely aggregates features from part of previous layers that are exponential offsets (*e.g.*, power of 2) away from current layer. The mechanism is formulated as

$$f_i = H_i([f_{i-2^0}, f_{i-2^1}, f_{i-2^2}, \dots]), i = 1, 2, \dots, L, \tag{10}$$

where H_i denotes convolution and ReLU operations of layer i, f_i denotes the output features of layer i, L denotes the number of layers in one SRB (*e.g.*, 16) and $[f_{i-2^0}, f_{i-2^1}, f_{i-2^2}, \dots]$ means concatenation operation in channel dimension.

Obviously, when gathering previous layers with sparse connection instead of dense connection, it efficiently decreases the number of parameters while retaining the strengths of feature propagation and feature reuse. In Fig. 2, we demonstrate the structure of SRB in which we stack several layers with sparse connection and utilize local residual connection to propagate features of preceding SRBs into subsequent SRBs.

By stacking different numbers of SRBs in two-stream structure, we can extract features of different receptive fields since the more convolutional layers with stride > 1 we stack, the larger receptive field it becomes. Therefore, the shallow branch learns shallow features with more detailed textures through a smaller receptive field, while the deep branch learns more semantic deep features through a larger receptive field.

3.3. Attention fusion

To fuse the shallow and deep features extracted from the two-stream structure, we adopt spatial attention (Hu et al., 2018a) and channel attention (Hu et al., 2018b) that have shown good learning ability in image classification task. It can be viewed as a guidance to bias the allocation of available processing resources towards the most informative features. As shown in Fig. 3, spatial attention recalibrates the information among spatial locations and channel attention considers interdependencies among channels.

Spatial attention. We denote the input features as F_I and the output features as F_O. At first, we reduce F_I by 4 times in spatial dimension by average pooling as

$$F_P = AvgPool(F_I). \tag{11}$$

Then we generate the spatial attention by applying sigmoid function to enlarged F_P through interpolation as

$$F_A = Sigmoid(Interpolate(F_P)). \tag{12}$$

Finally, we multiply the attention to the F_I and get the attention reassigned output feature F_O as

$$F_O = F_A \odot F_I, \tag{13}$$

where $\odot$ denotes element-wise multiplication.

Channel attention. We denote the input feature as F_I and the output feature as F_O. At first, the n-channel input feature is squeezed into a $1 \times 1 \times n$ vector through global average pooling as

$$F_P = AvgPool(F_I). \tag{14}$$

Then F_P is fed into two fully-connected (FC) layers to learn the relationships between different channels. The output of the two sequential FC layers is stretched by sigmoid function as the attention,

$$F_A = Sigmoid(FC(Relu(FC(F_P)))). \tag{15}$$

The attention is then multiplied to the original input to excite different channels with different weights as

$$F_O(:, :, c) = F_A(:, :, c) * F_I(c), c = 1, \dots, C. \tag{16}$$

The combination of spatial and channel attention mechanisms recalibrate diverse features by assigning different weights to different regions, which enables the network to identify more informative feature channels. Therefore, more detailed HR reconstruction can be achieved by using attention.

3.4. Loss function

Our proposed Softmax-L_1 loss is a compromise of L_1 loss and L_2 loss, which adds adaptive coefficients to L_1 loss to distribute different weights to different pixels. Softmax-L_1 loss increase penalties to hard pixels, which means more attentions are paid on pixels that are hard to be predicted. With the help of Softmax-L_1 loss, TSSN can generate better reconstruction results.

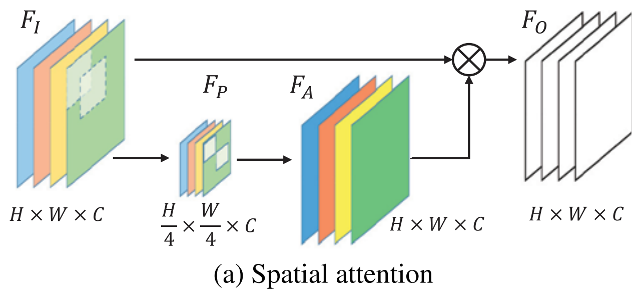
We denote I_{HQ} as generated high-quality image and I_{GT} as ground truth. The L_1 loss is formulated as

$$\mathcal{L}_1 = \left\| I_{HQ} - I_{GT} \right\|_1 . \tag{17}$$

Softmax-L_1 loss is calculated by reweighting the per-pixel loss with the Sigmoid values as

$$\mathcal{L}_s = \sum_{h,w,c} \sigma(|I_{HQ}(h,w,c) - I_{GT}(h,w,c)|) \ \ |I_{HQ}(h,w,c) - I_{GT}(h,w,c)|, \tag{18}$$

where σ denotes the Sigmoid function.



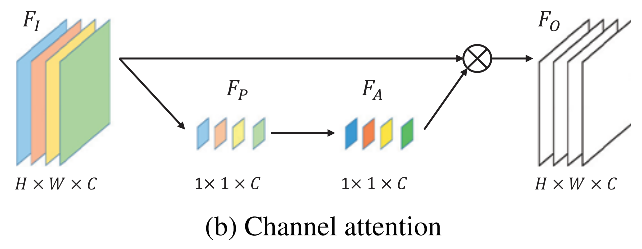


Fig. 3. Attention module for image restoration task: (a) Spatial attention and (b) Channel attention.

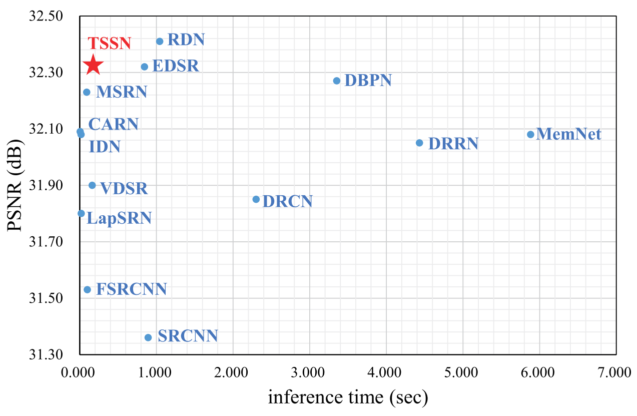


Fig. 4. Trade-off between performance and inference time.

4. Experiments

4.1. Settings

Datasets and metrics. We train our model on the newly released high-quality (2K resolution) dataset DIV2K used by Agustsson and Timofte (2017), which contains 800 training images, 100 validation images and 100 test images. Specifically, We train our model with all the training images and use 10 validation images for validation in the training process. In testing, we use Set5 (Bevilacqua et al., 2012), Set14 (Zeyde et al., 2010), BSD100 (Martin et al., 2001), Urban100 (Huang et al., 2015) and Manga109 (Matsui et al., 2017) as benchmark datasets for SISR. We use Kodak24, BSD68 (Martin et al., 2001), and Urban100 (Huang et al., 2015) for color and gray image denoising. LIVE1 and Classic5 (Foi et al., 2007) are used for JPEG deblocking. For fair comparison, The SISR and JPEG deblocking results are evaluated with PSNR and SSIM on the Y channel (*i.e.*, luminance) of transformed YCbCr space. We ignore the same amount of pixels as scales from the border.

Implementation details. During training, we randomly extract 48 × 48 LQ patches in RGB mode as input and extract corresponding HQ patches as ground truth. We augment the training images by random horizontal flips, vertical flips and 90° rotations. Our model is trained using Adam optimizer and Softmax-L_1 loss. The learning rate is initialized as $1e-4$ and halved at every 200 epochs. We use a mini-batch size of 16. Our proposed models are implemented with PyTorch framework and trained for 1000 epochs with a single NVIDIA GeForce GTX 1080Ti GPU.

4.2. Single image super-resolution

Results on benchmark datasets. We compare our model with state-of-the-art SISR methods including SRCNN (Dong et al., 2014), FSRCNN (Dong et al., 2016), VDSR (Kim et al., 2016a), LapSRN (Lai et al., 2017), MemNet (Tai et al., 2017b), EDSR (Lim et al., 2017), IDN (Hui et al., 2018), RDN (Zhang et al., 2018b), DBPN (Haris et al., 2018), CARN (Ahn et al., 2018) and MSRN (Li et al., 2018). Table 1 presents the quantitative evaluation results on five public benchmark datasets with scale factor x2, x3, x4 and x8. In the table, bold figures denote the best performance and underlined figures denote the second best results. We can find that our model demonstrates significant improvements compared to most of the compared methods. Specifically, our model achieves similar or even better performance over the champion of the NTIRE2017 SR Challenge (EDSR) and the champion of the NTIRE2018 SR Challenge (DBPN), which exploit amounts of superb training skills and utilize more parameters to construct deeper and wider networks with heavy computational budget. TSSN is inferior to RDN in the aspect of performance but superior in the inference speed, which makes a better trade-off between performance and inference time than RDN.

We also present the qualitative results in Fig. 5 in which the red bounding box represents the region we consider for comparison. We can find that our proposed model TSSN clearly reconstructs the detailed textures and edges, while images reconstructed by most of other methods are blurred. Taking *img*_092 in Urban100 as an example, the stripes are accurately reconstructed by our model while results of other methods are blurred or even mistakenly reconstructed. This illustrates that more accurate and visual pleasant super-resolution outputs can be reconstructed by our proposed model.

Trade-off between performance and inference time. To prove the efficiency of our SRB, we show the trade-off between performance and inference time in Fig. 4, in which the average PSNR and the average inference time are tested for scale x2 on BSD100. We can find that our method TSSN achieves state-of-the-art performance compared with those methods which are in the same order of magnitude of inference time. Although RDN achieves better PSNR value, it costs much more time than TSSN, which indicates that our model achieves a better trade-off.

4.3. Image denoising

We compare our model with state-of-the-art Gaussian image denoising methods including BM3D (Dabov et al., 2007b), CBM3D (Dabov et al., 2007a), TNRD (Chen and Pock, 2017), RED (Mao et al., 2016), DnCNN (Zhang et al., 2017a), MemNet (Tai et al., 2017b), IRCNN (Zhang et al., 2017b) and FFDNet (Zhang et al., 2018a). Noisy images are generated by adding additive white Gaussian noise of different levels (*i.e.*, 10, 30, 50 and 70) to clean images. Specifically, image denoising task includes: gray-scale image denoising and color image denoising.

Gray-scale image denoising. Noisy gray images are generated by adding additive white Gaussian noise to clean gray images with different noise levels $\sigma = 10, 30, 50$ and 70. When applying our model to gray-scale image denoising, we set the final reconstruction output as 1 channel. The quantitative results are shown in Table 2. We can find that our model TSSN largely outperforms previous methods on Kodak24 and Urban100 datasets and achieves similar performance with state-of-the-art methods on BSD68 dataset. TSSN achieves great performance in gray-scale image denoising is due to the effective two-stream structure and the efficient SRBs. TSSN compensates deep semantic features with

Table 1
Benchmark results of state-of-the-art SISR methods. Average PSNR/SSIM values for scale factor x2, x3, x4 and x8. Bold numbers indicate the best performance and underlined numbers indicate the second best performance.

Method	Scale	Set5		Set14		BSD100		Urban100		Manga109	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Bicubic	x2	33.66	0.9299	30.24	0.8688	29.56	0.8431	26.88	0.8403	30.80	0.9339
SRCNN	x2	36.66	0.9542	32.45	0.9067	31.36	0.8879	29.50	0.8946	35.60	0.9663
FSRCNN	x2	37.05	0.9560	32.66	0.9090	31.53	0.8920	29.88	0.9020	36.67	0.9710
VDSR	x2	37.53	0.9590	33.05	0.9130	31.90	0.8960	30.77	0.9140	37.22	0.9750
LapSRN	x2	37.52	0.9591	33.08	0.9130	31.08	0.8950	30.41	0.9101	37.27	0.9740
MemNet	x2	37.78	0.9597	33.28	0.9142	32.08	0.8978	31.31	0.9195	37.72	0.9740
EDSR	x2	38.11	0.9602	33.92	0.9195	32.32	0.9013	32.93	0.9351	39.10	0.9773
IDN	x2	37.83	0.9600	33.30	0.9148	32.08	0.8985	31.27	0.9196	38.02	0.9749
RDN	x2	**38.30**	0.9617	**34.14**	**0.9235**	**32.41**	0.9025	**33.17**	**0.9377**	**39.60**	**0.9791**
DBPN	x2	38.09	0.9600	33.85	0.9190	32.27	0.9000	32.55	0.9324	38.89	0.9775
CARN	x2	37.76	0.9590	33.52	0.9166	32.09	0.8978	31.92	0.9256	38.29	0.9754
MSRN	x2	38.08	0.9605	33.74	0.9170	32.23	0.9013	32.22	0.9326	38.82	0.9868
TSSN (ours)	x2	38.22	**0.9629**	33.88	**0.9235**	32.32	**0.9045**	32.91	0.9362	39.12	0.9783
Bicubic	x3	30.39	0.8682	27.55	0.7742	27.21	0.7385	24.46	0.7349	26.95	0.8556
SRCNN	x3	32.75	0.9090	29.30	0.8215	28.41	0.7863	26.24	0.7989	30.48	0.9117
FSRCNN	x3	33.18	0.9140	29.37	0.8240	28.53	0.7910	26.43	0.8080	31.10	0.9210
VDSR	x3	33.67	0.9210	29.78	0.8320	28.83	0.7990	27.14	0.8290	32.01	0.9340
LapSRN	x3	33.82	0.9227	29.87	0.8320	28.82	0.7980	27.07	0.8280	32.21	0.9350
MemNet	x3	34.09	0.9248	30.00	0.8350	28.96	0.8001	27.56	0.8376	32.51	0.9369
EDSR	x3	34.65	0.9280	30.52	0.8462	29.25	0.8093	28.80	0.8653	34.17	0.9476
IDN	x3	34.11	0.9253	29.99	0.8354	28.95	0.8013	27.42	0.8359	32.69	0.9378
RDN	x3	**34.78**	0.9299	**30.63**	0.8477	**29.33**	0.8107	**29.02**	**0.8695**	**34.58**	**0.9502**
DBPN	x3	–	–	–	–	–	–	–	–	–	–
CARN	x3	34.29	0.9255	30.29	0.8407	29.06	0.8034	28.06	0.8493	33.50	0.9430
MSRN	x3	34.38	0.9262	30.34	0.8395	29.08	0.8041	28.08	0.8554	33.44	0.9427
TSSN (ours)	x3	34.71	**0.9320**	30.56	**0.8510**	29.25	**0.8151**	28.81	0.8671	34.15	0.9490
Bicubic	x4	28.42	0.8104	26.00	0.7027	25.96	0.6675	23.14	0.6577	24.89	0.7866
SRCNN	x4	30.48	0.8628	27.50	0.7513	26.90	0.7101	24.52	0.7221	27.58	0.8555
FSRCNN	x4	30.72	0.8660	27.61	0.7550	26.98	0.7150	24.62	0.7280	27.90	0.8610
VDSR	x4	31.35	0.8830	28.02	0.7680	27.29	0.0726	25.18	0.7540	28.83	0.8870
LapSRN	x4	31.54	0.8850	28.19	0.7720	27.32	0.7270	25.21	0.7560	29.09	0.8900
MemNet	x4	31.74	0.8893	28.26	0.7723	27.40	0.7281	25.50	0.7630	29.42	0.8942
EDSR	x4	32.46	0.8968	28.80	0.7876	27.71	0.7420	26.64	0.8033	31.02	0.9148
IDN	x4	31.82	0.8903	28.25	0.7730	27.41	0.7297	25.41	0.7632	29.40	0.8936
RDN	x4	**32.61**	0.8999	**28.93**	0.7894	**27.80**	0.7436	**26.85**	**0.8089**	**31.45**	**0.9187**
DBPN	x4	32.47	0.8980	28.82	0.7860	27.72	0.7400	26.38	0.7946	30.91	0.9137
CARN	x4	32.13	0.8937	28.60	0.7806	27.58	0.7349	26.07	0.7837	30.50	0.9078
MSRN	x4	32.07	0.8903	28.60	0.7751	27.52	0.7273	26.04	0.7896	30.17	0.9034
TSSN (ours)	x4	32.53	**0.9018**	28.80	**0.7923**	27.71	**0.7487**	26.59	0.8039	31.02	0.9157
Bicubic	x8	24.40	0.6580	23.10	0.5660	23.67	0.5480	20.74	0.5160	21.47	0.6500
SRCNN	x8	25.33	0.6900	23.76	0.5910	24.13	0.5660	21.29	0.5440	22.46	0.6950
FSRCNN	x8	20.13	0.5520	19.75	0.4820	24.21	0.5680	21.32	0.5380	22.39	0.6730
VDSR	x8	25.93	0.7240	24.26	0.6140	24.49	0.5830	21.70	0.5710	23.16	0.7250
LapSRN	x8	26.15	0.7380	24.35	0.6200	24.54	0.5860	21.81	0.5810	23.39	0.7350
MemNet	x8	26.16	0.7414	24.38	0.6199	24.58	0.5842	21.89	0.5825	23.56	0.7387
EDSR	x8	26.96	0.7762	24.91	0.6420	24.81	0.5985	22.51	0.6221	24.69	0.7841
IDN	x8	–	–	–	–	–	–	–	–	–	–
RDN	x8	**27.23**	0.7854	**25.25**	0.6505	**24.91**	0.6032	**22.83**	**0.6374**	**25.14**	**0.7994**
DBPN	x8	27.21	0.7840	25.13	0.6480	24.88	0.6010	22.73	0.6312	**25.14**	0.7987
CARN	x8	–	–	–	–	–	–	–	–	–	–
MSRN	x8	26.59	0.7254	24.88	0.5961	24.70	0.5410	22.37	0.5977	24.28	0.7517
TSSN (ours)	x8	27.22	**0.7855**	25.04	**0.6522**	24.87	**0.6115**	22.70	0.6307	24.88	0.7882

Table 2
PSNR results on gray-scale image denoising. Bold and underlined numbers denote the best and the second best results.

Method	Kodak24				BSD100				Urban100			
	10	30	50	70	10	30	50	70	10	30	50	70
BM3D	34.39	29.13	26.99	25.73	33.31	27.76	25.62	24.44	34.47	28.75	25.94	24.27
TNRD	34.41	28.87	27.20	24.95	33.41	27.66	25.97	23.83	33.78	27.49	25.59	22.67
RED	35.02	29.77	27.66	26.39	**33.99**	**28.50**	**26.37**	**25.10**	34.91	29.18	26.51	24.82
DnCNN	34.90	29.62	27.51	26.08	33.88	28.36	26.23	24.90	34.73	28.88	26.28	24.36
MemNet	–	29.72	27.68	26.42	–	28.43	26.35	25.09	–	29.10	26.65	25.01
IRCNN	34.76	29.53	27.45	–	33.74	28.26	26.15	–	34.60	28.85	26.24	–
FFDNet	34.81	29.70	27.63	26.34	33.76	28.39	26.30	25.04	34.45	29.03	26.52	24.86
TSSN	**35.13**	**29.91**	**27.80**	**26.48**	33.97	28.49	26.36	25.06	**35.37**	**29.87**	**27.26**	**25.48**

shallow texture features, which utilizes multi-scale scope of context information, leading to better performance.

Qualitative results with noise level ($\sigma = 50$) are shown in Fig. 6 in which the red box marks the region for comparison. Taking *img*_100 in

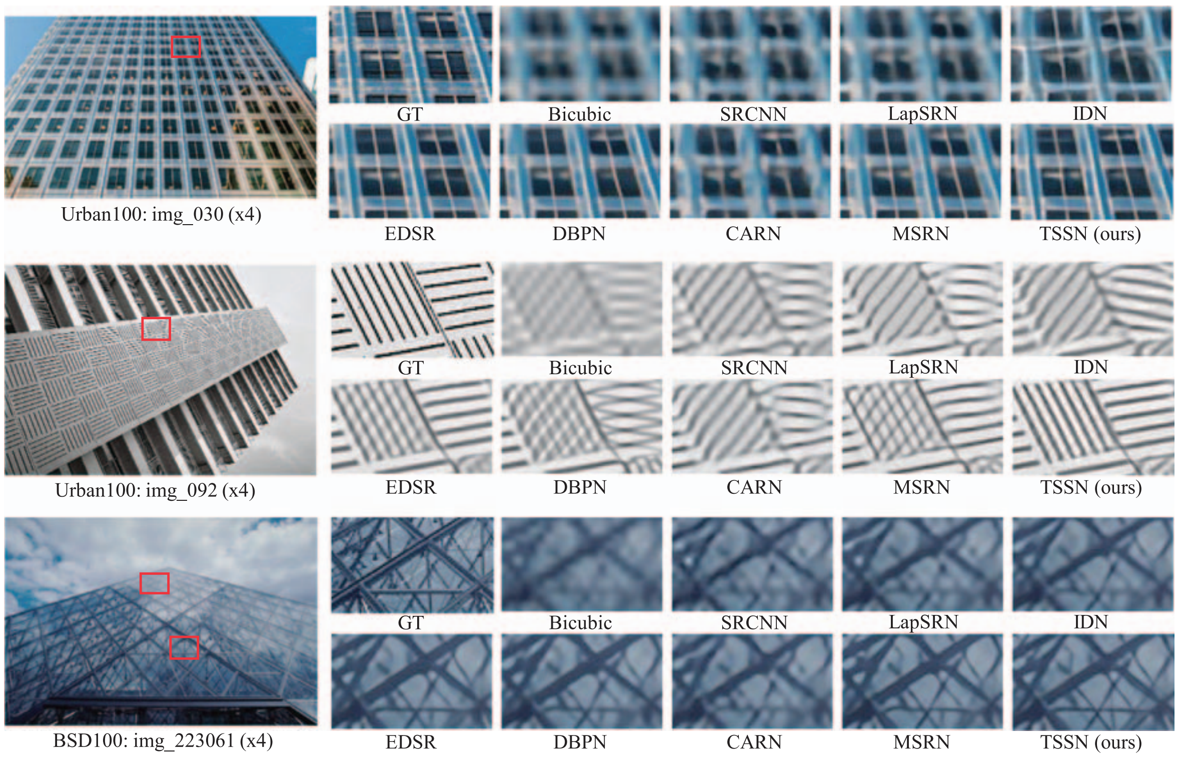


Fig. 5. Qualitative comparison between our model and other state-of-the-art methods on scale x4 super-resolution.

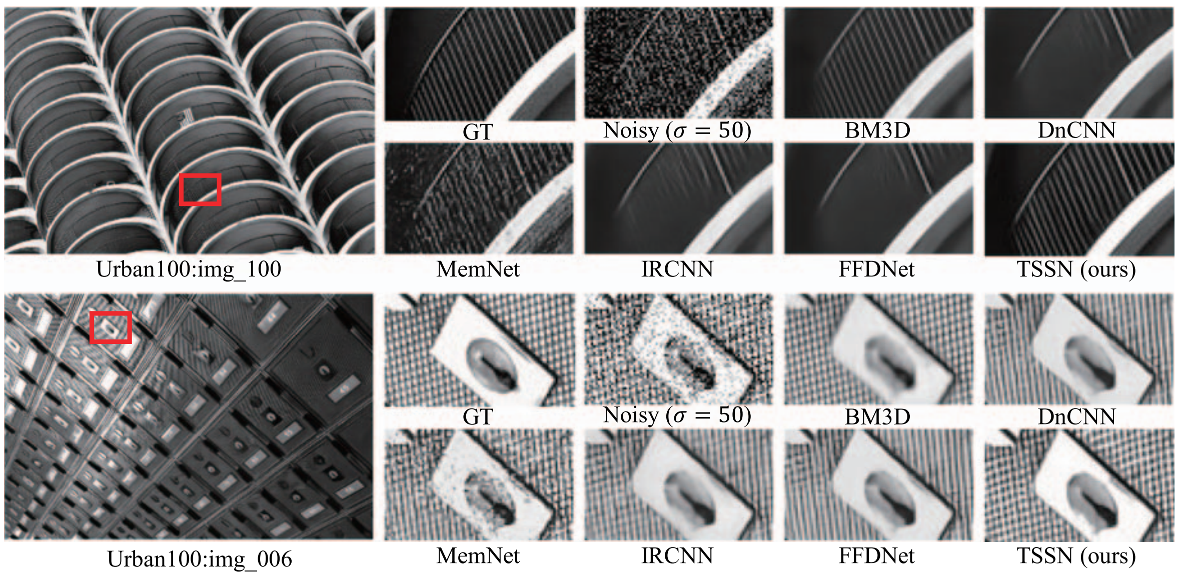


Fig. 6. Qualitative comparison between our model and other state-of-the-art methods on gray-scale image denoising.

Table 3
PSNR results on color image denoising. Bold and underlined numbers denote the best and the second best results.

Method	Kodak24				BSD100				Urban100			
	10	30	50	70	10	30	50	70	10	30	50	70
CBM3D	36.57	30.89	28.63	27.27	35.91	29.73	27.38	26.00	36.00	30.36	27.94	26.31
TNRD	34.33	28.83	27.17	24.94	33.36	27.64	25.96	23.83	33.60	27.40	25.52	22.63
RED	34.91	29.71	27.62	26.36	33.89	28.46	26.35	25.09	34.59	29.02	26.40	24.74
DnCNN	36.98	31.39	29.16	27.64	36.31	30.40	28.01	26.56	36.21	30.28	28.16	26.17
MemNet	–	29.67	27.65	26.40	–	28.39	26.33	25.08	–	28.93	26.53	24.93
IRCNN	36.70	31.24	28.93	–	36.06	30.22	27.86	–	35.81	30.28	27.69	–
FFDNet	36.81	31.39	29.10	27.68	36.14	30.31	27.96	26.53	35.77	30.53	28.05	26.39
TSSN	**37.29**	**31.89**	**29.58**	**28.14**	**36.43**	**30.61**	**28.25**	**26.80**	**36.65**	**31.58**	**29.17**	**27.46**

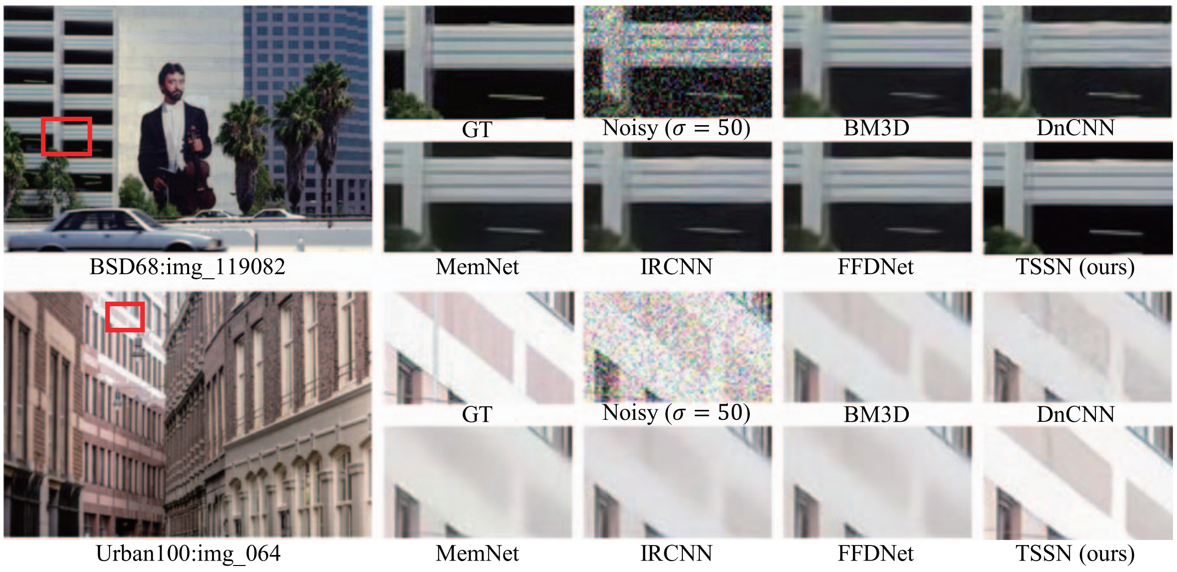


Fig. 7. Qualitative comparison between our model and other state-of-the-art methods on color image denoising. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

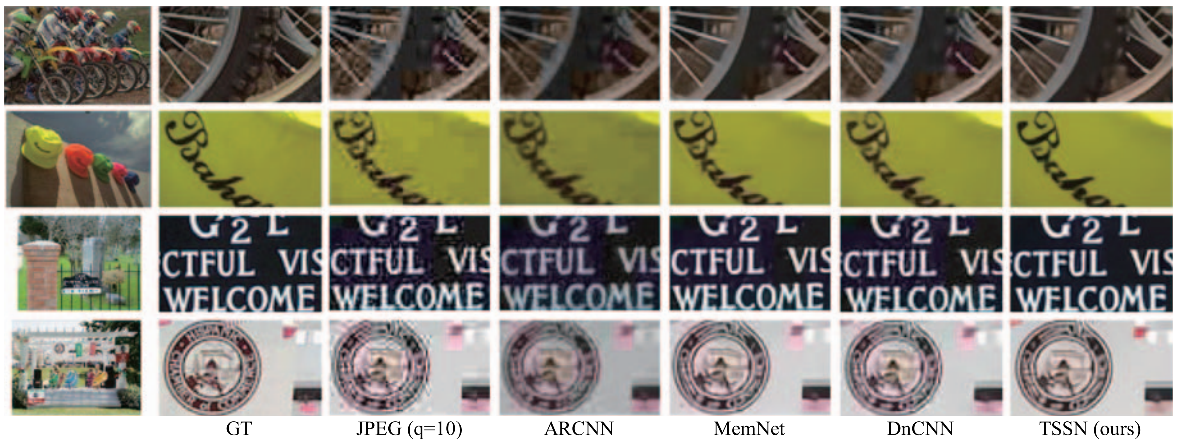


Fig. 8. Qualitative comparison between our model and other state-of-the-art methods on JPEG deblocking.

Table 4
Average PSNR/SSIM results on JPEG deblocking. Bold and underlined numbers denote the best and the second best results.

Dataset	Quality	JPEG		SA-DCT		ARCNN		TNRD		DnCNN		TSSN	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
LIVE1	10	27.77	0.7905	28.65	0.8093	28.98	0.8217	29.15	0.8111	29.19	0.8123	**30.09**	**0.8413**
	20	30.07	0.8683	30.81	0.8781	31.29	0.8871	31.46	0.8769	31.59	0.8802	**32.49**	**0.9000**
	30	31.41	0.9000	32.08	0.9078	32.69	0.9166	32.84	0.9059	32.98	0.9090	**33.95**	**0.9249**
	40	32.35	0.9173	32.99	0.9240	33.63	0.9306	–	–	33.96	0.9247	**34.95**	**0.9381**
Classic5	10	27.82	0.7800	28.88	0.8071	29.04	0.8111	29.28	0.7992	29.40	0.8026	**31.19**	**0.8404**
	20	30.12	0.8541	30.92	0.8663	31.16	0.8694	31.47	0.8576	31.63	0.8610	**33.40**	**0.8881**
	30	31.48	0.8844	32.14	0.8914	32.52	0.8967	32.78	0.8837	32.91	0.8861	**34.68**	**0.9090**
	40	32.43	0.9011	33.00	0.9055	33.34	0.9101	–	–	33.77	0.9003	**35.48**	**0.9205**

Urban100 as an example, we can find that BM3D, DnCNN, IRCNN and FFDNet fail to reconstruct image structures so that the reconstructed results are over-smooth. MemNet keeps the image structure well but fail to remove most of the noise. On the contrary, our TSSN not only generates detail preserving results but also successfully remove all the noise.

Color image denoising. Noisy color images are generated by adding additive white Gaussian noise to clean RGB images with different noise levels $\sigma = 10, 30, 50$ and 70. When applying our model to color image denoising, we set the final reconstruction output as three channels. The PSNR results are demonstrated in Table 3. We can find that our TSSN largely outperforms previous methods and achieves state-of-the-art performance on all the three benchmarks with four noise levels. Previous works (*e.g.*, MemNet) usually use gray image denoising for color image denoising channel by channel. However, we directly learn RGB image denoising in an end-to-end manner. From Table 3, we can find that our TSSN outperforms better than those methods which conduct denoising channel separately, indicating that it is better to conduct color image denoising jointly.

Visual results with noise level ($\sigma = 50$) are shown in Fig. 7. Taking *img*_064 in Urban100 as an example, we can find that other methods generates blurring results while our TSSN reconstructs shaper edges and remove noise greatly. MemNet is designed for gray image denoising, so in color image denoising, we conduct it channel by channel. Compared with MemNet, we can find that directly recovering color images performs better than methods through channels. Compared with other single-stream methods, by compensating deep semantic features with shallow structure features, our TSSN can well reconstruct shape structure lines, which is the reason why TSSN performs better on Urban100 dataset than other datasets.

4.4. JPEG deblocking

We also apply our TSSN to JPEG deblocking task. We compare our model with state-of-the-art JPEG deblocking methods such as SA-DCT (Foi et al., 2007), ARCNN (Dong et al., 2015), TNRD (Chen and Pock, 2017), and DnCNN (Zhang et al., 2017a). We use Matlab JPEG compression algorithm to generate our compressed training and testing images with four different JPEG quality levels $q = 10, 20, 30$ and 40. For fair comparison with other methods, we evaluate our model on the Y channel of transformed YCbCr space.

Average PSNR/SSIM results are shown in Table 4. We can see that our model achieves the best PSNR and SSIM results on both LIVE1 and Classic5 datasets with all JPEG qualities. These comparisons prove that our proposed TSSN is also effective for JPEG deblocking task.

We also show visual results with image quality ($q = 10$) in Fig. 8. We can find that ARCNN performs the worst due to its shallow CNN architecture and weak learning ability. MemNet and DnCNN fails to remove blocking artifacts around edges that leads to blurring results. In contrast, our TSSN reconstructs sharper edges and cleaner background area through the effective two-stream structure.

4.5. Ablation study

4.5.1. Two-stream structure

In order to prove the effectiveness of shallow features, we fix the number of SRBs in deep branch (*e.g.*, 4) and conduct experiments with different numbers of SRBs in shallow branch (*e.g.*, 1+4, 2+4, …), which means different levels of shallowness. We denote that $2+4$ represents the shallow branch has 2 SRBs and the deep one has 4 SRBs. We plot all the validation PSNR results of different settings across the whole training stage in Fig. 9 and all the experiments are conducted on scale x2. We find that $0+4$ is the worst setting in $x+4$ ($x = 0, 1, \dots, 4$) mode, which means that shallow features are essential for better performance. Shallow features play a complementary role for providing with more detailed information. We also conduct other comparisons. When the length of shallow branch is fixed, for example, when comparing $2+4$, $2+5$ and $2+6$, we can find that in deep branch, more SRBs lead to better results. Moreover, we find that $4+4$ is the second worst setting, which means the combination of shallow and deep features is better than the combination of only deep features. Shallow features should be learned by a moderate number of SRBs in the shallow branch. We find that the number of SRBs in shallow branch is better to be set slightly larger than half of the number of SRBs in deep branch. Considering the tradeoff between limited computation resources, inference time and performance, we set the number of SRBs in two branches to be $4+7$ in this paper.

4.5.2. Single-stream vs. Two-stream

To evaluate the necessity of our two-stream structure, we design a single-stream model that has only one deep branch and the shallow features are from its previous layer via skip connection. Therefore shallow and deep features are still utilized in the model but the features are mixed rather than separately learned. For fair comparison, we extract shallow features of the same shallowness and deep features of the same deepness for the two different structures. Quantitative results are shown in Table 5. We can find that our two-stream model greatly outperforms the single-stream model, which means that explicitly learning shallow features through shallow branch is better than directly using shallow features from deep branch.

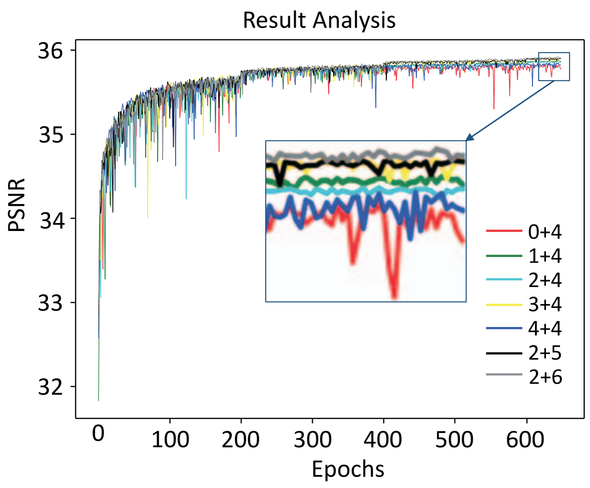


Fig. 9. Convergence analysis on different numbers of SRBs in two branches. The curves record PSNR on validation images of DIV2K dataset across the whole training stage and curves in different colors denote different settings. We zoom in the tail region of curves for better visualization. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 5
Comparison between single-stream structure and two-stream structure. Average PSNR/SSIM for scale factor x2. Bold numbers denote the best performance.

Scale (x2)	Single-stream	Two-stream (TSSN)
Set5	38.20/0.9628	**38.22/0.9629**
Set14	33.86/0.9228	**33.88/0.9235**
BSD100	32.30/0.9042	**32.32/0.9045**
Urban100	32.75/0.9349	**32.91/0.9362**
Manga109	**39.12**/0.9782	**39.12/0.9783**

Table 6
Evaluation of the effect of attention module. Average PSNR/SSIM for scale factor x2. Bold numbers denote the best performance.

Scale(x2)	No attention	Channel	Channel+Spatial
Set5	38.20/0.9628	38.20/0.9628	**38.22/0.9629**
Set14	33.81/0.9221	**33.95**/0.9233	33.88/**0.9235**
BSD100	32.29/0.9041	32.30/0.9043	**32.32/0.9045**
Urban100	32.59/0.9337	32.82/0.9354	**32.91/0.9362**
Manga109	38.97/0.9781	**39.13**/0.9782	39.12/**0.9783**

4.5.3. Attention module

To evaluate the effectiveness of the attention module, we conduct experiments to compare models that with and without attention module. Quantitative results are demonstrated in Table 6. We can find that network with attention module greatly outperforms network without attention module. Moreover, the combination of channel attention and spatial attention is slightly better than only channel attention.

4.5.4. Softmax-L_1 loss

To evaluate the effect of our proposed Softmax-L_1 loss, we conduct experiments with different loss functions. The results are demonstrated in Table 7. We can see that Softmax-L_1 loss can help generate better results than other loss functions. Furthermore, we plot the validation PSNR results of L_1 loss and Softmax-L_1 loss across the whole training process in Fig. 10. We can find that Softmax-L_1 loss facilitate our proposed model to gain better convergence and accuracy.

5. Conclusion

We propose a Two-Stream Sparse Network for accurate and efficient image restoration. Specifically, we propose a two-stream structure to learn shallow and deep features respectively: shallow features provide more detailed texture information while deep features provide

Table 7
Evaluation of the effect of softmax-L_1 loss. Average PSNR/SSIM for scale factor x2. Bold numbers denote the best performance.

Scale(x2)	L_1	L_2	Softmax-L_1
Set5	38.18/0.9627	38.15/0.9626	**38.22/0.9629**
Set14	33.82/0.9218	33.77/0.9212	**33.88/0.9235**
BSD100	32.33/0.9045	32.27/0.9041	**32.32/0.9045**
Urban100	32.85/0.9357	32.65/0.9341	**32.91/0.9362**
Manga109	**39.13**/0.9780	38.90/0.9778	39.12/**0.9783**

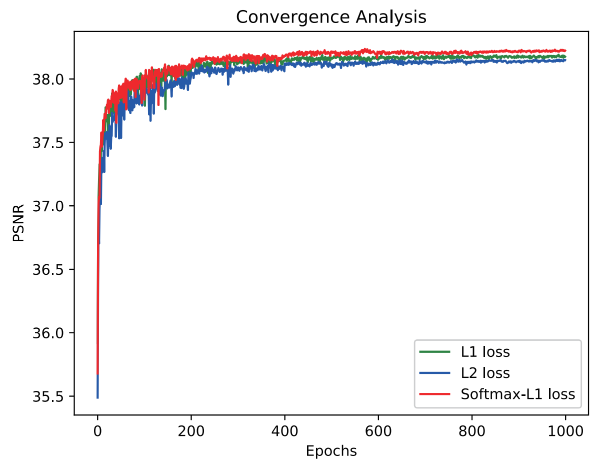


Fig. 10. Convergence analysis on L_1 loss and Softmax-L_1 loss. The curves record PSNR on validation images of DIV2K dataset across the whole training stage. Experiments are conducted on scale x2.

more general semantic information, which are then fused by attention mechanism. In each stream, we introduce sparse residual block with sparse connection and residual learning, which utilizes sparse connection to efficiently gather hierarchical features and helps stack more layers to deepen the network under the same computation budget, thus greatly improving reconstruction quality. Extensive experiments on three representative image restoration tasks demonstrate that our model achieves remarkable improvements in both quantitative metric and visual quality compared to most existing state-of-the-art methods. In future work, we will study how to further enhance the efficiency of our image restoration methods towards real application scenarios.

CRediT authorship contribution statement

Shuhui Wang: Conceptualization, Writing - review & editing, Methodology, Investigation. **Ling Hu:** Writing - original draft, Software. **Liang Li:** Methodology, Writing - review & editing. **Weigang Zhang:** Validation. **Qingming Huang:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the National Key R&D Program of China under Grant 2018AAA0102003, in part by National Natural Science Foundation of China: 61672497, 61771457, 61836002, 61620106009, U1636214, and 61931008, and in part by Key Research Program of Frontier Sciences of CAS: QYZDJ-SSW-SYS013.

References

Agustsson, E., Timofte, R., 2017. NTIRE 2017 challenge on single image super-resolution: Dataset and study. In: CVPR Workshops.
Ahn, N., Kang, B., Sohn, K., 2017. Fast, accurate, and lightweight super-resolution with cascading residual network. In: ECCV.
Bevilacqua, M., Roumy, A., Guillemot, C., Alberi-Morel, M., 2012. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In: BMVC.
Chen, Y., Pock, T., 2017. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. IEEE Trans. Pattern Anal. Mach. Intell.
Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K.O., 2007a. Color image denoising via sparse 3D collaborative filtering with grouping constraint in luminance-chrominance space. In: ICIP.
Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K.O., 2007b. Image denoising by sparse 3-D transform-domain collaborative filtering. IEEE Trans. Image Process.
Dong, C., Deng, Y., Loy, C.C., Tang, X., 2015. Compression artifacts reduction by a deep convolutional network. In: ICCV.
Dong, C., Loy, C.C., He, K., Tang, X., 2014. Learning a deep convolutional network for image super-resolution. In: ECCV.
Dong, C., Loy, C.C., Tang, X., 2016. Accelerating the super-resolution convolutional neural network. In: ECCV.
Foi, A., Katkovnik, V., Egiazarian, K.O., 2007. Pointwise shape-adaptive DCT for high-quality denoising and deblocking of grayscale and color images. IEEE Trans. Image Process.
Haris, M., Shakhnarovich, G., Ukita, N., 2018. Deep back-projection networks for super-resolution. In: CVPR.
Hu, J., Shen, L., Albanie, S., Sun, G., Vedaldi, A., 2018a. Gather-excite: Exploiting feature context in convolutional neural networks. In: NeurIPS.
Hu, J., Shen, L., Sun, G., 2018b. Squeeze-and-excitation networks. In: CVPR.
Hu, L., Wang, S., Li, L., Huang, Q., 2019. Two-stream sparse network for accurate image super-resolution. In: IEEE International Conference on Multimedia & Expo Workshops, ICME Workshops 2019, Shanghai, China, July 8–12, 2019, pp. 258–263.
Huang, G., Liu, Z., Der Maaten, L.V., Weinberger, K.Q., 2017a. Densely connected convolutional networks. In: CVPR. pp. 2261–2269.
Huang, Y., Shao, L., Frangi, A.F., 2017b. DOTE: dual cOnvolutional filTer lEarning for super-resolution and cross-modality synthesis in MRI. In: MICCAI.
Huang, J., Singh, A., Ahuja, N., 2015. Single image super-resolution from transformed self-exemplars. In: CVPR.
Hui, Z., Wang, X., Gao, X., 2018. Fast and accurate single image super-resolution via information distillation network. In: CVPR.
Johnson, J., Alahi, A., Fei-Fei, L., 2016. Perceptual losses for real-time style transfer and super-resolution. In: ECCV.
Kim, J., Lee, J.K., Lee, K.M., 2016a. Accurate image super-resolution using very deep convolutional networks. In: CVPR.
Kim, J., Lee, J.K., Lee, K.M., 2016b. Deeply-recursive convolutional network for image super-resolution. In: CVPR.
Lai, W., Huang, J., Ahuja, N., Yang, M., 2017. Deep Laplacian pyramid networks for fast and accurate super-resolution. In: CVPR.
Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A.P., Tejani, A., Totz, J., Wang, Z., Shi, W., 2017. Photo-realistic single image super-resolution using a generative adversarial network. In: CVPR.
Li, J., Fang, F., Mei, K., Zhang, G., 2018. Multi-scale residual network for image super-resolution. In: ECCV.
Lim, B., Son, S., Kim, H., Nah, S., Lee, K.M., 2017. Enhanced deep residual networks for single image super-resolution. In: CVPR Workshops.
Mao, X., Shen, C., Yang, Y., 2016. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In: NeurIPS.
Martin, D.R., Fowlkes, C.C., Tal, D., Malik, J., 2001. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: ICCV.
Matsui, Y., Ito, K., Aramaki, Y., Fujimoto, A., Ogawa, T., Yamasaki, T., Aizawa, K., 2017. Sketch-based manga retrieval using manga109 dataset. Multimedia Tools Appl.
Seif, G., Androutsos, D., 2018. Large receptive field networks for high-scale image super-resolution. In: CVPR Workshops.
Shi, W., Caballero, J., Huszar, F., Totz, J., Aitken, A.P., Bishop, R., Rueckert, D., Wang, Z., 2016. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In: CVPR.
Tai, Y., Yang, J., Liu, X., 2017a. Image super-resolution via deep recursive residual network. In: CVPR.
Tai, Y., Yang, J., Liu, X., Xu, C., 2017b. MemNet: A persistent memory network for image restoration. In: ICCV.
Tong, T., Li, G., Liu, X., Gao, Q., 2017. Image super-resolution using dense skip connections. In: ICCV.
Wu, Y., Wang, S., Huang, Q., 2017. Online asymmetric similarity learning for cross-modal retrieval. In: CVPR, Honolulu, HI, USA, July 21–26, pp. 3984–3993.
Xiao, A., Wang, Z., Wang, L., Ren, Y., 2018. Super-resolution for "Jilin-1" satellite video imagery via a convolutional network. Sensors.

Yang, J., Wright, J., Huang, T.S., Ma, Y., 2010. Image super-resolution via sparse representation. IEEE Trans. Image Process.

Yoshida, T., Takahashi, T., Deguchi, D., Ide, I., Murase, H., 2012. Robust face super-resolution using free-form deformations for low-quality surveillance video. In: ICME.

Zeiler, M.D., Fergus, R., 2014. Visualizing and understanding convolutional networks. In: ECCV.

Zeiler, M.D., Taylor, G.W., Fergus, R., 2011. Adaptive deconvolutional networks for mid and high level feature learning. In: ICCV.

Zeyde, R., Elad, M., Protter, M., 2010. On single image scale-up using sparse-representations. In: Curves and Surfaces - 7th International Conference.

Zhang, Y., Tian, Y., Kong, Y., Zhong, B., Fu, Y., 2018b. Residual dense network for image super-resolution. In: CVPR.

Zhang, K., Zuo, W., Chen, Y., Meng, D., Zhang, L., 2017a. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. IEEE Trans. Image Process.

Zhang, K., Zuo, W., Gu, S., Zhang, L., 2017b. Learning deep CNN denoiser prior for image restoration. In: CVPR.

Zhang, K., Zuo, W., Zhang, L., 2018a. FFDNet: Toward a fast and flexible solution for CNN-based image denoising. IEEE Trans. Image Process.

Zhao, H., Gallo, O., Frosio, I., Kautz, J., 2017. Loss functions for image restoration with neural networks. IEEE Trans. Comput. Imag.

Zhu, L., Deng, R., Maire, M., Deng, Z., Mori, G., Tan, P., 2018. Sparsely aggregated convolutional networks. In: ECCV.