DEEP LEARNING FOR IMAGE RESTORATION AND ENHANCEMENT

BY

DING LIU

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2018

Urbana, Illinois

ABSTRACT

Image restoration is the process of recovering an original clean image from its degraded version, and image enhancement takes the goal of improving the image quality either objectively or subjectively. Both of them play a key part in computer vision and image processing and have broad applications in industry. The past few years have witnessed the revival of deep learning in computer vision, and substantial progress has been made due to the use of deep neural networks.

In this dissertation, we use deep learning to address the problems of image restoration and enhancement, with the focus on the following topics: image and video super-resolution (SR), as well as image denoising. For these problems, deep neural networks are generally used as a regression model to predict the original clean image content from the input. However, designing a network structure that can effectively exploit the intrinsic image properties to achieve remarkable performances is not a trivial task.

For image SR, several models based on deep neural networks have been recently proposed and attained superior performance that overshadows all previous handcrafted models. The question then arises whether large-capacity and data-driven models have become the dominant solution to this ill-posed problem. We argue that domain expertise represented by the conventional sparse coding model is still valuable, and it can be combined with the key ingredients of deep learning to achieve further improved results. We experimentally show that a sparse coding model particularly designed for SR can be incarnated as a neural network, which can be trained from end to end. The interpretation of the network based on sparse coding leads to much more efficient and effective training, as well as a reduced model size. In addition, we design a unified framework to learn a mixture of sub-networks for image SR so as to further boost SR accuracy.

Video SR aims to generate a high-resolution (HR) frame from multiple low-resolution (LR) frames in a local temporal window. The inter-frame temporal relation is as crucial as the intra-frame spatial relation for tackling this problem. We design deep networks for utilizing the temporal relation from two aspects. First, we propose a temporal adaptive neural network that can adaptively determine the optimal scale of temporal dependency. Filters on various temporal scales are applied to the input LR sequence before their responses are adaptively aggregated. Second, we reduce the complexity of motion between neighboring frames using a spatial alignment network which is much more robust and efficient than competing alignment methods and can be jointly trained with the temporal adaptive network.

Image denoising, as another important task of image restoration, is dedicated to recovering the underlying image signal from its noisy measurement. First we customize a convolutional neural network for image denoising. Second we investigate the mutual relation between image denoising and high-level vision tasks in a deep learning fashion when image denoising serves as a preprocessing step for high-level vision tasks. We design a network that cascades two modules for image denoising and various high-level tasks, and use the joint loss for updating only the denoising network via back-propagation.

Self-similarity in natural images is widely used for image restoration by many classic approaches. We propose a non-local recurrent network (NLRN) as the first attempt to incorporate non-local operations into a recurrent neural network (RNN) for image restoration. Unlike existing methods that measure self-similarity in an isolated manner, the proposed non-local module can be flexibly integrated into existing deep networks for end-to-end training to capture deep feature correlation between each location and its neighborhood. We fully use an RNN for its parameter efficiency and allow deep feature correlation to be propagated along adjacent recurrent states. This design boosts robustness against inaccurate correlation estimation due to severely degraded images. Finally, we show that it is essential to choose a proper neighborhood size for computing deep feature correlation given degraded images, in order to obtain the best restoration performance.

To my parents, my elder brother, and my girlfriend.

ACKNOWLEDGMENTS

This dissertation would not have been possible without the help of a number of people during my PhD study. First, I would like to express my sincere gratitude to my research adviser, Professor Thomas S. Huang, for his generous support, guidance and kindness, and for providing me with the opportunity to pursue my passion. It is an honor and a privilege to be his student and I have learned a lot from his humble and respectful personality especially.

I also would like to thank the members of my doctoral committee, Professors Mark Hasegawa-Johnson, Minh Do and Alexander Schwing for their insightful comments and guidance when helping me prepare this dissertation.

During my PhD study over the past few years, I had a great honor to work with many talented people in the Image Formation and Processing (IFP) group, including Shiyu Chang, Bowen Cheng, Yuchen Fan, Yang Fu, Kuangxiao Gu, Wei Han, Jianbo Jiao, Pooya Khorrami, Xianming Liu, Tom Le Paine, Zengming Shen, Zhiqiang Shen, Honghui Shi, Jiangping Wang, Xinchao Wang, Zhangyang Wang, Zhaowen Wang, Yunchao Wei, Ning Xu, Yingzhen Yang, Jiahui Yu, Haichao Yu, Hanchao Yu, Xiaolin Zhang and Yuqian Zhou. I appreciate all of them for insightful daily discussions and fruitful research collaborations.

I am thankful to Professor Nasser Nasrabadi of West Virginia University, Dr. Xiaodi Hou, Dr. Yuxiao Hu and Dr. Lei Zhang. I feel very lucky to have had the chance to work under their supervision as an intern. The useful guidance that they gave me will benefit me for my career in the future. I also thank Dr. Jianchao Yang, Dr. Bihan Wen and Research Assistant Professor Chen Change Loy of the Chinese University of Hong Kong, who are my collaborators on the research work in this dissertation.

Last, I would like to give special thanks to my parents and my elder brother in China, as well as my girlfriend, who have always provided their unconditional love, support and understanding throughout my PhD study.

CONTENTS

Chapter 1

INTRODUCTION

Image restoration aims to restore an original clean image from its corrupted version, while image enhancement usually focuses on improving the interpretability or perception of information in images either objectively or subjectively. They include the topics such as image denoising, image super-resolution, image deblurring, haze removal, image compression artifact removal and so on. They belong to low-level computer vision tasks and have been widely used in a number of applications. Recently, an increasing interest in these fundamental topics has emerged from not only the research communities of computer vision and image processing but also industry, and substantial progress has been achieved. Each step forward facilitates the use of visual data for the fulfillment of further tasks.

In the past few years, we have witnessed the revival of deep learning in computer vision, mainly due to the large model capacity of deep neural networks, the occurrence of large labeled datasets, and the surge of computing power. Deep learning based approaches have been dominating in quite a number of computer vision tasks, including image classification, face recognition, object detection and so on.

In this dissertation, we propose to use deep learning to study image restoration and enhancement, with the focus on the following topics: image and video super-resolution (SR), as well as image denoising. A straightforward approach to increase the network model capacity by adding more layers and raising the number of trainable parameters is usually not the optimal solution for these tasks. To this end, the core idea is to design task-specific network architecture to model the complex mapping between the input image and its output. In particular, we propose the networks of interpretable design based on the natural priors, such as sparsity and self-similarity. We also develop a neural network structure which enables us to study the influence of high-level vision tasks on image restoration. Specifically, this dissertation addresses the following topics.

Single Image Super-Resolution Single image super-resolution is an ill-posed problem of restoring a visually pleasing high-resolution (HR) image from its low-resolution (LR) observation. HR images have higher pixel densities and finer details than LR images. To regularize the solution of the problem, previous methods have focused on designing good priors for natural images such as sparse representation, or directly learning the priors from a large data set with models such as deep neural networks. We argue that domain expertise from the conventional sparse coding model can be combined with the key ingredients of deep learning to achieve further improved results. We demonstrate that a sparse coding model particularly designed for super-resolution can be incarnated as a neural network with the merit of end-to-end optimization over training data. The network has a cascaded structure which boosts the SR performance for both fixed and incremental scaling factors. The proposed training and testing schemes can be extended for robust handling of images with additional degradation such as noise and blurring. A subjective assessment is conducted and analyzed in order to thoroughly evaluate various SR techniques. Our proposed model is tested on several benchmark datasets, and shows superiority over recent competing methods for various scaling factors both quantitatively and perceptually.

Furthermore, we propose learning a mixture of SR inference modules in a unified framework to further enhance the performance of single image SR. Specifically, a number of SR inference modules specialized in different image local patterns are first independently applied on the LR image to obtain various HR estimates, and the resultant HR estimates are adaptively aggregated to form the final HR image. By selecting neural networks as the SR inference module, the whole procedure can be incorporated into a unified network and be optimized jointly. Extensive experiments are conducted to investigate the relation between restoration performance and different types of network architecture. Compared with other current image SR approaches, our proposed method achieves better restoration results on a wide range of images consistently while allowing more flexible design choices.

Video Super-Resolution Video SR aims at estimating an HR video sequence from an LR one. Given that deep learning has been successfully applied to the task of single image SR, which demonstrates the strong capability of neural networks for modeling spatial relation within one single image, the key challenge to conduct video SR is how to efficiently and effectively exploit the temporal dependency among consecutive LR frames other than the spatial relation. However, this remains challenging because complex motion is difficult to model and can have detrimental effects if not handled properly. We tackle the problem of learning temporal dynamics from two aspects. First, we propose a temporal adaptive neural network that can adaptively determine the optimal scale of temporal dependency. Inspired by the Inception module in GoogLeNet [1], filters of various temporal scales are applied to the input LR sequence before their responses are adaptively aggregated, in order to fully exploit the temporal relation among consecutive LR frames. Second, we decrease the complexity of motion among neighboring frames using a spatial alignment network that can be end-to-end trained with the temporal adaptive network and has the merit of increasing the robustness to complex motion and the efficiency compared to competing image alignment methods. We show that the temporal adaptive design considerably improves SR quality over its plain counterparts, and the spatial alignment network is able to attain comparable SR performance with the sophisticated optical flow based approach, but requires much less running time. Overall our proposed model with learned temporal dynamics is shown to achieve state-of-the-art SR results in terms of not only spatial consistency but also temporal coherence on public video datasets.

Image Denoising Conventionally, image denoising and high-level vision tasks are handled separately in computer vision. We cope with the two jointly and explore the mutual influence between them. First we propose a convolutional neural network for image denoising which achieves the state-of-the-art performance. Second we propose a deep neural network solution that cascades two modules for image denoising and various high-level tasks, respectively, and use the joint loss for updating only the denoising network via back-propagation. We demonstrate that on one hand, the proposed denoiser has the generality to overcome the performance degradation of different high-level vision tasks. On the other hand, with the guidance of high-level vision information, the denoising network can generate more visually appealing results. To the best of our knowledge, this is the first work investigating the benefit of exploiting image semantics simultaneously for image denoising and high-level vision tasks via deep learning.

General Image Restoration Many classic methods have shown non-local self-similarity in natural images to be an effective prior for image restoration. However, it remains unclear and challenging to make use of this intrinsic property via deep networks. We propose a non-local recurrent network (NLRN) as the first attempt to incorporate non-local operations into a recurrent neural network (RNN) for image restoration. We make the following contributions: (1) Unlike existing methods that measure self-similarity in an isolated manner, the proposed non-local module can be flexibly integrated into existing deep networks for end-to-end training to capture deep feature correlation between each location and its neighborhood. (2) We fully employ the RNN structure for its parameter efficiency and allow deep feature correlation to be propagated along adjacent recurrent states. This new design boosts robustness against inaccurate correlation estimation due to severely degraded images. (3) We show that it is necessary to maintain a confined neighborhood for computing deep feature correlation given degraded images. This is in contrast to existing practice [2] that deploys the whole image. Extensive experiments on both image denoising and super-resolution tasks are conducted. Thanks to the recurrent non-local operations and correlation propagation, the proposed NLRN achieves results superior to those of the state-of-the-art methods with far fewer parameters.

The rest of this dissertation is organized as follows. We first introduce the sparse coding based network for single image SR in Chapter 2. Learning a mixture of deep networks for image SR to further boost SR accuracy is then discussed in Chapter 3. Our deep learning based approach for video SR is presented in Chapter 4. The problem of image denoising and its connection with high-level vision tasks is explored in Chapter 5. A non-local recurrent network for general image restoration is provided in Chapter 6. Finally, Chapter 7 concludes with a summary.

Chapter 2

SPARSE CODING BASED NETWORK FOR SINGLE IMAGE SUPER-RESOLUTION

2.1 Introduction

Single image super-resolution is usually cast as an inverse problem of recovering the original high-resolution (HR) image from one low-resolution (LR) observation image. Since the known variables in LR images are greatly outnumbered by the unknowns in typically desired HR images, this problem is highly ill-posed and has limited the use of SR techniques in many practical applications [3, 4].

A large number of single image SR methods have been proposed, exploiting various priors of natural images to regularize the solution of SR. Analytical priors, such as bicubic interpolation, work well for smooth regions, while image models based on statistics of edges [5] and gradients [6] can recover sharper structures. In the patch-based SR methods, HR patch candidates are represented as the sparse linear combination of dictionary atoms trained from external databases [7, 8, 9], or recovered from similar examples in the LR image itself at different locations and across different scales [10, 11]. A regression model is built between LR and HR patches in [12, 13]. A comprehensive review of more SR methods can be found in [14].

More recently, inspired by the great success achieved by deep learning [15] in other computer vision tasks, researchers have begun to use neural networks with deep architecture for image SR. Multiple layers of collaborative auto-encoders are stacked together in [16, 17] for robust matching of self-similar patches. Deep convolutional neural networks (CNN) [18] and deconvolutional networks [19] are designed that directly learn the non-linear mapping from LR space to HR space in a way similar to coupled sparse coding [8]. As these deep networks allow end-to-end training of all the model components between LR input and HR output, significant improvements have been observed over their shadow counterparts.

The networks in [16, 18] are built with generic architecture, which means all their knowledge about SR is learned from training data. On the other hand, people's domain expertise for the SR problem, such as natural image prior and image degradation model, is largely ignored in deep learning based approaches. It is then worthwhile to investigate whether domain expertise can be used to design better deep model architecture, or whether deep learning can be leveraged to improve the quality of handcrafted models.

In this chapter, we extend the conventional sparse coding model [7] using several key ideas from deep learning, and show that domain expertise is complementary to large learning capacity in further improving SR performance. First, based on the learned iterative shrinkage and thresholding algorithm (LISTA) [20], we implement a feed-forward neural network in which each layer strictly corresponds to one step in the processing flow of sparse coding based image SR. In this way, the sparse representation prior is effectively encoded in our network structure; at the same time, all the components of sparse coding can be trained jointly through back-propagation. This simple model, which is named sparse coding based network (SCN), achieves notable improvement over the generic CNN model [18] in terms of both recovery accuracy and human perception, and yet has a compact model size. Moreover, with the correct understanding of each layer's physical meaning, we have a more principled way to initialize the parameters of SCN, which helps to improve optimization speed and quality.

A single network is only able to perform image SR by a particular scaling factor. In [18], different networks are trained for different scaling factors. In this section, we propose a cascade of multiple SCNs to achieve SR for arbitrary factors. This approach, motivated by the self-similarity based SR approach [10], not only increases the scaling flexibility of our model, but also reduces artifacts for large scaling factors. Moreover, inspired by the multi-pass scheme of image denoising [21], we demonstrate that the SR results can be further enhanced by cascading multiple SCNs for SR of a fixed scaling factor. The cascade of SCNs (CSCN) can also benefit from the end-to-end training of deep network with a specially designed multi-scale cost function.

In practical SR scenarios, the real LR measurements usually suffer from various types of corruptions, such as noise and blurring. Sometimes the degradation process is even too complicated or unclear. We propose several schemes using our SCN to robustly handle such practical SR cases. When the degradation mechanism is unknown, we fine-tune the generic SCN with the requirement of only a small amount of real training data and manage to adapt our model to the new scenario. When the forward model for LR generation is clear, we propose an iterative SR scheme incorporating SCN with additional regularization based on priors from the degradation mechanism.

Subjective assessment is important to the SR technology because the commercial products equipped with such technology are usually evaluated subjectively by the end users. In order to thoroughly compare our model with other prevailing SR methods, we conduct a systematic subjective evaluation among these methods, in which the assessment results are statistically analyzed and one score is given for each method.

In short, the contributions of this section include:

- combining the domain expertise of sparse coding and the merits of deep learning to achieve better SR performance with faster training and smaller model size;
- exploring network cascading for arbitrary scaling factors in order to further enhance SR performance;
- utilizing SCN to robustly handle the practical SR scenarios with non-ideal LR measurements;
- conducting a subjective evaluation on a number of recent state-of-the-art SR methods.

2.2 Related Work

2.2.1 Image SR Using Sparse Coding

The sparse representation based SR method [7] models the transform from each local patch $\boldsymbol{y} \in \mathbb{R}^{m_y}$ in the bicubic-upscaled LR image to the corresponding patch $\boldsymbol{x} \in \mathbb{R}^{m_x}$ in the HR image. The dimension m_y is not necessarily the same as m_x when image features other than raw pixels are used to represent patch $\boldsymbol{y}$. It is assumed that the LR(HR) patch $\boldsymbol{y}(\boldsymbol{x})$ can be represented with respect to an overcomplete dictionary $\boldsymbol{D}_y(\boldsymbol{D}_x)$ using some sparse linear coefficients $\boldsymbol{\alpha}_y(\boldsymbol{\alpha}_x) \in \mathbb{R}^n$, which are known as sparse code. Since the degradation process from $\boldsymbol{x}$ to $\boldsymbol{y}$ is nearly linear, the patch pair can share the same sparse code $\boldsymbol{\alpha}_y = \boldsymbol{\alpha}_x = \boldsymbol{\alpha}$ if the dictionaries $\boldsymbol{D}_y$ and $\boldsymbol{D}_x$ are defined properly. Therefore, for an input LR patch $\boldsymbol{y}$, the HR patch can be recovered as

$$\boldsymbol{x} = \boldsymbol{D}_x\boldsymbol{\alpha}, \quad \text{s.t. } \boldsymbol{\alpha} = \arg\min_{\boldsymbol{z}} \|\boldsymbol{y} - \boldsymbol{D}_y\boldsymbol{z}\|_2^2 + \lambda\|\boldsymbol{z}\|_1, \tag{2.1}$$

where $\|\cdot\|_1$ denotes the ℓ_1 norm which is convex and sparsity-inducing, and λ is a regularization coefficient.

In order to learn the dictionary pair $(\boldsymbol{D}_y, \boldsymbol{D}_x)$, the goal is to minimize the recovery error of $\boldsymbol{x}$ and $\boldsymbol{y}$, and thus the loss function L in [8] is defined as

$$L = \frac{1}{2}\left(\gamma\|\boldsymbol{x} - \boldsymbol{D}_x\boldsymbol{z}\|_2^2 + (1-\gamma)\|\boldsymbol{y} - \boldsymbol{D}_y\boldsymbol{z}\|_2^2\right), \tag{2.2}$$

where γ $(0 < \gamma \leq 1)$ balances the two reconstruction errors. Then the optimal dictionary pair $\{\boldsymbol{D}_x^*, \boldsymbol{D}_y^*\}$ can be found by minimizing the empirical expectation of (2.2) over all the training LR/HR pairs,

$$\begin{gathered}
\min_{\boldsymbol{D}_x,\boldsymbol{D}_y} \frac{1}{N}\sum_{i=1}^{N} L(\boldsymbol{D}_x, \boldsymbol{D}_y, \boldsymbol{x}_i, \boldsymbol{y}_i) \\
\text{s.t. } \boldsymbol{z}_i = \arg\min_{\boldsymbol{\alpha}} \|\boldsymbol{y}_i - \boldsymbol{D}_y\boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_1, i = 1, 2, ..., N, \\
\|D_x(:,k)\|_2 \leq 1, \quad \|D_y(:,k)\|_2 \leq 1, k = 1, 2, ..., K.
\end{gathered} \tag{2.3}$$

Since the objective function in (2.2) is highly nonconvex, the dictionary pair $\boldsymbol{D}_y, \boldsymbol{D}_x)$ is usually learned alternatively while keeping the other fixed [8].

2.2.2 Network Implementation of Sparse Coding

There is an intimate connection between sparse coding and neural networks, which has been well studied in [22, 20]. A feed-forward neural network as illustrated in Figure 2.1 is proposed in [20] to efficiently approximate the sparse code $\boldsymbol{\alpha}$ of input signal $\boldsymbol{y}$ as it would be obtained by solving (2.1) for a given dictionary $\boldsymbol{D}_y$. The network has a finite number of recurrent stages, each of which updates the intermediate sparse code according to

$$\boldsymbol{z}_{k+1} = h_{\boldsymbol{\theta}}(\boldsymbol{W}\boldsymbol{y} + \boldsymbol{S}\boldsymbol{z}_k), \tag{2.4}$$

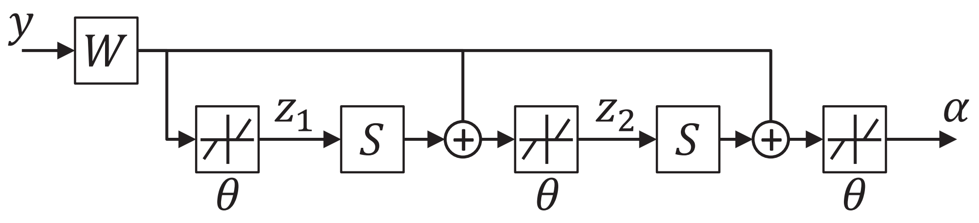


Figure 2.1: A LISTA network [20] with two time-unfolded recurrent stages, whose output $\boldsymbol{\alpha}$ is an approximation of the sparse code of input signal $\boldsymbol{y}$. The linear weights $\boldsymbol{W}$, $\boldsymbol{S}$ and the shrinkage thresholds $\boldsymbol{\theta}$ are learned from data.

where $h_{\boldsymbol{\theta}}$ is an element-wise shrinkage function defined as $[h_{\boldsymbol{\theta}}(\boldsymbol{a})]_i = \text{sign}(a_i)(|a_i| - \theta_i)_+$ with positive thresholds $\boldsymbol{\theta}$.

Different from the iterative shrinkage and thresholding algorithm (ISTA) [23, 24] which finds an analytical relationship between network parameters (weights $\boldsymbol{W}$, $\boldsymbol{S}$ and thresholds $\boldsymbol{\theta}$) and sparse coding parameters ($\boldsymbol{D}_y$ and λ), the authors of [20] learn all the network parameters from training data using a back-propagation algorithm called learned ISTA (LISTA). In this way, a good approximation of the underlying sparse code can be obtained within a fixed number of recurrent stages.

2.2.3 Generic Convolutional Neural Network for SR

As a successful example of deep learning for single image SR, Dong et al. [18] propose a fully convolutional neural network to directly learn the mapping from the input LR image to the output HR image. It is designed to utilize three convolutional layers to mimic the patch extraction and representation, non-linear mapping and reconstruction of the sparse representation based SR methods, respectively. Due to the end-to-end training strategy that jointly optimizes all the parameters and the large learning capacity of neural networks, this method notably outperforms its conventional shadow counterpart.

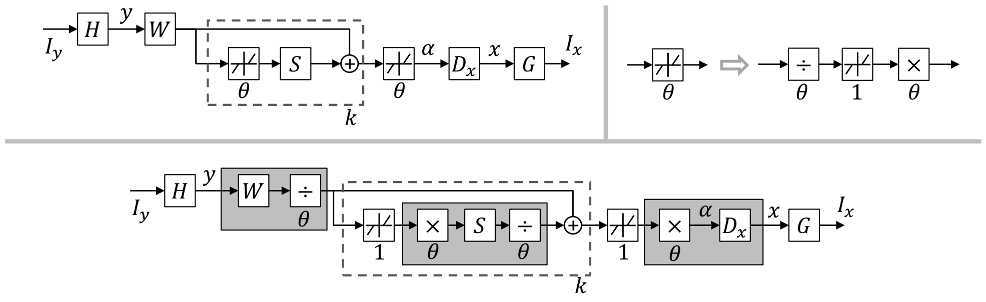


Figure 2.2: Top left: the proposed SCN model with a patch extraction layer $\boldsymbol{H}$, a LISTA sub-network for sparse coding (with k recurrent stages denoted by the dashed box), a HR patch recovery layer $\boldsymbol{D}_x$, and a patch combination layer $\boldsymbol{G}$. Top right: a neuron with an adjustable threshold decomposed into two linear scaling layers and a unit-threshold neuron. Bottom: the SCN re-organized with unit-threshold neurons and adjacent linear layers merged together in the gray boxes.

2.3 Sparse Coding Based Network for Image SR

2.3.1 Network Architecture

Given the fact that sparse coding can be effectively implemented with a LISTA network, it is straightforward to build a multi-layer neural network that mimics the processing flow of the sparse coding based SR method [7]. Like most patch-based SR methods, our sparse coding based network (SCN) takes the bicubic-upscaled LR image $\boldsymbol{I}_y$ as input, and outputs the full HR image $\boldsymbol{I}_x$. Figure 2.2 shows the main network structure, and each of the layers is described in the following.

The input image $\boldsymbol{I}_y$ first goes through a convolutional layer $\boldsymbol{H}$ which extracts features for each LR patch. There are m_y filters of spatial size $s_y \times s_y$ in this layer, so that our input patch size is $s_y \times s_y$ and its feature representation $\boldsymbol{y}$ has m_y dimensions.

Each LR patch $\boldsymbol{y}$ is then fed into a LISTA network with a finite number of k recurrent stages to obtain its sparse code $\boldsymbol{\alpha} \in \mathbb{R}^n$. Each stage of LISTA consists of two linear layers parameterized by $\boldsymbol{W} \in \mathbb{R}^{n \times m_y}$ and $\boldsymbol{S} \in \mathbb{R}^{n \times n}$, and a nonlinear neuron layer with activation function $h_{\boldsymbol{\theta}}$. The activation thresholds $\boldsymbol{\theta} \in \mathbb{R}^n$ are also to be updated during training, which complicates the learning algorithm. To restrict all the tunable parameters in our linear layers, we do a simple trick to rewrite the activation function as

$$[h_{\boldsymbol{\theta}}(\boldsymbol{a})]_i = \operatorname{sign}(a_i)\theta_i(|a_i|/\theta_i - 1)_+ = \theta_i h_1(a_i/\theta_i). \tag{2.5}$$

Eq. (2.5) indicates that the original neuron with an adjustable threshold can be decomposed into two linear scaling layers and a unit-threshold neuron, as shown in the top-right of Figure 2.2. The weights of the two scaling layers are diagonal matrices defined by $\boldsymbol{\theta}$ and its element-wise reciprocal, respectively.

The sparse code $\boldsymbol{\alpha}$ is then multiplied with HR dictionary $\boldsymbol{D}_x \in \mathbb{R}^{m_x \times n}$ in the next linear layer, reconstructing HR patch $\boldsymbol{x}$ of size $s_x \times s_x = m_x$.

In the final layer $\boldsymbol{G}$, all the recovered patches are put back to the corresponding positions in the HR image $\boldsymbol{I}_x$. This is realized via a convolutional filter of m_x channels with spatial size $s_g \times s_g$. The size s_g is determined as the number of neighboring patches that overlap with the same pixel in each spatial direction. The filter will assign appropriate weights to the overlapped recoveries from different patches and take their weighted average as the final prediction in $\boldsymbol{I}_x$.

As illustrated in the bottom of Figure 2.2, after some simple reorganizations of the layer connections, the network described above has some adjacent linear layers which can be merged into a single layer. This helps to reduce the computation load as well as redundant parameters in the network. The layers $\boldsymbol{H}$ and $\boldsymbol{G}$ are not merged because we apply additional nonlinear normalization operations on patches $\boldsymbol{y}$ and $\boldsymbol{x}$, which will be detailed in Section 2.6.1.

Thus, there are totally 5 trainable layers in our network: 2 convolutional layers $\boldsymbol{H}$ and $\boldsymbol{G}$, and 3 linear layers shown as gray boxes in Figure 2.2. The k recurrent layers share the same weights and are therefore conceptually regarded as one. Note that all the linear layers are actually implemented as convolutional layers applied on each patch with filter spatial size of 1×1, a structure similar to the network in network [25]. Also note that all these layers have only weights but no biases (zero biases).

Mean square error (MSE) is employed as the cost function to train the network, and our optimization objective can be expressed as

$$\min_{\boldsymbol{\Theta}} \sum_i \| SCN(\boldsymbol{I}_y^{(i)}; \boldsymbol{\Theta}) - \boldsymbol{I}_x^{(i)} \|_2^2, \tag{2.6}$$

where $\boldsymbol{I}_y^{(i)}$ and $\boldsymbol{I}_x^{(i)}$ are the i-th pair of LR/HR training data, and $SCN(\boldsymbol{I}_y; \boldsymbol{\Theta})$ denotes the HR image for $\boldsymbol{I}_y$ predicted using the SCN model with parameter set $\boldsymbol{\Theta}$. All the parameters are optimized through the standard back-propagation algorithm. Although it is possible to use other cost terms that are more correlated with human visual perception than MSE, our experimental results show that simply minimizing MSE leads to improvement in subjective quality.

2.3.2 Advantages over Previous Models

The construction of our SCN follows exactly each step in the sparse coding based SR method [7]. If the network parameters are set according to the dictionaries learned in [7], it can reproduce almost the same results. However, after training, SCN learns a more complex regression function and can no longer be converted to an equivalent sparse coding model. The advantage of SCN comes from its ability to jointly optimize all the layer parameters from end to end, while in [7] some variables are manually designed and some are optimized individually by fixing all the others.

Technically, our network is also a CNN and it has similar layers as the CNN model proposed in [18] for patch extraction and reconstruction. The key difference is that we have a LISTA sub-network specifically designed to enforce sparse representation prior, while in [18] a generic rectified linear unit (ReLU) [26] is used for nonlinear mapping. Since SCN is designed based on our domain knowledge in sparse coding, we are able to obtain a better interpretation of the filter responses and have a better way to initialize the filter parameters in training. We will see in the experiments that all these contribute to better SR results, faster training speed and smaller model size than a vanilla CNN.

2.3.3 Network Cascade

In this section, we investigate two different network cascade techniques in order to fully exploit our SCN model in SR applications.

Network Cascade for SR of a Fixed Scaling Factor

First, we observe that the SR results can be further improved by cascading multiple SCNs trained for the same objective in (2.6), which is inspired by the multi-pass scheme in [21]. The only difference for training these SCNs is to replace the bicubic interpolated input by its latest HR estimate, while the target output remains the same.

The first SCN acts as a function approximator to model the non-linear mapping from the bicubic upscaled image to the ground-truth image. The following SCN acts as another function approximator, with the starting point changed to a better estimate: the output of its previous SCN.

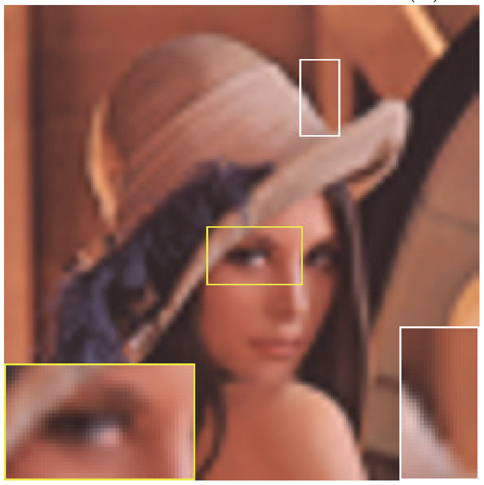
In other words, the cascade of SCNs as a whole can be considered as a new deeper network having more powerful learning capability, which is able to better approximate the mapping from the LR inputs to the HR counterparts, and these SCNs can be trained jointly to pursue even better SR performance.

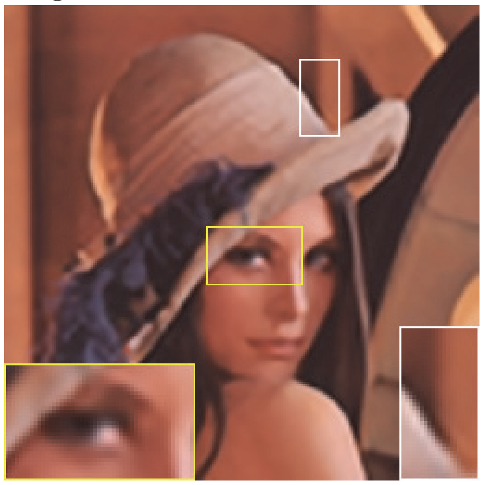
Network Cascade for Scalable SR

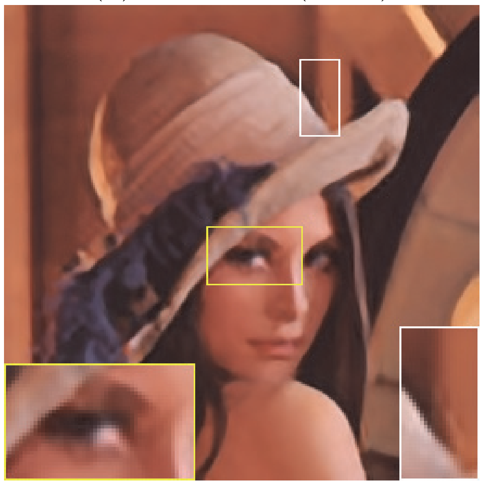
Like most SR models learned from external training examples, the SCN discussed previously can only upscale images by a fixed factor. A separate model needs to be trained for each scaling factor to achieve the best performance, which limits the flexibility and scalability in practical use. One way to overcome this difficulty is to repeatedly enlarge the image by a fixed scale until the resulting HR image reaches a desired size. This practice is commonly adopted in the self-similarity based methods [10, 11, 16], but is not so popular in other cases for the fear of error accumulation during repetitive upscaling.

In our case, however, it is observed that a cascade of SCNs trained for small scaling factors can generate even better SR results than a single SCN trained for a large scaling factor, especially when the target scaling factor is large (greater than 2). This is illustrated by the example in Figure 2.3. Here an input image is magnified by $\times 4$ times in two ways: with a single SCN$\times 4$ model through the processing flow (a) $\rightarrow$ (b) $\rightarrow$ (d), and with a cascade of two SCN$\times 2$ models through (a) $\rightarrow$ (c) $\rightarrow$ (e). It can be seen that the input to the second cascaded SCN$\times 2$ in (c) is already sharper and contains fewer artifacts than the bicubic$\times 4$ input to the single SCN$\times 4$ in (b), which

(a) LR image


(b) bicubic×4 (28.52)


(c) SCN×2 & bicubic×2 (30.27)


(d) SCN×4 (30.22)

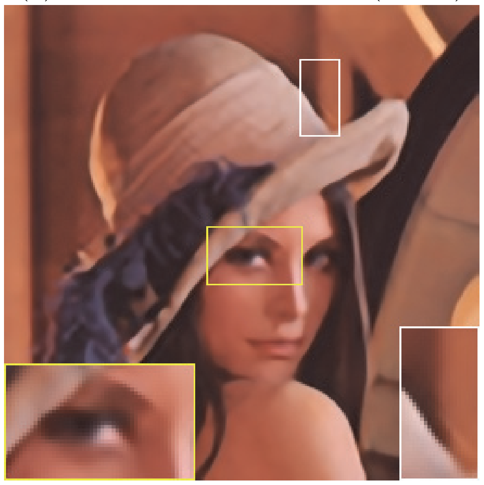

(e) SCN×2 & SCN×2 (30.72)

Figure 2.3: SR results for the "Lena" image upscaled by 4 times. (a) → (b) → (d) represents the processing flow with a single SCN×4 model. (a) → (c) → (e) represents the processing flow with two cascaded SCN×2 models. PSNR is given in parentheses.

naturally leads to the better final result in (e) than in (d).

To get a better understanding of the above observation, we can draw a loose analogy between the SR process and a communication system. Bicubic interpolation is like a noisy channel through which an image is "transmitted" from LR domain to HR domain. And our SCN model (or any SR algorithm)

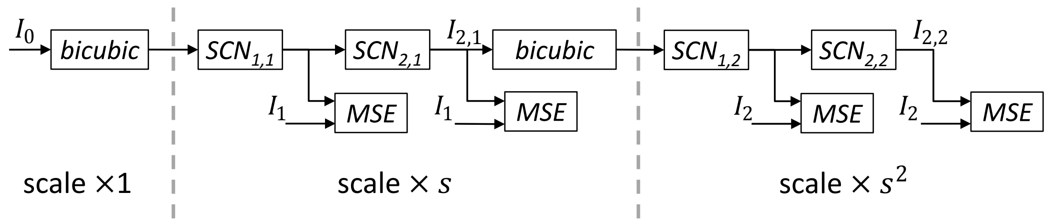


Figure 2.4: Training cascade of SCNs with multi-scale objectives.

behaves as a receiver which recovers clean signals from noisy observations. A cascade of SCNs is then like a set of relay stations that enhance signal-to-noise ratio before the signal becomes too weak for further transmission. Therefore, cascading will work only when each SCN can restore enough useful information to compensate for the new artifacts it introduces as well as the magnified artifacts from previous stages.

Training Cascade of Networks

Taking into account the two aforementioned cascade techniques, we can consider the cascade of all SCNs as a deeper network (CSCN), in which the final output of the consecutive SCNs of the same ground truth is connected to the input of the next SCN with bicubic interpolation in the between. To construct the cascade, besides stacking several SCNs trained individually with respect to (2.6), we can also optimize all of them jointly as shown in Figure 2.4. Without loss of generality, we assume each stage in Section 2.3.3 has the same scaling factor s. Let $\hat{\boldsymbol{I}}_{j,k}$ $(j > 0, k > 0)$ denote the output image of the j-th SCN in the k-th stage upscaled by a total of $\times s^k$ times. In the same stage, each output of SCNs is compared with the associated ground truth image $\boldsymbol{I}_k$ according to the MSE cost, leading to a multi-scale objective function:

$$\min_{\{\boldsymbol{\Theta}_{j,k}\}} \sum_i \sum_j \sum_k \left\| SCN(\hat{\boldsymbol{I}}^{(i)}_{j-1,k}; \boldsymbol{\Theta}_{j,k}) - \boldsymbol{I}^{(i)}_k \right\|_2^2, \tag{2.7}$$

where i denotes the data index, and j, k denote the SCN index. For simplicity of notation, $\hat{\boldsymbol{I}}_{0,k}$ specially denotes the bicubic interpolated image of the final output in the $(k-1)$-th stage upscaled by a total of $\times s^{k-1}$ times. This multi-scale objective function makes full use of the supervision information in all scales. All the layer parameters $\{\boldsymbol{\Theta}_{j,k}\}$ in (2.7) could be optimized from end to end by back-propagation. The SCNs share the same training objective and can be trained simultaneously, taking advantage of the merit of deep learning. For the SCNs with different training objectives, we use a greedy algorithm here to train them sequentially from the beginning of the cascade so that we do not need to care about the gradient of bicubic layers. Applying back-propagation through a bicubic layer or its trainable surrogate will be considered in future work.

2.4 Robust SR for Real Scenarios

Most recent SR works generate the LR images for both training and testing by downscaling HR images using bicubic interpolation [7, 27]. However, this assumption of the forward model may not always hold in practice. For example, the real LR measurements are usually blurred, or corrupted with noise. Sometimes, the LR generation mechanism may be complicated, or even unknown. We now investigate the practical SR problem and propose two approaches to handle such non-ideal LR measurements, using the generic SCN. In the case that the underlying mechanism of the real LR generation is unclear or complicated, we propose the data-driven approach by fine-tuning the learned generic SCN with a limited number of real LR measurements as well as their corresponding HR counterparts. On the other hand, if the real training samples are unavailable but the LR generation mechanism is clear, we formulate this inverse problem as the regularized HR image reconstruction problem which can be solved using iterative methods. The proposed methods demonstrate the robustness of our SCN model to different SR scenarios. In the following, we elaborate on the details of these two approaches, respectively.

2.4.1 Data-Driven SR by Fine-Tuning

Deep learning models can be efficiently transferred from one task to another by re-using the intermediate representation in the original neural network [28]. This method has proven successful on a number of high-level vision tasks, even if there is a limited amount of training data in the new task [29].

The success of super-resolution algorithms usually highly depends on the accuracy of the model of the imaging process. When the underlying mechanism of the generation of LR images is not clear, we can take advantage of the aforementioned merit of deep learning models by learning our model in a data-driven manner, to adapt it for a particular task. Specifically, we start training from the generic SCN model while using a very limited amount of training data from a new SR scenario, and manage to adapt it to the new SR scenario and obtain promising results. In this way, it is demonstrated that the SCN has the strong capability of learning complex mappings from the non-ideal LR measurements to their HR counterparts as well as the high flexibility of adapting to various SR tasks.

2.4.2 Iterative SR with Regularization

The second approach considers the case that the mechanism of generating the real LR images is relatively simple and clear, indicating the training data is always available if we synthesize LR images with the known degradation process. We propose an iterative SR scheme which incorporates the generic SCN model with additional regularization based on task-related priors (e.g. the known kernel for deblurring, or the data sparsity for denoising). In this section, we specifically discuss handling blurred and noisy LR measurements in detail as examples, though the iterative SR methods can be generalized to other practical imaging models.

Blurry Image Upscaling

The real LR images can be generated with various types of blurring. Directly applying the generic SCN model is obviously not optimal. Instead, with the known blurring kernel, we propose to estimate the regularized version of the HR image $\hat{\boldsymbol{I}}_x$ based on the directly upscaled image $\tilde{\boldsymbol{I}}_x$ by the learned SCN as follows:

$$\hat{\boldsymbol{I}}_x = \arg\min_{\boldsymbol{I}} \|\boldsymbol{I} - \tilde{\boldsymbol{I}}_x\|_2, \quad \text{s.t. } D{\cdot}B \cdot \boldsymbol{I} = \boldsymbol{I}_y^0, \tag{2.8}$$

where $\boldsymbol{I}_y^0$ is the original blurred LR input, and the operators B and D are blurring and sub-sampling respectively. Similar to the previous work [7], we use back-projection to iteratively estimate the regularized HR input on which our model can perform better. Specifically, given the regularized estimate

$\hat{\boldsymbol{I}}_x^{i-1}$ at iteration $i-1$, we estimate a less blurred LR image $\boldsymbol{I}_y^{i-1}$ by downsampling $\hat{\boldsymbol{I}}_x^{i}$ using bicubic interpolation. The upscaled $\tilde{\boldsymbol{I}}_x^{i}$ by learned SCN serves as the regularizer for the i-th iteration as follows:

$$\hat{\boldsymbol{I}}_x^{i} = \arg\min_{\boldsymbol{I}} \|\boldsymbol{I} - \tilde{\boldsymbol{I}}_x^{i}\|_2^2 + \|D \cdot B \cdot \boldsymbol{I} - \boldsymbol{I}_y^0\|_2^2. \tag{2.9}$$

Here we use a penalty method to form an unconstrained problem. The upscaled HR image $\tilde{\boldsymbol{I}}_x^{i}$ can be computed as $SCN(\boldsymbol{I}_y^{i-1}, \boldsymbol{\Theta})$. The same process is repeated until convergence. We have applied the proposed iterative scheme to LR images generated from Gaussian blurring and sub-sampling as an example. The empirical performance is illustrated in Section 2.6.

Noisy Image Upscaling

Noise is a ubiquitous cause of corruption in image acquisition. State-of-the-art image denoising methods usually adopt priors such as patch similarity [30], patch sparsity [31, 21], or both [32], as regularizers in image restoration. In this section, we propose a regularized noisy image upscaling scheme for specifically handling noisy LR images in order to obtain improved SR quality. Though any denoising algorithm can be used in our proposed scheme, here we apply spatial similarity combined with transform domain image patch group-sparsity as our regularizer [32], to form the regularized iterative SR problem as an example.

Similar to the method in Section 2.4.2, we iteratively estimate the less noisy HR image from the denoised LR image. Given the denoised LR estimate $\hat{\boldsymbol{I}}_y^{i-1}$ at iteration $i-1$, we directly upscale it, using the learned generic SCN, to obtain the HR image $\hat{\boldsymbol{I}}_x^{i-1}$. It is then downsampled using bicubic interpolation, to generate the LR image $\tilde{\boldsymbol{I}}_y^{i}$, which is used in the fidelity term in the i-th iteration of LR image denoising. The same process is repeated until convergence. The iterative LR image denoising problem is formulated as follows:

$$\begin{aligned} \left\{\hat{\boldsymbol{I}}_y^{i}, \{\hat{\alpha}_i\}\right\} = &\arg\min_{\boldsymbol{I}, \{\boldsymbol{\alpha}_i\}} \|\boldsymbol{I} - \tilde{\boldsymbol{I}}_y^{i}\|_2^2 \\ &+ \sum_{j=1}^{N} \left\{\|W_{3D} G_j \boldsymbol{I} - \alpha_j\|_2^2 + \tau \|\alpha_j\|_0\right\}, \end{aligned} \tag{2.10}$$

where the operator G_j generates the 3D vectorized tensor, which groups the j-th overlapping patch from the LR image I, together with the spatially similar patches within its neighborhood by block matching [32]. The codes $\{\alpha_j\}$ of the patch groups in the domain of 3D sparsifying transform W_{3D} are sparse, which is enforced by the l_0 norm penalty [33]. The weight τ controls the sparsity level, which normally depends on the remaining noise level in $\tilde{I}_y^i$ [34, 33].

In (2.10), we use the patch group sparsity as our denoising regularizer. The 3D sparsifying transform W_{3D} can be commonly used analytical transforms, such as the discrete cosine transform (DCT) or the wavelet transform. The state-of-the-art BM3D denoising algorithm [32] is based on such an approach, but further improved by more sophisticated engineering stages. In order to achieve the best practical SR quality, we demonstrate the empirical performance comparison using BM3D as the regularizer in Section 2.6. Additionally, our proposed iterative method is a general practical SR framework, which is not dedicated to SCN. One can conveniently extend it to other SR methods, which generates $\tilde{I}_y^i$ in i^{th} iteration. The performances of these methods are compared in Section 2.6.

2.5 Subjective Evaluation Protocol

Subjective perception is an important metric to evaluate SR techniques for commercial use, other than the quantitative evaluation. In order to more thoroughly compare various SR methods and quantify the subjective perception, we utilize an online platform for subjective evaluation of SR results from several methods [35], including bicubic, SC [8], SE [11], self-example regression (SER) [36], CNN [18] and CSCN. Each participant is invited to conduct several pair-wise comparisons of SR results from different methods. The SR methods of displayed SR images in each pair are randomly selected. Ground truth HR images are also included when they are available as references. For each pair, the participant needs to select the better one in terms of perceptual quality. A snapshot of our evaluation web page[1] is shown in Figure 2.5.

Specifically, there are SR results over 6 images with different scaling fac-

[1]`www.ifp.illinois.edu/~wang308/survey`

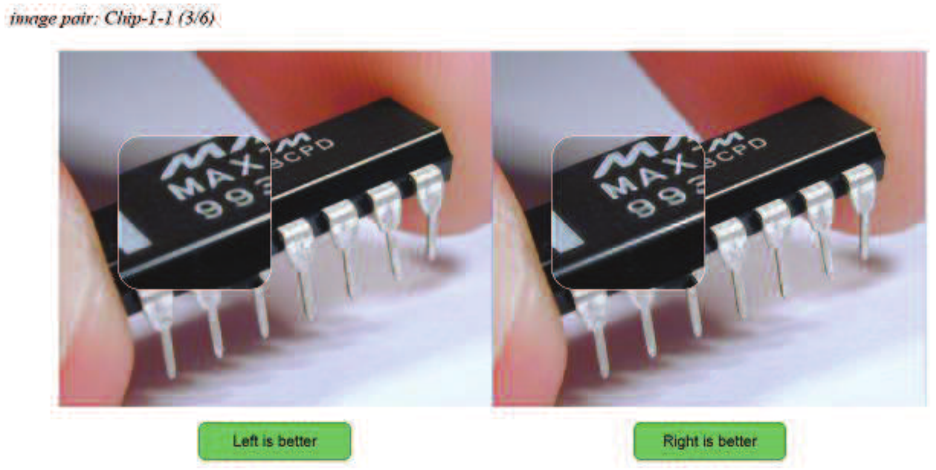


Figure 2.5: The user interface of a web-based image quality evaluation, where two images are displayed side by side and local details can be magnified by moving the mouse over the corresponding region.

tors: “kid”×4, “chip”×4, “statue”×4, “lion”×3, “temple”×3 and “train”×3. The images are shown in Figure 2.6. All the visual comparison results are then summarized into a 7×7 winning matrix for 7 methods (including ground truth). A Bradley-Terry [37] model is calculated based on these results and the subjective score is estimated for each method according to this model. In the Bradley-Terry model, the probability that an object X is favored over Y is assumed to be

$$p(X \succ Y) = \frac{e^{s_X}}{e^{s_X} + e^{s_Y}} = \frac{1}{1 + e^{s_Y - s_X}}, \tag{2.11}$$

where s_X and s_Y are the subjective scores for X and Y. The scores $\boldsymbol{s}$ for all the objects can be jointly estimated by maximizing the log likelihood of the pairwise comparison observations:

$$\max_{\boldsymbol{s}} \sum_{i,j} w_{ij} \log\left(\frac{1}{1 + e^{s_j - s_i}}\right), \tag{2.12}$$

where w_{ij} is the (i, j)-th element in the winning matrix $\boldsymbol{W}$, meaning the number of times when method i is favored over method j. We use the Newton-Raphson method to solve Eq. (2.12) and set the score for ground truth method as 1 to avoid the scale ambiguity.

The experiment results are detailed in Section 2.6

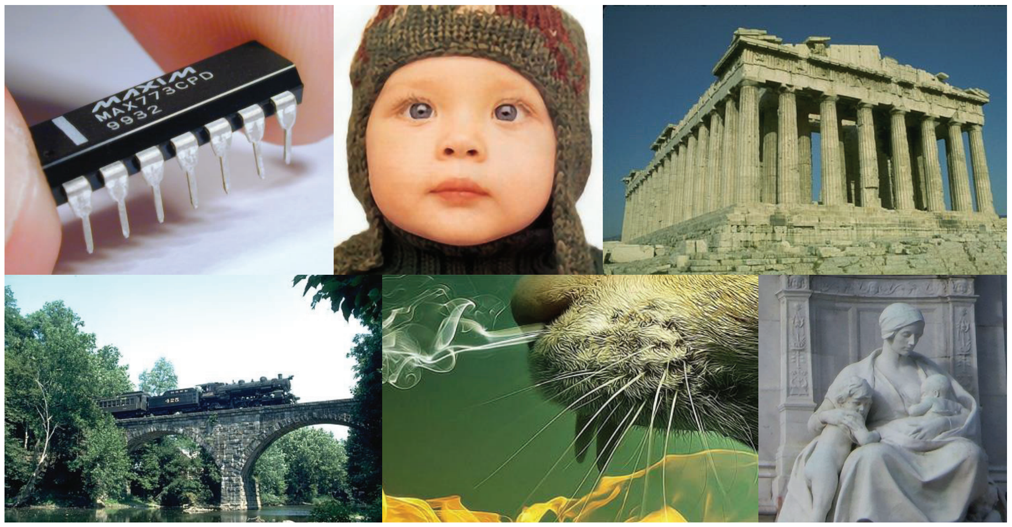


Figure 2.6: The six images used in subjective evaluation.

2.6 Experiments

We evaluate and compare the performance of our models using the same data and protocols as in [27], which are commonly adopted in SR literature. All our models are learned from a training set with 91 images, and tested on Set5 [38], Set14 [39] and BSD100 [40] which contain 5, 14 and 100 images respectively. We have also trained on other larger data sets, and observe marginal performance change (around 0.1 dB). The original images are downsized by bicubic interpolation to generate LR-HR image pairs for both training and evaluation. The training data are augmented with translation, rotation and scaling.

2.6.1 Implementation Details

We determine the number of nodes in each layer of our SCN mainly according to the corresponding settings used in sparse coding [8]. Unless otherwise stated, we use input LR patch size s_y=9, LR feature dimension m_y=100, dictionary size n=128, output HR patch size s_x=5, and patch aggregation filter size s_g=5. All the convolution layers have a stride of 1. Each LR patch $\boldsymbol{y}$ is normalized by its mean and variance, and the same mean and variance are used to restore the final HR patch $\boldsymbol{x}$. We crop 56×56 regions from each image to obtain fixed-sized input samples to the network, which produces outputs of size 44×44.

To reduce the number of parameters, we implement the LR patch extraction layer $\boldsymbol{H}$ as the combination of two layers: the first layer has 4 trainable filters each of which is shifted to 25 fixed positions by the second layer. Similarly, the patch combination layer $\boldsymbol{G}$ is also split into a fixed layer which aligns pixels in overlapping patches and a trainable layer whose weights are used to combine overlapping pixels. In this way, the number of parameters in these two layers is reduced by more than an order, and there is no observable loss in performance.

We employ a standard stochastic gradient descent algorithm to train our networks with mini-batch size of 64. Based on the understanding of each layer's role in sparse coding, we use Harr-like gradient filters to initialize layer $\boldsymbol{H}$, and use uniform weights to initialize layer $\boldsymbol{G}$. All the remaining three linear layers are related to the dictionary pair $(\boldsymbol{D}_x, \boldsymbol{D}_y)$ in sparse coding. To initialize them, we first randomly set $\boldsymbol{D}_x$ and $\boldsymbol{D}_y$ with Gaussian noise, and then find the corresponding layer weights as in ISTA [23]:

$$\boldsymbol{w}_1 = C \cdot \boldsymbol{D}_y^T,\ \boldsymbol{w}_2 = \boldsymbol{I} - \boldsymbol{D}_y^T \boldsymbol{D}_y,\ \boldsymbol{w}_3 = (CL)^{-1} \cdot \boldsymbol{D}_x, \tag{2.13}$$

where $\boldsymbol{w}_1$, $\boldsymbol{w}_2$ and $\boldsymbol{w}_3$ denote the weights of the three subsequent layers after layer $\boldsymbol{H}$. L is the upper bound on the largest eigenvalue of $\boldsymbol{D}_y^T \boldsymbol{D}_y$, and C is the threshold value before normalization. We empirically set L=C=5.

The proposed models are all trained using the CUDA ConvNet package [15] on a workstation with 12 Intel Xeon 2.67GHz CPUs and 1 GTX680 GPU. Training a SCN usually takes less than one day. Note that this package is customized for classification networks, and its efficiency can be further optimized for our SCN model.

In testing, to make the entire image covered by output samples, we crop input samples with overlap and extend the boundary of original image by reflection. Note we shave the image border in the same way as [18] for objective evaluations to ensure fair comparison. Only the luminance channel is processed with our method, and bicubic interpolation is applied to the chrominance channels, as their high frequency components are less noticeable to human eyes. To achieve arbitrary scaling factors using CSCN, we upscale an image by ×2 times repeatedly until it is at least as large as the desired size. Then a bicubic interpolation is used to downscale it to the target resolution

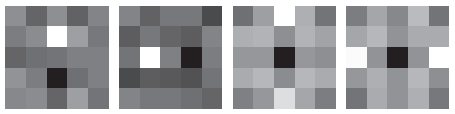


Figure 2.7: The four learned filters in the first layer $\boldsymbol{H}$.

Table 2.1: Time consumption for SCN to upscale the "baby" image from 256×256 to 512×512 using different dictionary size n.

n	64	96	128	256	512
time (s)	0.159	0.192	0.230	0.445	1.214

if necessary.

When reporting our best results in Section 2.6.3, we also use the multi-view testing strategy commonly employed in image classification. For patch-based image SR, multi-view testing is implicitly used when predictions from multiple overlapping patches are averaged. Here, besides sampling overlapping patches, we also add more views by flipping and transposing the patch. Such strategy is found to improve SR performance for general algorithms at the sheer cost of computation.

2.6.2 Algorithm Analysis

We first visualize the four filters learned in the first layer $\boldsymbol{H}$ in Figure 2.7. The filter patterns do not change much from the initial first and second order gradient operators. Some additional small coefficients are introduced in a highly structured form so that the filters can capture richer high frequency details.

The performance of several networks during training is measured on Set5 in Figure 2.8. Our SCN improves significantly over sparse coding (SC) [8], as it leverages data more effectively with end-to-end training. The SCN initialized according to (2.13) can converge faster and better than the same model with random initialization, which indicates that the understanding of SCN based on sparse coding can help its optimization. We also train a CNN model [18] of the same size as SCN, but find its convergence speed much slower. It is reported in [18] that training a CNN takes 8×10^8 back-propagations (equivalent to 12.5×10^6 mini-batches here). To achieve the same performance as CNN, our SCN requires less than 1% back-propagations.

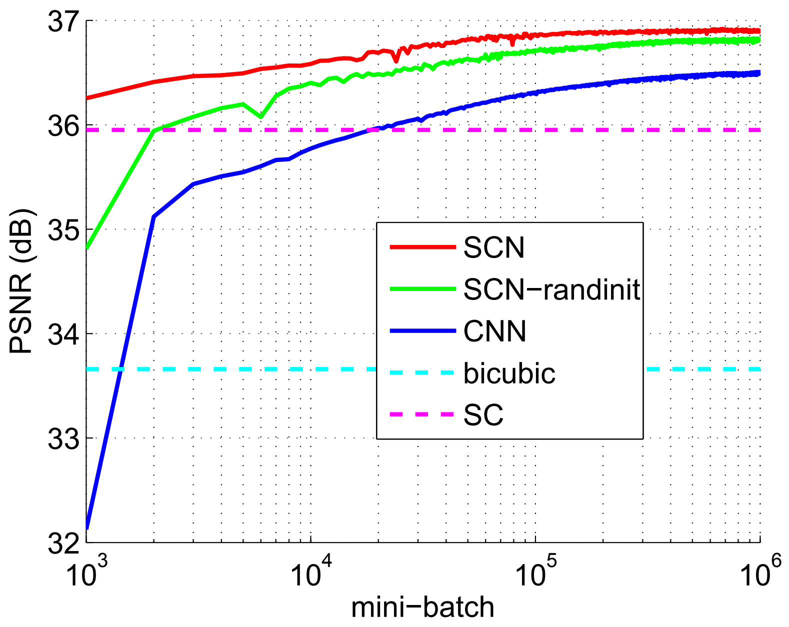


Figure 2.8: The PSNR change for ×2 SR on Set5 during training using different methods: SCN; SCN with random initialization; CNN. The horizontal dash lines show the benchmarks of bicubic interpolation and sparse coding (SC).

The network size of SCN is mainly determined by the dictionary size n. Besides the default value n=128, we have tried other sizes and plot their performance versus the number of network parameters in Figure 2.9. The PSNR of SCN does not drop too much as n decreases from 128 to 64, but the model size and computation time can be reduced significantly, as shown in Table 2.1. Figure 2.9 also shows the performance of CNN with various sizes. Our smallest SCN can achieve higher PSNR than the largest model (CNN-L) in [41] while only using about 20% parameters.

Different numbers of recurrent stages k have been tested for SCN, and we find increasing k from 1 to 3 only improves performance by less than 0.1 dB. As a tradeoff between speed and accuracy, we use k=1 throughout this section.

In Table 2.2, different network structures with cascade for scalable SR in Section 2.3.3 (in each row) are compared at different scaling factors (in each column). SCN×a denotes the model trained with fixed scaling factor a without any cascade technique. For a fixed a, we use SCN×a as a basic module and apply it one or more times to super-resolve images for different upscaling factors, as shown in Table 2.2. It is observed that SCN×2 can perform as

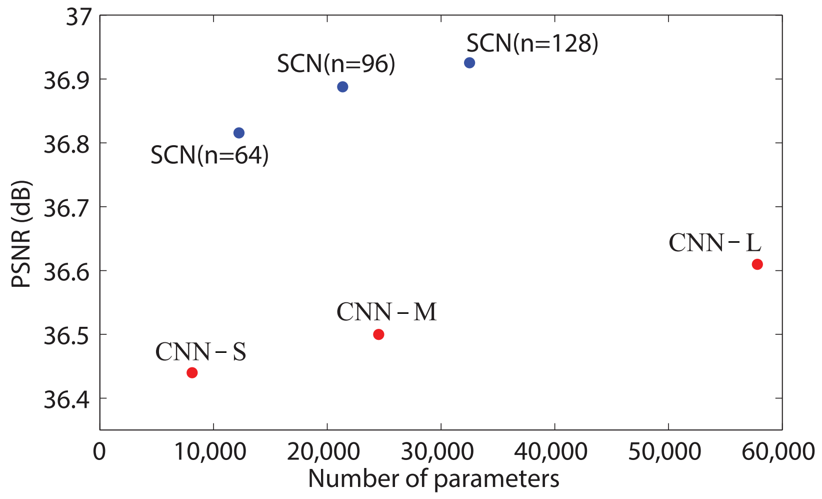


Figure 2.9: PSNR for ×2 SR on Set5 using SCN and CNN with various network sizes.

Table 2.2: PSNR of different network cascading schemes on Set5, evaluated for different scaling factors in each column.

scaling factor	×1.5	×2	×3	×4
SCN×1.5	40.14	36.41	30.33	29.02
SCN×2	**40.15**	**36.93**	32.99	30.70
SCN×3	39.88	36.76	32.87	30.63
SCN×4	39.69	36.54	32.76	30.55
CSCN	**40.15**	**36.93**	**33.10**	**30.86**

well as the scale-specific model for small scaling factor (1.5), and much better for large scaling factors (3 and 4). Note that the cascade of SCN×1.5 does not lead to good results since artifacts quickly get amplified through many repetitive upscalings. Therefore, we use SCN×2 as the default building block for CSCN, and drop the notation ×2 when there is no ambiguity. The last row in Table 2.2 shows that a CSCN trained using the multi-scale objective in (2.7) can further improve the SR results for scaling factors 3 and 4, as the second SCN in the cascade is trained to be robust to the artifacts generated by the first one.

As shown in [41], the amount of training data plays an important role in the field of deep learning. In order to evaluate the effect of various amounts of data on training CSCN, we change the training set from a relatively small set of 91 images (Set91) [27] to two other sets: the 199 out of 200 training

Table 2.3: Effect of various training sets on the PSNR of ×2 upscaling with single view SCN.

Training Set	Test Set			
	Set5	Set14	BSD100	ILSVRC (100)
Set91	36.93	32.56	31.40	32.13
BSD200	**36.97**	**32.69**	**31.55**	32.27
ILSVRC (7.5k)	36.84	32.67	31.51	**32.31**

Table 2.4: PSNR (SSIM) comparison on three test data sets among different methods. Red indicates the best and blue indicates the second best performance. The performance gain of our best model over all the others' best is shown in the last row.

Data Set	Set5			Set14			BSD100		
Upscaling	×2	×3	×4	×2	×3	×4	×2	×3	×4
A+ [42]	36.55 (0.9544)	32.59 (0.9088)	30.29 (0.8603)	32.28 (0.9056)	29.13 (0.8188)	27.33 (0.7491)	30.78 (0.8773)	28.18 (0.7808)	26.77 (0.7085)
CNN [18]	36.34 (0.9521)	32.39 (0.9033)	30.09 (0.8530)	32.18 (0.9039)	29.00 (0.8145)	27.20 (0.7413)	31.11 (0.8835)	28.20 (0.7794)	26.70 (0.7018)
CNN-L [41]	36.66 (0.9542)	32.75 (0.9090)	30.49 (0.8628)	32.45 (0.9067)	29.30 (0.8215)	27.50 (0.7513)	31.36 (0.8879)	28.41 (0.7863)	26.90 (0.7103)
CSCN	37.00 (0.9557)	33.18 (0.9153)	30.94 (0.8755)	32.65 (0.9081)	29.41 (0.8234)	27.71 (0.7592)	31.46 (0.8891)	28.52 (0.7883)	27.06 (0.7167)
CSCN-MV	37.21 (0.9571)	33.34 (0.9173)	31.14 (0.8789)	32.80 (0.9101)	29.57 (0.8263)	27.81 (0.7619)	31.60 (0.8915)	28.60 (0.7905)	27.14 (0.7191)
Our Improvement	0.55 (0.0029)	0.59 (0.0083)	0.65 (0.0161)	0.35 (0.0034)	0.27 (0.0048)	0.31 (0.0106)	0.24 (0.0036)	0.19 (0.0042)	0.24 (0.0088)

images [2] in BSD500 dataset (BSD200) [40], and a subset of 7,500 images from the ILSVRC2013 dataset [43]. A model of exactly the same architecture without any cascade is trained on each data set, and another 100 images from the ILSVRC2013 dataset are included as an additional test set. From Table 2.3, we can observe that the CSCN trained on BSD200 consistently outperforms its counterpart trained on Set91 by around 0.1 dB on all test data. However, the performance of the model trained on ILSVRC2013 is slightly different from the one trained on BSD200, which shows the saturation of the performance as the amount of training data increases. The inferior quality of images in ILSVRC2013 may be a hurdle to further improve the performance. Therefore, our method is robust to training data and can benefit marginally from a larger set of training images.

[2] Since one out of 200 training images coincides with one image in Set5, we exclude it from our training set.

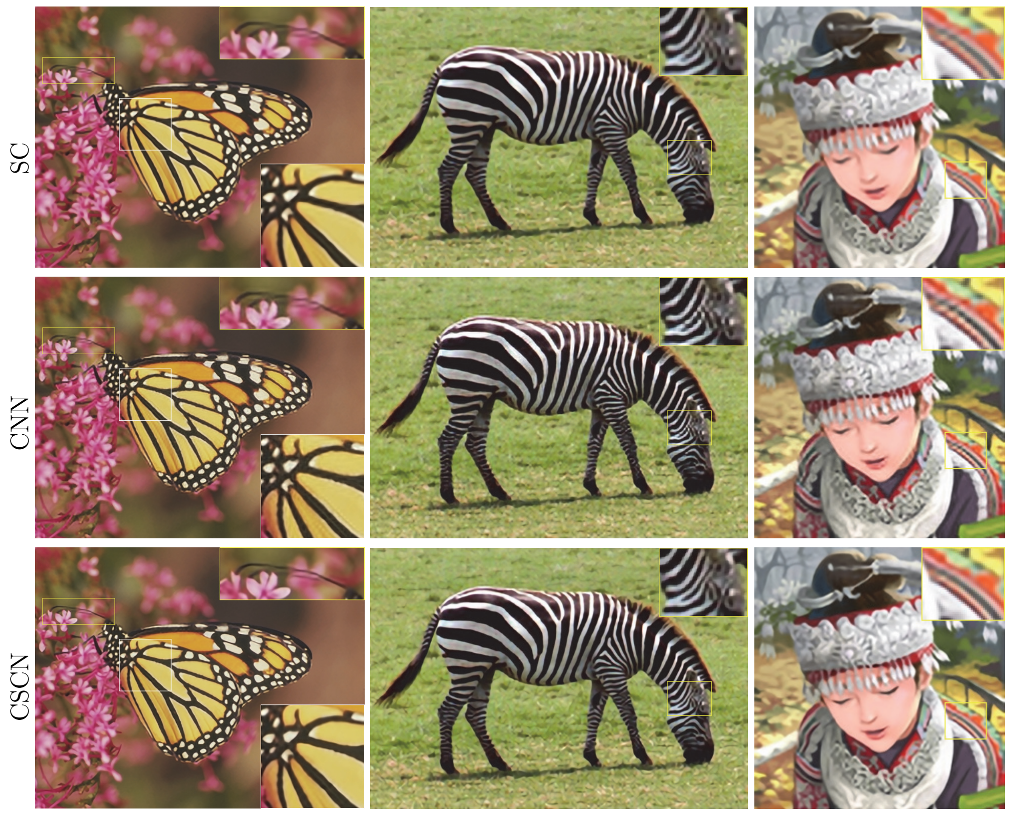
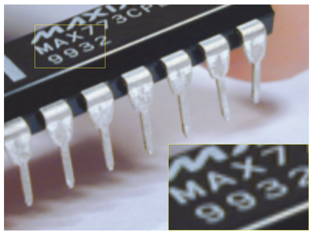


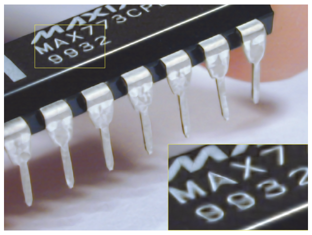
Figure 2.10: SR results given by SC [8] (first row), CNN [18] (second row) and our CSCN (third row). Images from left to right: the "monarch" image upscaled by ×3; the "zebra" image upscaled by ×3; the "comic" image upscaled by ×3.

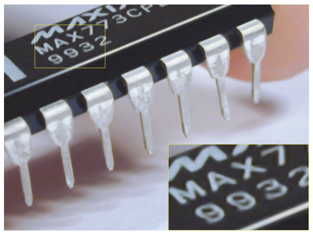
2.6.3 Comparison with State of the Art

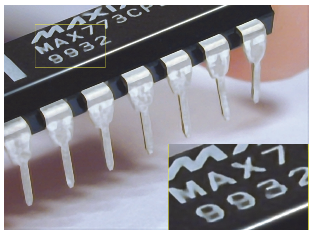
We compare the proposed CSCN with other recent SR methods on all the images in Set5, Set14 and BSD100 for different scaling factors. Table 2.4 shows the PSNR and structural similarity (SSIM) [44] for adjusted anchored neighborhood regression (A+) [42], CNN [18], CNN trained with larger model size and much more data (CNN-L) [41], the proposed CSCN, and CSCN with our multi-view testing (CSCN-MV). We do not list other methods [8, 27, 39, 12, 45] whose performance is worse than A+ or CNN-L.

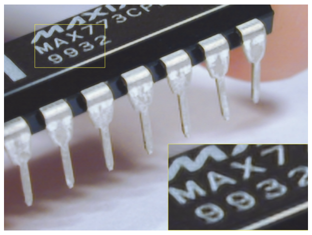
It can be seen from Table 2.4 that CSCN performs consistently better than all previous methods in both PSNR and SSIM, and with multi-view testing the results can be further improved. CNN-L improves over CNN by increasing model parameters and training data. However, it is still not as good as CSCN which is trained with a much smaller model size and on a much


(a) bicubic


(b) SE [11]


(c) SC [8]


(d) DNC [16]


(e) CNN [18]

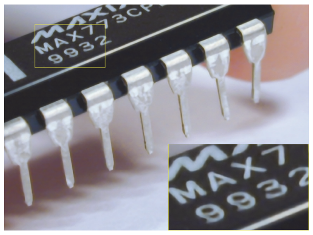

(f) CSCN

Figure 2.11: The "chip" image upscaled by $\times 4$ times using different methods.

smaller data set. Clearly, the better model structure of CSCN makes it less dependent on model capacity and training data in improving performance. Our models are generally more advantageous for large scaling factors due to the cascade structure. A larger performance gain is seen on Set5 than the other two test sets because Set5 has more similar statistics as the training set.

The visual qualities of the SR results generated by sparse coding (SC) [8], CNN and CSCN are compared in Figure 2.10. Our approach produces image patterns with shaper boundaries and richer textures, and is free of the ringing artifacts observable in the other two methods.

Figure 2.11 shows the SR results on the "chip" image compared among more methods including the self-example based method (SE) [11] and the deep network cascade (DNC) [16]. SE and DNC can generate very sharp edges on this image, but also introduce artifacts and blurs on corners and fine structures due to the lack of self-similar patches. In contrast, the CSCN method recovers all the structures of the characters without any distortion.

Table 2.5: PSNR of ×3 upscaling on LR images with different blurring kernels.

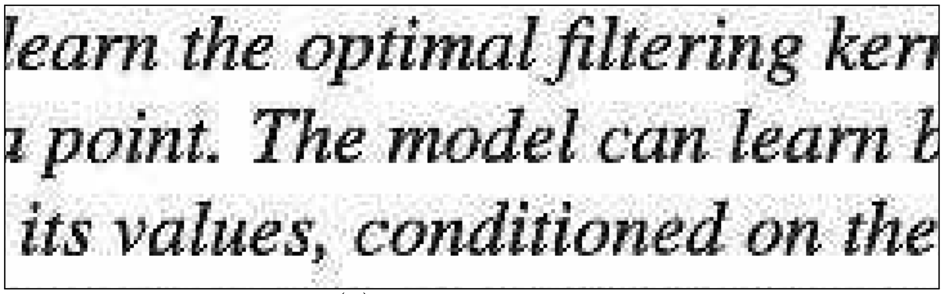
Kernel	Gaussian $\sigma = 1.0$			Gaussian $\sigma = 1.6$		
Method	CSR [46]	NLM [47]	SCN	CSR [46]	GSC [48]	SCN
Butterfly	27.87	26.93	**28.70**	28.19	25.48	**29.03**
Parrots	30.17	29.93	**30.75**	30.68	29.20	**30.83**
Parthenon	26.89	–	**27.06**	27.23	26.44	**27.40**
Bike	24.41	24.38	**24.81**	24.72	23.78	**25.11**
Flower	29.14	28.86	**29.50**	29.54	28.30	**29.78**
Girl	**33.59**	33.44	33.57	**33.68**	33.13	33.65
Hat	31.09	30.81	**31.32**	31.33	30.29	**31.62**
Leaves	26.99	26.47	**27.45**	27.60	24.78	**27.87**
Plants	33.92	33.27	**34.35**	34.00	32.33	**34.53**
Raccoon	**29.09**	–	28.99	**29.29**	28.81	29.16
Average	29.32	29.26	**29.65**	29.63	28.25	**29.90**

2.6.4 Robustness to Real SR Scenarios

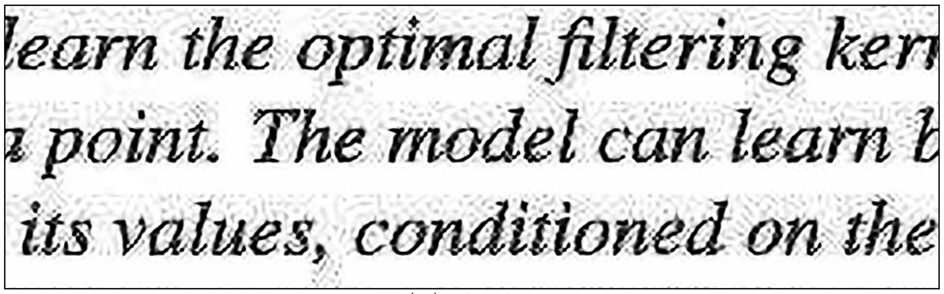
We evaluate the performance of the proposed practical SR methods in Section 2.4 by providing the empirical results of several experiments for the two aforementioned approaches.

Data-driven SR by Fine-tuning

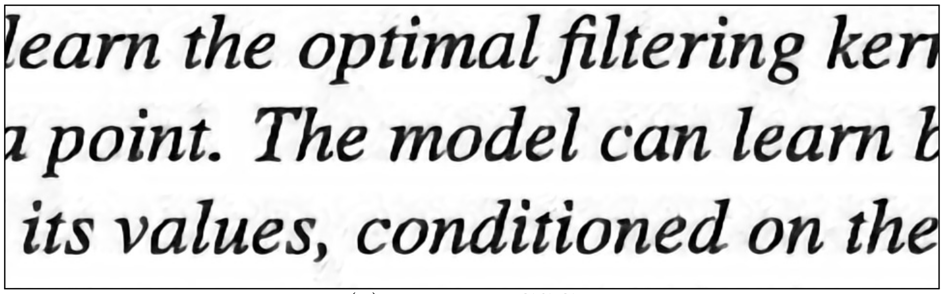
The method proposed in Section 2.4.1 is data-driven, and thus the generic SCN can be easily adapted for a particular task, with a small number of training samples. We demonstrate the performance of this method in the application of enlarging low-DPI scanned document images with heavy noise. We first obtain several pairs of LR and HR images by scanning a document under two settings of 150DPI and 300DPI. Then we fine-tune our generic CSCN model using only one pair of scanned images for a few iterations. Figure 2.12 illustrates the visualization of the upscaled image from the 150DPI scanned image. As shown by the SR results in Figure 2.12, the CSCN before adaptation is very sensitive to LR measurement corruption, so the enlarged texts in (b) are much more corrupted than they are in the nearest neighbor upscaled image (a). However, the adapted CSCN model removes almost all the artifacts and can restore clear texts in (c), which is promising for practical applications such as quality enhancement of online scanned books and restoration of legacy documents.


learn the optimal filtering ker
a point. The model can learn b
its values, conditioned on the

(a) nearest neighbor


learn the optimal filtering ker
a point. The model can learn b
its values, conditioned on the

(b) CSCN


learn the optimal filtering ker
a point. The model can learn b
its values, conditioned on the

(c) adapted CSCN

Figure 2.12: Low-DPI scanned document upscaled by ×4 times using different methods.

Regularized Iterative SR

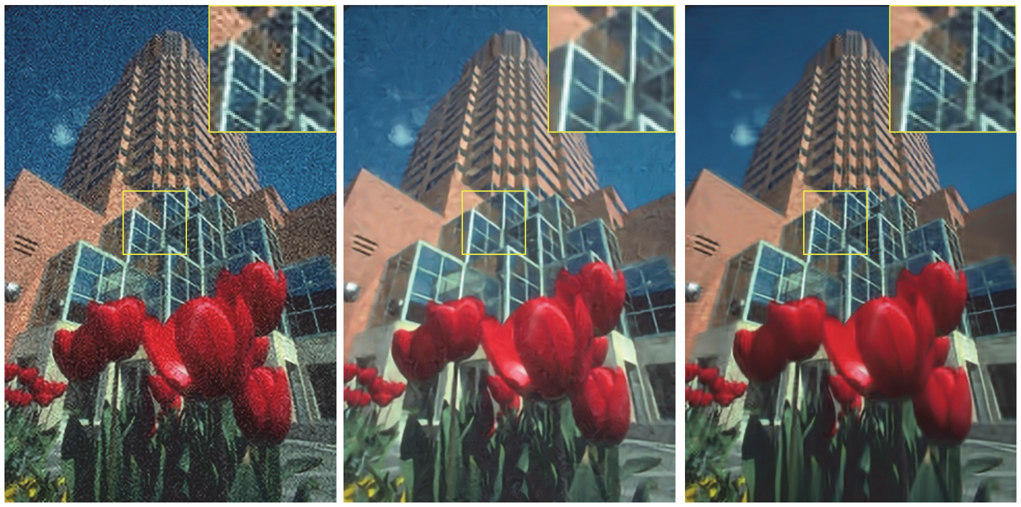
We now show experimental results of practical SR for blurred and noisy LR images, using the proposed regularized iterative methods in Section 2.4.2. We first compare the SR performance on blurry images using the method proposed in Section 2.4.2 with several other recent methods [48, 46, 47], using the same test images and settings. All these methods are designed for blurry LR input, while our model is trained on sharp LR input. As shown in Table 2.5, our model achieves much better results than the competitors. Note that the inference time of our model is much less than that of the conventional sparse coding based methods.

Table 2.6: PSNR values for $\times 2$ upscaling noisy LR images in Set5 by directly using SCN (Direct SCN), directly using CNN-L (Direct CNN-L), SCN after fine-tuning on new noisy training data (Fine-tuning SCN), the iterative method of BM3D & SCN (Iterative BM3D-SCN), and the iterative method of BM3D & CNN-L (Iterative BM3D-CNN-L).

σ	5	10	15	20
Direct SCN	30.23	25.11	21.81	19.45
Direct CNN-L	30.47	25.32	21.91	19.46
Fine-tuning SCN	33.03	31.00	29.46	28.44
Iterative BM3D-SCN	**33.51**	**31.22**	**29.65**	**28.61**
Iterative BM3D-CNN-L	33.42	31.16	29.62	28.59

To test the performance of upscaling noisy LR images, we simulate additive Gaussian noise for the LR input images at four different noise levels (σ = 5, 10, 15, 20) as the noisy input images. We compare the practical SR results in Set5 obtained from the following algorithms: directly using SCN, our proposed iterative SCN method using BM3D as denoising regularizer (iterative BM3D-SCN), and fine-tuning SCN with additional noisy training pairs. Note that knowing the underlying corruption model of real LR image (e.g., noise distribution or blurring kernel), one can always synthesizes real training pairs for fine-tuning the generic SCN. In other words, once the iterative SR method is feasible, one can always apply our proposed data-driven method for SR alternatively. However, the other way around is not true. Therefore, the knowledge of the corruption model of real measurements can be considered as a stronger assumption, compared to providing real training image pairs. Correspondingly, the SR performances of these two methods are evaluated when both can be applied. We also provide the results of methods directly using another generic SR model: CNN-L [41], and the similar iterative SR method involving CNN-L (iterative BM3D-CNN-L).

The practical SR results are listed in Table 2.6. We observed the improved PSNR using our proposed regularized iterative SR method over all noise levels. The proposed iterative BM3D-SCN achieves much higher PSNR than the method of directly using SCN. The performance gap (in terms of SR PSNR) between iterative BM3D-SCN and direct SCN becomes larger as the noise level increases. A similar result can be observed in the comparison of iterative BM3D-CNN-L and direct CNN-L. Compared to the method of fine-tuning SCN, the iterative BM3D-SCN method demonstrates better empirical


(a) direct SCN
PSNR=24.00 dB

(b) fine-tuning SCN
PSNR=27.54 dB

(c) iterative BM3D-SCN
PSNR=27.86 dB

Figure 2.13: The “building” image corrupted by additive Gaussian noise of $\sigma = 10$ and then upscaled by $\times 2$ times using different methods.

performance, with 0.3 dB improvement on average. The iterative BM3D-CNN-L method provides results comparable to those of the iterative BM3D-SCN method, which demonstrates that our proposed regularized iterative SCN scheme can be easily extended for other SR methods, and is able to effectively handle noisy LR measurements.

An example of upscaling noisy LR images using the aforementioned methods is demonstrated in Figure 2.13. Both find-tuning SCN and iterative BM3D-SCN are able to significantly suppress the additive noise, while many artifacts induced by noise are observed in the SR result of direct SCN. It is notable that the fine-tuning SCN method performs better at recovering the texture and the iterative BM3D-SCN method is preferable in smooth regions.

2.6.5 Subjective Evaluation

We have a total of 270 participants giving 720 pairwise comparisons over six images with different scaling factors, which are shown in Figure 2.6. Not every participant completed all the comparisons but their partial responses are still useful.

Figure 2.14 shows the estimated scores for the six SR methods in our evaluation, with the score for ground truth method normalized to one. As

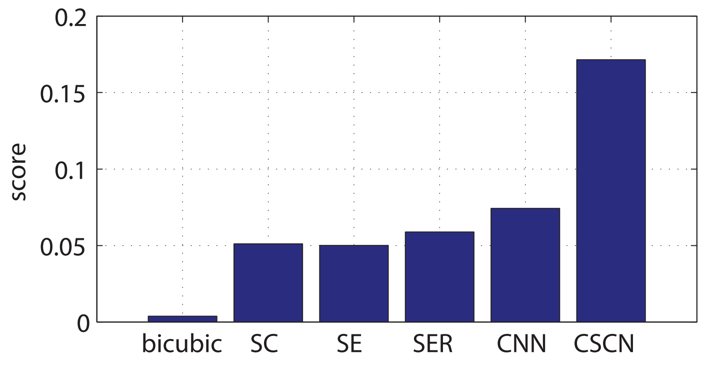


Figure 2.14: Subjective SR quality scores for different methods including bicubic, SC [8], SE [11], SER [36], CNN [18] and the proposed CSCN. The score for ground truth result is one.

expected, all the SR methods have much lower scores than ground truth, showing the great challenge in the SR problem. The bicubic interpolation is significantly worse than other SR methods. The CSCN method outperforms other previous state-of-the-art methods by a large margin, demonstrating its superior visual quality. It should be noted that the visual difference between some image pairs is very subtle. Nevertheless, the human subjects are able to perceive such differences when seeing the two images side by side, and therefore make consistent ratings. The CNN model becomes less competitive in the subjective evaluation than it is in PSNR comparison. This indicates that the visually appealing image appearance produced by CSCN should be attributed to the regularization from sparse representation, which cannot be easily learned by merely minimizing reconstruction error as in CNN.

2.7 Conclusion

We propose a new model for image SR by combining the strengths of sparse coding and deep network, and make considerable improvement over existing deep and shallow SR models both quantitatively and qualitatively. Besides producing good SR results, the domain knowledge in the form of sparse coding can also benefit training speed and model compactness. Furthermore, we investigate the cascade of networks to enhance SR performance. In addition, the robustness to real SR scenarios is discussed for handling non-ideal LR measurements. More generally, our observation is in line with other recent extensions made to CNN with better domain knowledge for different tasks.

Chapter 3

LEARNING A MIXTURE OF DEEP NETWORKS FOR SINGLE IMAGE SUPER-RESOLUTION

3.1 Introduction

Single image super-resolution (SR) is usually cast as an inverse problem of recovering the original high-resolution (HR) image from the low-resolution (LR) observation image. This technique can be utilized in the applications where high resolution is important, such as photo enhancement, satellite imaging and SDTV to HDTV conversion [49]. The main difficulty resides in the loss of much information in the degradation process. Since the known variables from the LR image are usually greatly outnumbered by those from the HR image, this is a highly ill-posed problem.

A large number of single image SR methods have been proposed in the literature, including interpolation based method [50], edge model based method [5] and example based method [51, 10, 7, 27, 41, 45]. Since the former two methods usually suffer the sharp drop in restoration performance with large upscaling factors, the example based method has drawn great attention from the community recently. It usually learns the mapping from LR images to HR images in a patch-by-patch manner, with the help of sparse representation [7, 35], random forest [52] and so on. The neighbor embedding method [51, 27] and neural network based method [41] are two representatives of this category.

Neighbor embedding is proposed in [51, 38] which estimates HR patches as a weighted average of local neighbors with the same weights as in LR feature space, based on the assumption that LR/HR patch pairs share similar local geometry in low-dimensional nonlinear manifolds. The coding coefficients are first acquired by representing each LR patch as a weighted average of local neighbors, and then the HR counterpart is estimated by the multiplication of the coding coefficients with the corresponding training HR patches. Anchored neighborhood regression (ANR), which partitions the feature space into a number of clusters using the learned dictionary atoms as a set of anchor points, is utilized in [27] to improve the neighbor embedding methods. A regressor is then learned for each cluster of patches. This approach has demonstrated superiority over the counterpart of global regression in [27]. Other variants of learning a mixture of SR regressors can be found in [42, 53, 54].

Recently, neural network based models have demonstrated the strong capability for single image SR [16, 41, 55], due to its large model capacity and the end-to-end learning strategy to get rid of hand-crafted features. Cui et al. [16] propose using a cascade of stacked collaborative local autoencoders for robust matching of self-similar patches, in order to increase the resolution of inputs gradually. Dong et al. [41] exploit a fully convolutional neural network (CNN) to approximate the complex non-linear mapping between the LR image and the HR counterpart. A neural network that closely mimics the sparse coding approach for image SR is designed by Wang et al. [55, 56]. Kim et al. propose a very deep neural network with residual architecture to exploit contextual information over large image regions [57].

In this chapter, we propose a method to combine the merits of the neighborhood embedding methods and the neural network based methods via learning a mixture of neural networks for single image SR. The entire image signal space can be partitioned into several subspaces, and we dedicate one SR module to the image signals in each subspace, the synergy of which allows a better capture of the complex relation between the LR image signal and its HR counterpart than the generic model. In order to take advantage of the end-to-end learning strategy of neural network based methods, we choose neural networks as the SR inference modules and incorporate these modules into one unified network, and design a branch in the network to predict the pixel-level weights for HR estimates from each SR module before they are adaptively aggregated to form the final HR image.

A systematic analysis of different network structures is conducted with the focus on the relation between SR performance and various network structures via extensive experiments, where the benefit of utilizing a mixture of SR models is demonstrated. Our proposed approach is contrasted with other current popular approaches on a large number of test images, and achieves state-of-the-art performance consistently along with more flexibility of model

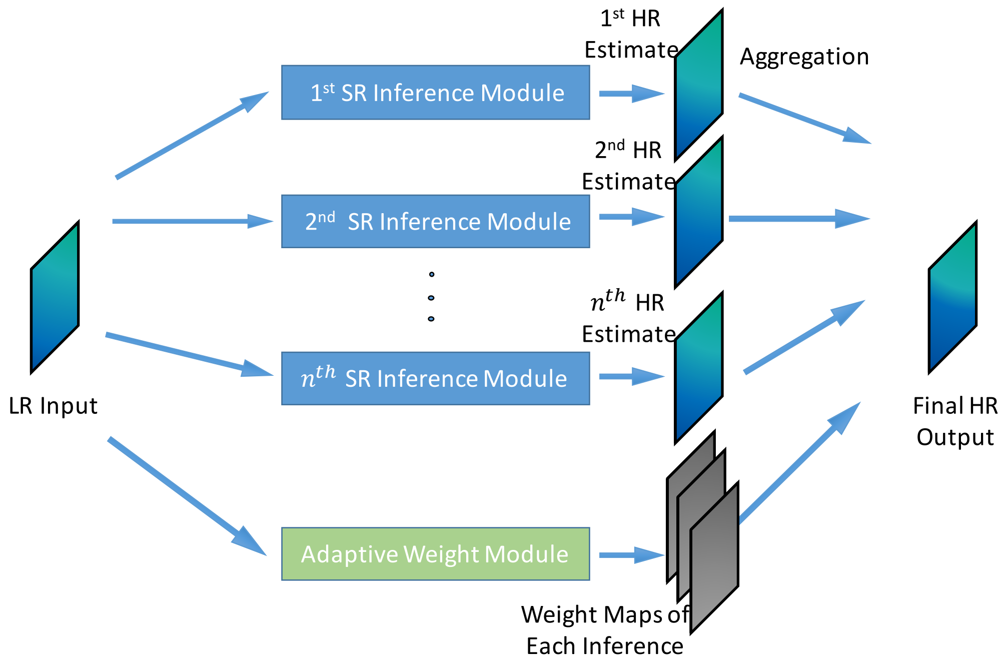


Figure 3.1: The overview of our proposed method. It consists of a number of SR inference modules and an adaptive weight module. Each SR inference module is dedicated to inferencing a certain class of image local patterns, and is independently applied on the LR image to predict one HR estimate. These estimates are adaptively combined using pixel-wise aggregation weights from the adaptive weight module in order to form the final HR image.

design choices.

This section is organized as follows. The proposed method is introduced and explained in detail in Section 3.2. Section 3.3 describes our experiments, in which we analyze thoroughly different network structures and compare the performance of our method with other current SR methods both quantitatively and qualitatively. Finally in Section 3.4 we conclude this section.

3.2 Proposed Method

3.2.1 Overview

First we give the overview of our method. The LR image serves as the input to our method. There are a number of **SR inference modules** $\{B_i\}_{i=1}^N$ in

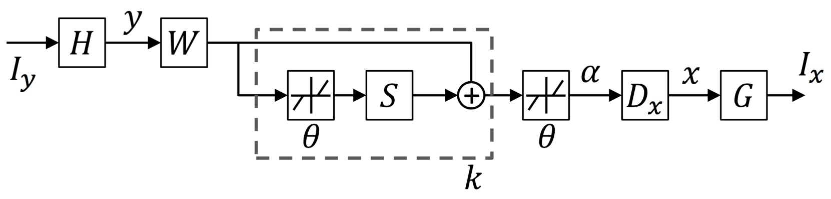


Figure 3.2: The network architecture of SCN [55], which serves as the SR inference module in our method.

our method. Each of them, B_i, is dedicated to inferencing a certain class of image patches, and applied on the LR input image to predict a HR estimate. We also devise an **adaptive weight module**, T, to adaptively combine at the pixel-level the HR estimates from SR inference modules. When we select neural networks as the SR inference modules, all the components can be incorporated into a unified neural network and be jointly learned. The final estimated HR image is adaptively aggregated from the estimates of all SR inference modules. The overview of our method is shown in Figure 3.1.

3.2.2 Network Architecture

SR Inference Module

Taking the LR image as input, each SR inference module is designed to better capture the complex relation between a certain class of LR image signals and its HR counterpart, while predicting a HR estimate. For the sake of inference accuracy, we choose as the SR inference module a recent sparse coding based network (SCN) in [55], which implicitly incorporates the sparse prior into neural networks via employing the learned iterative shrinkage and thresholding algorithm (LISTA), and closely mimics the sparse coding based image SR method [8]. The architecture of SCN is shown in Figure 3.2. Note that the design of the SR inference module is not limited to SCN, and all other neural network based SR models, e.g. SRCNN [41], can work as the SR inference module as well. The output of B_i serves as an estimate of the final HR frame.

Adaptive Weight Module

The goal of this module is to model the selectivity of the HR estimates from every SR inference module. We propose assigning pixel-wise aggregation weights of each HR estimate, and again the design of this module is open to any operation in the field of neural networks. Taking into account the computation cost and efficiency, we utilize only three convolutional layers for this module, and ReLU is applied on the filter responses to introduce non-linearity. This module finally outputs the pixel-level weight maps for all the HR estimates.

Aggregation

Each SR inference module's output is pixel-wisely multiplied with its corresponding weight map from the adaptive weight module, and then these products are summed to form the final estimated HR frame. If we use $\boldsymbol{y}$ to denote the LR input image, a function $W(\boldsymbol{y}; \theta_w)$ with parameters θ_w to represent the behavior of the adaptive weight module, and a function $F_{B_i}(\boldsymbol{y}; \theta_{B_i})$ with parameters θ_{B_i} to represent the output of SR inference module B_i, the final estimated HR image $F(\boldsymbol{y}; \boldsymbol{\Theta})$ can be expressed as

$$F(\boldsymbol{y}; \boldsymbol{\Theta}) = \sum_{i=1}^{N} W_i(\boldsymbol{y}; \theta_w) \odot F_{B_i}(\boldsymbol{y}; \theta_{B_i}), \tag{3.1}$$

where $\odot$ denotes the point-wise multiplication.

3.2.3 Training Objective

In training, our model tries to minimize the loss between the target HR frame and the predicted output, as

$$\min_{\boldsymbol{\Theta}} \sum_{j} \|F(\boldsymbol{y}_j; \boldsymbol{\Theta}) - \boldsymbol{x}_j\|_2^2, \tag{3.2}$$

where $F(\boldsymbol{y}; \boldsymbol{\Theta})$ represents the output of our model, $\boldsymbol{x}_j$ is the j-th HR image and $\boldsymbol{y}_j$ is the corresponding LR image; $\boldsymbol{\Theta}$ is the set of all parameters in our model.

If we plug Eq. (3.1) into Eq. (3.2), the cost function then can be expanded as:

$$\min_{\theta_w,\{\theta_{B_i}\}_{i=1}^N} \sum_j \| \sum_{i=1}^N W_i(\boldsymbol{y}_j;\theta_w) \odot F_{B_i}(\boldsymbol{y}_j;\theta_{B_i}) - \boldsymbol{x}_j\|_2^2. \tag{3.3}$$

3.3 Experiments

3.3.1 Data Sets and Implementation Details

We conduct experiments following the protocols in [27]. Different learning based methods use different training data in the literature. We choose 91 images proposed in [7] to be consistent with [42, 52, 55]. These training data are augmented with translation, rotation and scaling, providing approximately 8 million training samples of 56×56 pixels.

Our model is tested on three benchmark data sets, which are Set5 [38], Set14 [39] and BSD100 [40]. The ground truth images are downscaled by bicubic interpolation to generate LR/HR image pairs for both training and testing.

Following the convention in [27, 55], we convert each color image into the YCbCr colorspace and only process the luminance channel with our model, and bicubic interpolation is applied to the chrominance channels, because the human visual system is more sensitive to details in intensity than in color.

Each SR inference module adopts the network architecture of SCN, while the filters of all three convolutional layers in the adaptive weight module have the spatial size of 5×5 and the numbers of filters of three layers are set to be $32, 16$ and N, which is the number of SR inference modules.

Our network is implemented using Caffe [58] and is trained on a machine with 12 Intel Xeon 2.67GHz CPUs and one Nvidia TITAN X GPU. For the adaptive weight module, we employ a constant learning rate of 10^{-5} and initialize the weights from Gaussian distribution, while we stick to the learning rate and the initialization method in [55] for the SR inference modules. The standard gradient descent algorithm is employed to train our network with a batch size of 64 and the momentum of 0.9.

We train our model for the upscaling factor of two. For larger upscaling factors, we adopt the model cascade technique in [55] to apply $\times 2$ models multiple times until the resulting image reaches at least the desired size. The resulting image is downsized via bicubic interpolation to the target resolution if necessary.

3.3.2 SR Performance vs. Network Architecture

In this section we investigate the relation between various numbers of SR inference modules and SR performance. For the sake of our analysis, we increase the number of inference modules as we decrease the module capacity of each of them, so that the total model capacity is approximately consistent and thus the comparison is fair. Since the chosen SR inference module, SCN [55], closely mimics the sparse coding based SR method, we can reduce the module capacity of each inference module by decreasing the embedded dictionary size n (i.e. the number of filters in SCN) for sparse representation. We compare the following cases:

- one inference module with $n = 128$, which is equivalent to the structure of SCN in [55], denoted as *SCN (n=128)*. Note that there is no need to include the adaptive weight module in this case.
- two inference modules with $n = 64$, denoted as *MSCN-2 (n=64)*.
- four inference modules with $n = 32$, denoted as *MSCN-4 (n=32)*.

The average peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) [44] are measured to quantitatively compare the SR performance of these models over Set5, Set14 and BSD100 for various upscaling factors ($\times 2$, $\times 3$, $\times 4$), and the results are displayed in Table 3.1.

It can be observed that *MSCN-2 (n=64)* usually outperforms the original SCN network, i.e. *SCN (n=128)*, and *MSCN-4 (n=32)* can achieve the best SR performance by improving the performance marginally over *MSCN-2 (n=64)*. This demonstrates the effectiveness of our approach that each SR inference model is able to super-resolve its own class of image signals better than one single generic inference model.

In order to further analyze the adaptive weight module, we select several input images, namely, *butterfly*, *zebra*, *barbara*, and visualize the four weight maps for every SR inference module in the network. Moreover, we record

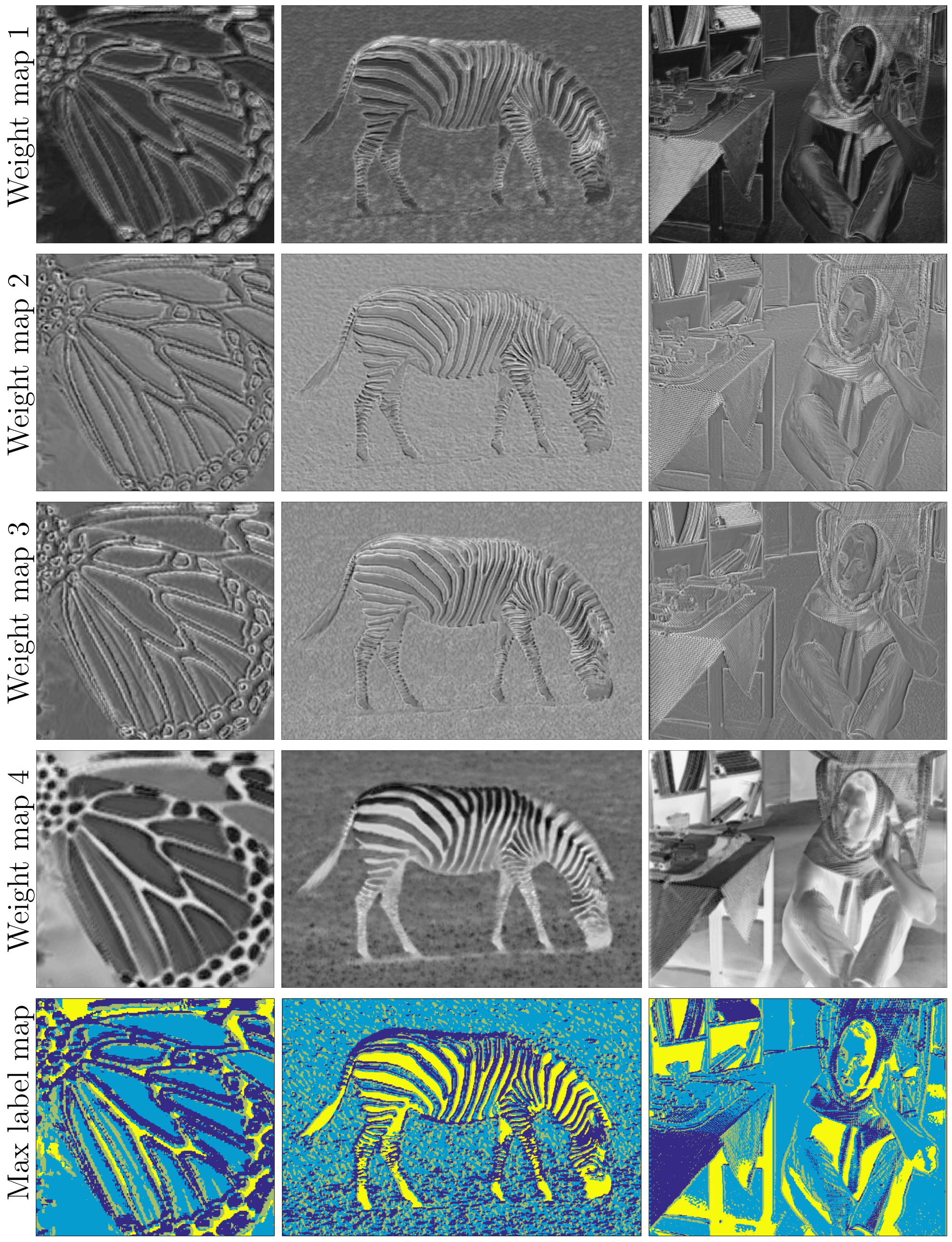


Figure 3.3: Weight maps for the HR estimate from every SR inference module in *MSCN-4* are given in the first four rows. The map (*max label map*) which records the index of the maximum weight across all weight maps at every pixel is shown in the last row. Images from left to right: the *butterfly* image upscaled by $\times 2$; the *zebra* image upscaled by $\times 2$; the *barbara* image upscaled by $\times 2$.

Table 3.1: PSNR (in dB) and SSIM comparisons on Set5, Set14 and BSD100 for ×2, ×3 and ×4 upscaling factors among various network structures. Red indicates the best and blue indicates the second best performance.

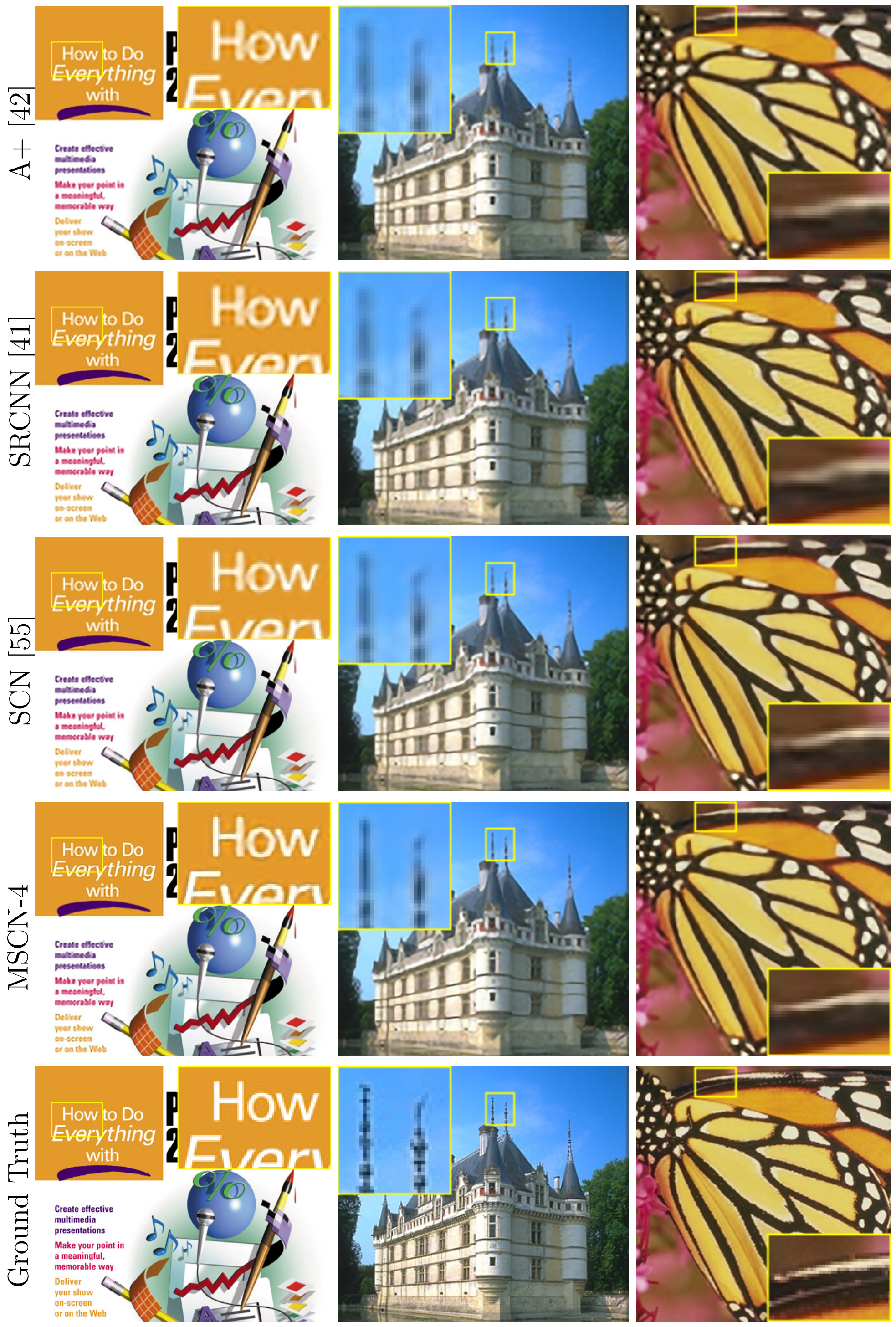
Benchmark		SCN ($n = 128$)	MSCN-2 ($n = 64$)	MSCN-4 ($n = 32$)
Set5	×2	36.93 / 0.9552	37.00 / 0.9558	36.99 / 0.9559
	×3	33.10 / 0.9136	33.15 / 0.9133	33.13 / 0.9130
	×4	30.86 / 0.8710	30.92 / 0.8709	30.93 / 0.8712
Set14	×2	32.56 / 0.9069	32.70 / 0.9074	32.72 / 0.9076
	×3	29.41 / 0.8235	29.53 / 0.8253	29.56 / 0.8256
	×4	27.64 / 0.7578	27.76 / 0.7601	27.79 / 0.7607
BSD100	×2	31.40 / 0.8884	31.54 / 0.8913	31.56 / 0.8914
	×3	28.50 / 0.7885	28.56 / 0.7920	28.59 / 0.7926
	×4	27.03 / 0.7161	27.10 / 0.7207	27.13 / 0.7216

the index of the maximum weight across all weight maps at every pixel and generate a *max label map*. These results are displayed in Figure 3.3.

From these visualizations it can be seen that weight map 4 shows high responses in many uniform regions, and thus mainly contributes to the low frequency regions of HR predictions. In contrast, weight maps 1, 2 and 3 have large responses in regions with various edges and textures, and restore the high frequency details of HR predictions. These weight maps reveal that these sub-networks work in a complementary manner for constructing the final HR predictions. In the *max label map*, similar structures and patterns of images usually share the same label, indicating that such similar textures and patterns are favored to be super-resolved by the same inference model.

3.3.3 Comparison with State-of-the-Art

We conduct experiments on all the images in Set5, Set14 and BSD100 for different upscaling factors (×2, ×3, and ×4), to quantitatively and qualitatively compare our own approach with a number of state-of-the-art image SR methods. Table 3.2 shows the PSNR and SSIM for adjusted anchored neighborhood regression (A+) [42], SRCNN [41], RFL [52], SelfEx [45] and our proposed model, *MSCN-4 (n=128)* , that consists of four SCN modules with $n = 128$. The single generic SCN without multi-view testing in [55],



(a) *ppt3* (b) *102061* (c) *butterfly*

Figure 3.4: Visual comparisons of SR results among different methods. From left to right: the *ppt3* image upscaled by $\times3$; the *102061* image upscaled by $\times3$; the *butterfly* image upscaled by $\times4$.

Table 3.2: PSNR (SSIM) comparison on three test data sets for various upscaling factors among different methods. The best performance is indicated in red and the second best performance is shown in blue. The performance gain of our best model over all the other models' best is shown in the last row.

Data Set	Set5			Set14			BSD100		
Upscaling	×2	×3	×4	×2	×3	×4	×2	×3	×4
A+ [42]	36.55 (0.9544)	32.59 (0.9088)	30.29 (0.8603)	32.28 (0.9056)	29.13 (0.8188)	27.33 (0.7491)	31.21 (0.8863)	28.29 (0.7835)	26.82 (0.7087)
SRCNN [41]	36.66 (0.9542)	32.75 (0.9090)	30.49 (0.8628)	32.45 (0.9067)	29.30 (0.8215)	27.50 (0.7513)	31.36 (0.8879)	28.41 (0.7863)	26.90 (0.7103)
RFL [52]	36.54 (0.9537)	32.43 (0.9057)	30.14 (0.8548)	32.26 (0.9040)	29.05 (0.8164)	27.24 (0.7451)	31.16 (0.8840)	28.22 (0.7806)	26.75 (0.7054)
SelfEx [45]	36.49 (0.9537)	32.58 (0.9093)	30.31 (0.8619)	32.22 (0.9034)	29.16 (0.8196)	27.40 (0.7518)	31.18 (0.8855)	28.29 (0.7840)	26.84 (0.7106)
SCN [55]	36.93 (0.9552)	33.10 (0.9144)	30.86 (0.8732)	32.56 (0.9074)	29.41 (0.8238)	27.64 (0.7578)	31.40 (0.8884)	28.50 (0.7885)	27.03 (0.7161)
MSCN-4	37.16 (0.9565)	33.33 (0.9155)	31.08 (0.8740)	32.85 (0.9084)	29.65 (0.8272)	27.87 (0.7624)	31.65 (0.8928)	28.66 (0.7941)	27.19 (0.7229)
Our Improvement	0.23 (0.0013)	0.23 (0.0011)	0.22 (0.0008)	0.29 (0.0010)	0.24 (0.0034)	0.23 (0.0046)	0.25 (0.0044)	0.16 (0.0056)	0.16 (0.0068)

i.e. *SCN (n=128)*, is also included for comparison as the baseline. Note that all the methods use the same 91 images [7] for training except SRCNN [41], which uses 395,909 images from ImageNet as training data.

It can be observed that our proposed model achieves the best SR performance consistently over three data sets for various upscaling factors. It outperforms *SCN (n=128)* which obtains the second best results by about 0.2 dB across all the data sets, owing to the power of multiple inference modules.

We compare the visual quality of SR results among various methods in Figure 3.4. The region inside the bounding box is zoomed in and shown for the sake of visual comparison. Our proposed model *MSCN-4 (n=128)* is able to recover sharper edges and generate less artifacts in the SR inferences.

3.3.4 SR Performance vs. Inference Time

The inference time is an important factor of SR algorithms other than the SR performance. The relation between the SR performance and the inference time of our approach is analyzed in this section. Specifically, we measure the average inference time of different network structures in our method for upscaling factor ×2 on Set14. The inference time costs versus the PSNR values are displayed in Figure 3.5, where several other current SR methods

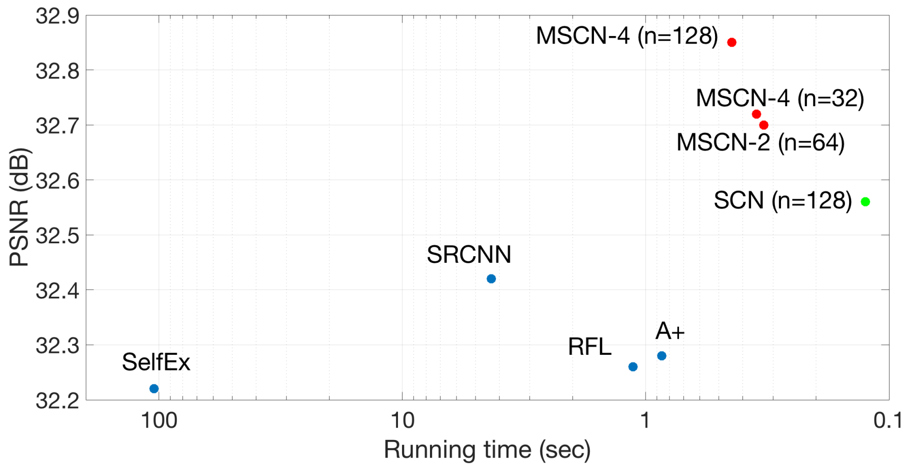


Figure 3.5: The average PSNR and the average inference time for upscaling factor ×2 on Set14 are compared among different network structures of our method and other SR methods. SRCNN uses the public slower implementation of CPU.

[45, 52, 41, 42] are included as reference (the inference time of SRCNN is from the public slower implementation of CPU). We can see that generally, the more modules our network has, the more inference time is needed and the better SR results are achieved. By adjusting the number of SR inference modules in our network structure, we can make the tradeoff between SR performance and computation complexity. However, our slowest network still has the superiority in terms of inference time, compared with other previous SR methods.

3.4 Conclusion

In this section, we propose to jointly learn a mixture of deep networks for single image super-resolution, each of which serves as a SR inference module to handle a certain class of image signals. An adaptive weight module is designed to predict pixel-level aggregation weights of the HR estimates. Various network structures are analyzed in terms of the SR performance and the inference time, which validates the effectiveness of our proposed model design. Extensive experiments show that our proposed model is able to achieve outstanding SR performance along with more flexibility of design.

Chapter 4

ROBUST VIDEO SUPER-RESOLUTION WITH LEARNED TEMPORAL DYNAMICS

4.1 Introduction

Video super-resolution (SR) can be cast as the problem of estimating high-resolution (HR) frames from a low-resolution (LR) sequence, which has been extensively studied in the past few decades. This problem has attracted growing attention recently as the high-definition display (e.g. HDTV) has become prevalent in the market and even ultra-high-definition video formats, such as 4K UHD (2160p) and 8K UHD (4320p), have emerged. There is an increasing demand for converting low-quality video sequences into high-definition ones so that they can be played on the high-definition displays in a visually pleasant manner.

Video SR can be tackled from two perspectives: utilizing the intra-frame spatial relation and the inter-frame temporal relation. In the recent years, neural network based models have become dominant for image SR to model the spatial relation due to the large model capacity and the end-to-end learning strategy [18, 55, 41, 57, 59, 60, 61, 62, 63]. We have witnessed the remarkable improvement of image SR accuracy from neural network based approaches with the help of deeper architecture, more layers and more trainable parameters.

Instead of only considering the intra-frame spatial relation for single image SR, the inter-frame temporal relation is more important for video SR, as research suggests that the human vision system is more sensitive to motion [64]. Hence it is crucial for video SR algorithms to capture and model the effect of motion information on visual perception. To meet this need, a number of video SR algorithms are proposed [65, 66, 67, 68, 69, 70, 71] to conduct pixel-level motion and blur kernel estimation based on image priors, e.g., sparsity and total variation. These methods usually formulate a sophisticated optimization problem which incurs heavy computational cost and thus they suffer considerable inference time. Recently, neural network based models have also emerged in this domain [70, 72, 73]. Some of them model the temporal relation on a fixed temporal scale via explicitly conducting motion compensation to align consecutive frames as inputs to the network model [70, 73], while the rest develop recurrent network architecture to use the long-term temporal dependency [72].

Motion estimation is crucial for temporal dependency modeling and can drastically influence the result of video SR. Rigid and smooth motion is usually easy to model among neighboring frames, in which case it is beneficial to include neighboring frames to super-resolve the center frame. In contrast, with complex motion or parallax presented across neighboring frames, motion estimation becomes challenging and erroneous motion compensation can undermine the result of video SR.

To this end, we propose a **temporal adaptive neural network** that is able to robustly handle various types of motion and adaptively select the optimal range of temporal dependency to extract useful information among consecutive frames and alleviate the detrimental effect of erroneous motion estimation simultaneously. Our network takes as input a number of aligned LR frames after motion compensation, and applies filters of different temporal sizes to generate multiple HR frame estimates. The resultant HR estimates are adaptively aggregated according to the confidence of motion compensation which is inferred from the temporal adaptive network. The proposed network architecture extends the idea of the Inception module in GoogLeNet [1] to the temporal domain. Our model gains the robustness to imperfect motion compensation through network learning, instead of simply boosting optical flow quality by using computationally more expensive methods as in [70], or extracting motion information only from a single fixed temporal scale as in [73].

Furthermore, we develop a very deep network architecture by increasing the number of layers and trainable parameters in our temporal adaptive network, in order to push the limit of the SR accuracy. Convolutions are directly applied in the LR image space and a deconvolution layer is appended at last to upscale the feature representation and generate the HR prediction in the desired spatial size. This design has the benefit of drastically decreasing the computation cost and thus facilitating the training effort as well as substantially reducing inference time. We provide a thorough analysis of this temporal adaptive design and systematically investigate the relation between the SR performance and various network structures. We demonstrate that the temporal adaptive design obtains a clear advantage in handling complex motion over the counterpart using temporal filters with fixed length.

Besides modeling motion information in the temporal domain, we also conduct motion compensation in the spatial domain to enhance the temporal modeling. We explore multiple image alignment methods to improve video SR. We find that the sophisticated optical flow based approach may not be optimal, as the estimation error on complex motion adversely affects the subsequent SR. Therefore, we reduce the complexity of motion by estimating only a small number of parameters of spatial transform, and provide a more robust approach to aligning frames. Moreover, inspired by the spatial transformer network [74], we propose a **spatial alignment network**, which infers the spatial transform between consecutive frames and outputs aligned frames for video SR. Compared to the conventional optical flow based methods, this spatial alignment network needs much less inference time and can be cascaded with the temporal adaptive network and trained jointly. We observe that reducing the complexity of motion in this manner increases the robustness of image alignment to complex motion and thus provides better SR performance, which shows that the spatial alignment network is beneficial to SR by providing aligned input frames.

We conduct a comprehensive comparison of our proposed method and other recent video SR approaches on public video datasets. Extensive video SR results show that our method outperforms competing methods by a large margin in terms of not only spatial consistency but also temporal coherence. In addition, as SR can be utilized as a pre-processing step to improve the performance of high-level vision tasks, we choose video face recognition as an example and evaluate the improvement of its performance with the help of various video SR approaches. Our proposed method is able to achieve the most improvement showing that it can not only produce visually pleasant HR results, but also better recover semantically faithful information benefiting high-level vision tasks. A more detailed version can be found in [75].

4.2 Related Work

In this section, we give a brief review of deep learning techniques for SR. We first review the methods for image SR and then discuss those for video SR.

4.2.1 Deep Learning for Image SR

Image SR has been widely studied over a few decades and quite a number of approaches have been developed. Here we only focus on neural network based methods, which have shown impressive performance for image SR. Dong et al. [18, 41] pioneer a three-layer fully CNN, termed SRCNN, to approximate the complex non-linear mapping between the LR image and the HR counterpart. Due to the end-to-end training strategy that jointly optimizes all the parameters and the large learning capacity of neural networks, this method notably outperforms the conventional shadow counterparts. A sparse coding network (SCN) that closely mimics the sparse representation approach for image SR is designed by Wang et al. [55, 56], demonstrating the benefit of domain expertise from sparse coding in the design of neural networks for image SR. A very deep CNN with residual architecture and many more layers and trainable weights, termed VDSR, is proposed by Kim et al. [57] to attain impressive SR accuracy. Kim et al. [59] design another network which has recursive architecture with skip-connection for image SR, dubbed as DRCN, to boost performance while exploiting a small number of model parameters. Liu et al. [76] propose to learn a mixture of networks that can work in a complementary manner to further improve SR results. Shi et al. [60] propose an efficient sub-pixel convolutional neural network, termed ESPCN, which directly applies convolutions on the LR space of images and learns an array of upscaling filters in the last layer of their network, in order to considerably reduce the computation cost and achieve real-time SR. Dong et al. [61] adopt a similar strategy to enhance SRCNN with smaller filter sizes and more convolution layers. More recently, Ledig et al. [77] utilize a generative adversarial network to produce plausible-looking fine details and textures for image SR of large upscaling factors. Lai et al. [62] propose the Laplacian pyramid super-resolution network to progressively reconstruct the sub-band residuals of HR images. Tai et al. [63] propose another deep recursive residual network which improves DRCN with more convolutional layers and more residual units. Yu et al. [78] extend a CNN based model for computed tomography SR. Han et al. [79] propose a dual-state recurrent network to exploit both LR and HR signals jointly and allow information to be exchanged between LR and HR spaces in both directions.

4.2.2 Deep Learning for Video SR

With the popularity of neural networks for image SR, researchers have started developing video SR methods of neural networks. Liao et al. [70] first generate an ensemble of SR draft via motion compensation under different parameter settings, and then use a CNN to reconstruct the HR frame from all drafts. Huang et al. [72] avoid explicit motion estimation by extending SRCNN for single image SR along the temporal dimension forming a recurrent convolutional network to capture the long-term temporal dependency. This neural network models the long-term temporal information in bidirectional “recurrent-like” connections. These methods either suffer considerable running time due to the conventional optical flow based alignment methods, or fail to precisely handle large-displacement and other complicated motions which degrade the final SR performance. Kappeler et al. [73] extend SRCNN for video SR by taking as input consecutive motion compensated frames on a fixed temporal scale and experiment with various ways to merge feature representations of video frames. However, the sophisticated and time-consuming optical flow based frame alignment is the shortcoming of this method.

4.3 Temporal Adaptive Neural Network

In this section, we introduce our temporal adaptive neural network. We start by giving an overview of the network, and then provide the detailed architecture. We finally show the training objective of the network.

4.3.1 Overview

Our model aims to estimate an HR frame from a set of local LR frames. The main challenge of video SR lies in the proper utilization of temporal information to handle various types of motion specifically. To address this problem,

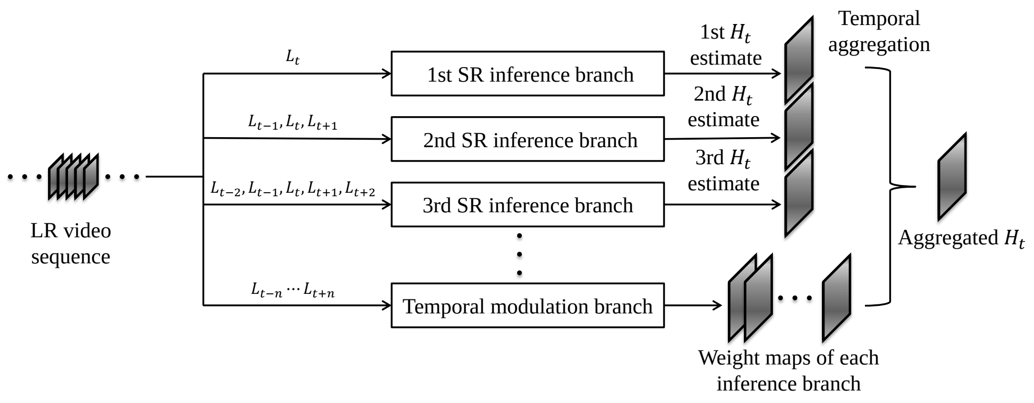


Figure 4.1: The overview of the temporal adaptive neural network. It consists of a number of SR inference branches and a temporal modulation branch. Each SR inference branch works on a different temporal scale and utilizes its temporal dependency to provide an HR frame prediction. These predictions are adaptively combined using pixel-level aggregations from the temporal modulation branch to generate the final HR frame.

we design a neural network to adaptively select the optimal temporal scale for video SR. The network has a number of **SR inference branches** $\{B_i\}_{i=1}^N$, where each B_i works on a different temporal scale i, and uses its temporal dependency on its scale to predict an HR estimate. We design an extra **temporal modulation branch**, T, to determine the optimal temporal scale and adaptively combine all the HR estimates based on motion information, at the pixel-level. All SR inference branches and the temporal modulation branch are incorporated and jointly learned in a unified network. The final estimated HR frame is aggregated from the estimates of all SR inference branches considering the motion information on various temporal scales. The overview of the temporal adaptive network is shown in Figure 4.1.

4.3.2 Network Architecture

SR inference branch

The SR inference branch B_i works on $2i-1$ consecutive LR frames. c denotes the number of channels for every input LR frame and r is the upscaling factor. The filters of SR inference branch B_i in the first layer have the temporal length of $2i-1$. Specifically, if we apply convolutions in the first layer, the convolutional filters are designed to have $(2i-1) \times c$ channels.

First we customize a recent neural network based SR model, ESPCN [60], as the SR inference branch due to its high SR accuracy and low computation cost, and use this lightweight model to validate the effectiveness of the temporal adaptive network. This network has three layers in total. The first layer is a convolutional layer with 64 kernels of size 5×5; the second layer is another convolutional layer with 32 kernels of size 3×3; the last layer is a sub-pixel convolution layer which uses $r^2 \times c$ kernels of size 3×3 to generate feature maps in $r^2 \times c$ channels and rearranges them into the HR prediction of c channels. The Rectified Linear Unit (ReLU) [26] is chosen as the activation function of the first and second layer. All the convolutions are conducted with stride one. Interested readers are referred to [60] for the details of ESPCN.

Afterward we develop a much deeper network with more trainable parameters as the SR inference branch in order to push the limit of the SR accuracy of our model, since more layers and larger model size usually lead to performance gain. This network employs an idea similar to ESPCN by directly extracting features from the LR image space in order to lower the computation cost, and consists of 19 convolutional layers and one deconvolution layer. Each convolutional layer has 64 kernels of size 3×3 with stride one and a ReLU as the activation function. The last deconvolution layer is used to enlarge the spatial dimension of feature maps into the desired size, which has c kernels of size $(2r - r \bmod 2) \times (2r - r \bmod 2)$ with stride r and results in the HR prediction of c channels.

Note that the design of the SR inference branch is not limited to these two choices, and all other network based SR models, such as SRCNN and SCN, can work as the SR inference branch as well. The output of B_i serves as an estimate of the final HR frame.

Temporal modulation branch

The principle of this branch is to learn the selectivity of our model on different temporal scales according to motion information. We propose to assign pixel-level aggregation weights on each HR estimate, and in practice this branch is applied on the largest temporal scale. For a model of N SR inference branches, the temporal modulation branch takes $2N - 1$ consecutive frames as input. Considering the computation cost and efficiency, for this branch we adopt a structure similar to that of the SR inference branch. The temporal modulation branch outputs the pixel-level weight maps on all N possible temporal scales.

Aggregation

The output of each SR inference branch is multiplied with its corresponding weight map from the temporal modulation branch in a pixel-wise manner, and then these products are summed to form the final estimated HR frame.

4.3.3 Training Objective

In training, we minimize the loss between the target HR frame and the predicted output as

$$\min_{\boldsymbol{\Theta}} \sum_{j} \|F(\boldsymbol{y}^{(j)}; \boldsymbol{\Theta}) - \boldsymbol{x}^{(j)}\|_2^2, \tag{4.1}$$

where $F(\boldsymbol{y}; \boldsymbol{\Theta})$ represents the output of the temporal adaptive network, $\boldsymbol{x}^{(j)}$ is the j-th HR frame and $\boldsymbol{y}^{(j)}$ are all the associated LR frames; $\boldsymbol{\Theta}$ is the set of parameters in the network.

If we use an extra function $W(\boldsymbol{y}; \theta_w)$ with parameter θ_w to represent the behavior of the temporal modulation branch, the cost function then can be expanded as:

$$\min_{\theta_w, \{\theta_{B_i}\}_{i=1}^{N}} \sum_{j} \| \sum_{i=1}^{N} W_i(\boldsymbol{y}^{(j)}; \theta_w) \odot F_{B_i}(\boldsymbol{y}^{(j)}; \theta_{B_i}) - \boldsymbol{x}^{(j)}\|_2^2. \tag{4.2}$$

Here $\odot$ denotes the pointwise multiplication and $F_{B_i}(\boldsymbol{y}; \theta_{B_i})$ is the output of the SR inference branch B_i.

In practice, we first train each SR inference branch B_i individually as in (4.1) using the same HR frame as the training target, and then use the resultant models to initialize the SR inference branches when training the temporal adaptive network following (4.2). This training strategy speeds up the convergence dramatically without sacrificing the prediction accuracy of SR.

4.4 Spatial Alignment Methods

The alignment of consecutive LR frames is usually applied as a prepossessing step for video SR, and has been proved beneficial in prior works [70, 73]. We investigate several image alignment methods in order to provide better motion compensated frames for the temporal adaptive network.

4.4.1 Rectified Optical Flow Alignment

It is well known that since the complex motion is difficult to model, the conventional optical flow based image alignment using erroneous motion estimation may introduce artifacts, which can be propagated to the following SR step and have a detrimental effect on it. We try simplifying the motion in the patch level to integer translations for avoiding interpolation which may cause blur or aliasing. Given a patch and its optical flow, we estimate the integer translation along the horizontal and vertical directions by rounding the average horizontal and vertical displacement of all pixels in this patch, respectively. This scheme, termed *rectified optical flow alignment*, proves more beneficial to the following SR than the conventional optical flow based image alignment, which will be shown in Section 4.5.4.

4.4.2 Spatial Alignment Network

In order to align neighboring frames, we propose a spatial alignment network which has the merits of efficient inference and end-to-end training with the SR network. The architecture of the spatial alignment network and its cascade with the temporal adaptive network are shown in Figure 4.2.

Each time the spatial alignment network takes as input one LR reference frame and one LR neighboring frame (as the source frame), and generates as output an aligned version of this neighboring frame. Specifically, these two LR frames are first fed into a localization network to predict the spatial transform parameter $\hat{\theta}_{ST}$, which is then applied to the source frame in the spatial transform layer, to produce the LR aligned frame. Given the success of the rectified optical flow alignment, we design the localization network to infer only two translation parameters. The LR source frame and reference frame are stacked to form the input of $2 \times c$ channels. The localization network

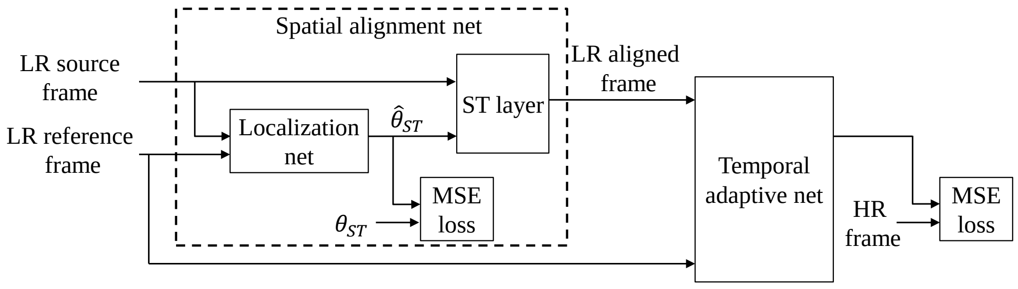


Figure 4.2: The architecture of the spatial alignment network and its cascade with the SR network. Each time one LR reference frame and its neighboring LR frame (as the source frame) are first fed into a localization net to regress the alignment parameter $\hat{\theta}_{ST}$. Then $\hat{\theta}_{ST}$ is applied to the source frame in the spatial transform layer to generate an LR aligned frame. The LR reference frame and all the aligned neighboring frames are the input to the subsequent SR network. During the joint training, we minimize the weighted sum of the MSE loss in the SR network and the MSE loss between $\hat{\theta}_{ST}$ and the ground truth θ_{ST} in the spatial alignment network.

has two convolutional layers of 32 kernels of size 9×9. Each convolutional layer is followed by a max-pooling layer with stride two and kernel of size two. Then there are two fully connected layers which have 100 and two nodes, respectively, to regress the two translation parameters. In practice, the spatial alignment network works on the patch level for better motion modeling, and only the center region in each patch is kept in order to avoid the empty area near the boundary after translation. In training, the loss in the spatial alignment network is defined as the mean squared error (MSE) between $\hat{\theta}_{ST}$ and the ground truth θ_{ST}, which can be acquired from other image alignment methods, e.g. the rectified optical flow alignment.

The LR reference frame and all the resultant aligned neighboring frames from this network are used together as input to the temporal adaptive network for SR.

We propose to train this network and the temporal adaptive network in an end-to-end fashion, since joint learning is usually more advantageous than separate learning. During the joint training, we minimize the weighted sum of the loss in the spatial alignment network and the loss from the temporal adaptive network, as

$$\min_{\{\boldsymbol{\Theta},\theta_L\}} \sum_j \|F(\boldsymbol{y}^{(j)};\boldsymbol{\Theta}) - \boldsymbol{x}^{(j)}\|_2^2 + \lambda \sum_j \sum_{k\in\mathcal{N}_j} \|\hat{\theta}_{ST}^{(k)} - \theta_{ST}^{(k)}\|_2^2, \tag{4.3}$$

where $\mathcal{N}_j$ denotes the set of LR frames associated to the j-th HR frame, and λ is the scaling factor for balancing these two losses. θ_L represents the set of parameters in the localization net.

4.5 Experiments

We discuss our experiments in this section. We first introduce the datasets that we use and provide the implementation details of our system. Next, we provide a thorough analysis of our network architecture and spatial alignment methods. We then compare our results with state-of-the-art, and show the influence of video SR with respect to the high-level vision tasks. Finally we provide the running time analysis of our proposed approach.

4.5.1 Datasets

Since the amount of training data plays an important factor in training neural network, we combine three public datasets of uncompressed videos: LIVE Video Quality Assessment Database [80], MCL-V Database [81] and TUM 1080p Data Set [82], so as to collect sufficient training data. We prepare data in 3D volume from video clips that have abundant textures and details and are separated by shots or scenes. We test our method on six videos: *calendar, city, foliage, penguin, temple* and *walk* from [70], each of which has 31 frames. In addition, we choose a 4K video dataset, Ultra Video Group Database [83], as a second test set for the SR performance comparison. This dataset consists of seven 120fps sequences: *Beauty, Bosphorus, HoneyBee, Jockey, ReadySteadyGo, ShakeNDry* and *YachtRide*, covering various types of scenes and motion. *ShakeNDry* has 300 frames while the other six have 600 frames. The original HR frames are downsized by bicubic interpolation to generate LR frames for training.

4.5.2 Implementation Details

Following the convention in [18, 55, 72, 60], we convert each frame into the YCbCr color space and only process the luminance channel using our model. Hence each frame has $c = 1$ channel. We focus on the upscaling factor of four, which is usually the most challenging case in video SR. The input LR frames to the temporal adaptive network are the volume of $5 \times 30 \times 30$ pixels, i.e. patches of 30×30 pixels from five consecutive frames. These data are augmented with rotation, reflection and scaling, providing about 10 million training samples. We implement our model using Caffe [58]. We apply a constant learning rate of 10^{-4} for the first two layers and 10^{-5} for the last layer, a batch size of 64 with momentum of 0.9. We terminate the training after five million iterations. Experiments are conducted on a workstation with one GTX Titan X GPU. Based on the architecture of each SR inference branch, we can initialize the parameters from the single frame SR model, except that the filter weights in the first layer are evenly divided along the temporal dimension. In practice, it is observed that this initialization strategy leads to faster convergence and usually improves the performance.

4.5.3 Analysis of Network Architecture

We experiment with different types of architecture for the temporal adaptive network using the two SR inference branch of three layers and 20 layers, respectively, in order to investigate the relation between the SR performance and various structures. Recall that B_i denotes the SR inference branch working on $2i-1$ LR frames, and T is the temporal adaptive branch. We explore the structures that contain (1) only B_1, (2) only B_2, (3) only B_3, (4) $B_{1,2}$, (5) $B_{1,2}+T$, (6) $B_{1,2,3}$ and (6) $B_{1,2,3}+T$. $B_{1,2}$ denotes the straight average of HR predictions from B_1 and B_2, and $B_{1,2,3}$ follows the similar definition. Note that in the case of (1), each frame is super-resolved independently. For the experiments in this section, LR consecutive frames are aligned as in Section 4.4.1.

The PSNR (unit: dB) comparisons of six test sequences by 4x upscaling are shown in Table 4.1. Average PSNR of all the frames in each video is shown in the table. It can be observed that in both the cases of SR inference branch of three layers and 20 layers, generally the network performance is

Table 4.1: PSNR comparisons of different network structures: average PSNR of all the frames in each video sequence by 4x upscaling is displayed. Best results are shown in bold. From left to right, B_i: the HR prediction from the i-th SR inference branch; $B_{1,2}$: the straight average of HR predictions from B_1 and B_2; $B_{1,2}+T$: the adaptive aggregation of the outputs of B_1 and B_2 with joint learning. $B_{1,2,3}$ and $B_{1,2,3}+T$ follow the similar definitions as $B_{1,2}$ and $B_{1,2}+T$, respectively.

	B_1	B_2	B_3	$B_{1,2}$	$B_{1,2}+T$	$B_{1,2,3}$	$B_{1,2,3}+T$
calendar	20.88	21.16	21.32	21.10	21.26	21.24	**21.51**
city	25.70	25.91	26.25	25.89	25.97	26.10	**26.46**
foliage	24.29	24.51	24.81	24.47	24.58	24.70	**24.98**
penguin	36.46	36.41	36.40	36.56	36.53	36.59	**36.65**
temple	28.97	29.30	29.72	29.33	29.46	29.64	**30.02**
walk	27.69	27.78	27.90	27.88	27.90	28.00	**28.16**
average	27.33	27.51	27.73	27.54	27.62	27.71	**27.96**

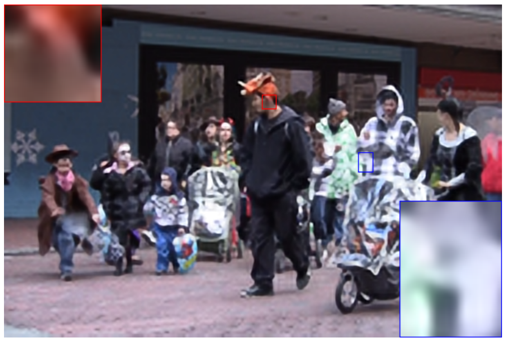
(a) Temporal adaptive network using the SR inference branch of three layers

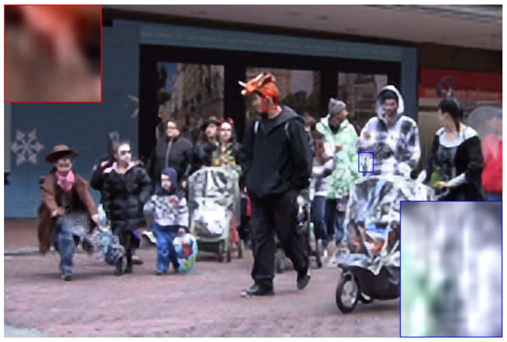
	B_1	B_2	B_3	$B_{1,2}$	$B_{1,2}+T$	$B_{1,2,3}$	$B_{1,2,3}+T$
calendar	21.68	22.22	22.22	22.15	22.34	22.27	**22.60**
city	25.89	26.46	26.72	26.36	26.55	26.58	**27.01**
foliage	24.47	25.21	25.32	25.07	25.29	25.26	**25.56**
penguin	36.95	36.86	36.85	36.99	37.01	37.03	**37.09**
temple	29.60	30.66	30.97	30.52	30.78	30.91	**31.36**
walk	28.09	28.62	28.70	28.58	28.66	28.75	**29.06**
average	27.78	28.34	28.46	28.28	28.43	28.47	**28.78**

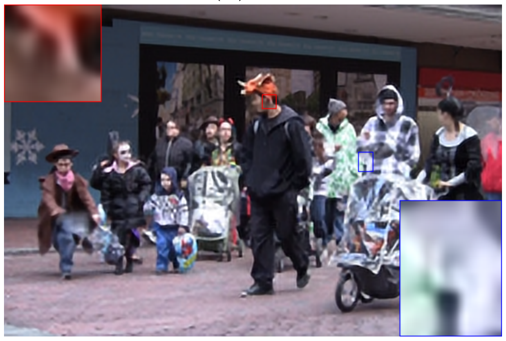
(b) Temporal adaptive network using the SR inference branch of 20 layers

enhanced as more frames are involved, and $B_{1,2,3}+T$ performs the best among all the structures. $B_{1,2}+T$ obtains higher PSNRs than $B_{1,2}$ and $B_{1,2,3}+T$ is superior to $B_{1,2,3}$, which demonstrates the advantage of adaptive aggregation over the straight averaging on various temporal scales. For the same temporal architecture, the model with 20-layer SR inference branch always outperforms the model with three-layer SR inference branch, showing that more layers and more weights increasing the capacity of temporal adaptive network can lead to better restoration accuracy.

In order to show the visual difference of SR results among various structures, we choose one frame from *walk*, and show the SR results from B_1, B_3, $B_{1,2,3}+T$ as well as the ground truth HR frame in Figure 4.3 with two zoom-in regions. The region in the blue bounding box contains part of a flying pigeon which is subject to complex motion among consecutive frames and thus is challenging for accurate motion estimation. It can be observed


(a) B_1


(b) B_3


(c) $B_{1,2,3} + T$

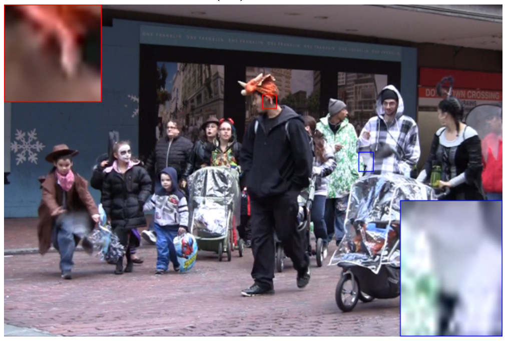

(d) Ground truth

Figure 4.3: Examples of SR results from *walk* by 4x upscaling using different network structures. Compared with B_1 and B_3, the temporal adaptive architecture $B_{1,2,3} + T$ is able to effectively handle both the rigid motion shown in the top left zoom-in region and the complex motion in the bottom right zoom-in region.

that the HR inference from B_1 has many fewer artifacts than that from B_3, indicating the short-term temporal dependency alleviates the detrimental effect of erroneous motion estimation in this case. In contrast, the zoom-in region in the red bounding box includes the ear and part of the neck of the pedestrian with nearly rigid motion, in which case the HR inference from B_3 is able to recover more details. This result shows the necessity of the long-term temporal dependency. $B_{1,2,3} + T$ is able to generate better HR estimates compared with its single-branch counterparts in these two zoom-in regions, and thus shows the effectiveness of the temporal adaptive design in our model.

To further analyze the temporal adaptive branch, we visualize the weight maps given by the temporal modulation branch of two frames from *foliage* and *temple* for each SR inference branch. In addition, we use the index of the maximum weight among all SR inference branches at each pixel to draw a

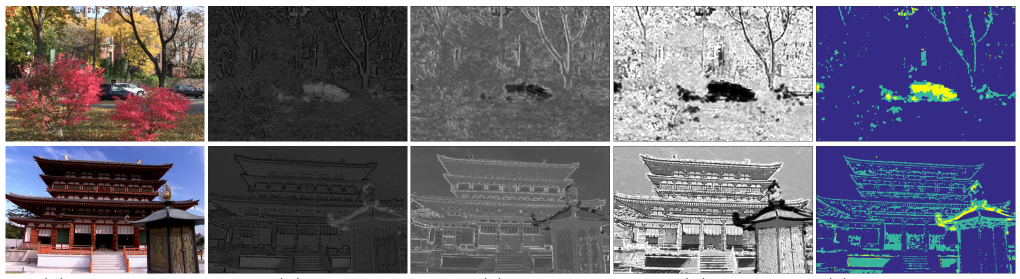


Figure 4.4: Weight maps of three SR inference branches given by the temporal modulation branch in the architecture $B_{1,2,3} + T$. The max label map records the index of the maximum weight among all the SR inference branches at every pixel, which is shown in the last column. B_1, B_2 and B_3 are indicated in yellow, teal and blue, respectively. Frames from top to bottom: *foliage* and *temple* by 4x upscaling.

max label map. These results are displayed in Figure 4.4. It can be seen that B_1 mainly contributes the region of cars in *foliage* and the top region of the lantern in *temple*, which are subject to large displacements caused by object motion and camera motion. In contrast, the weight map of B_3 has larger responses in the region subject to rigid and smooth motion, such as the plants of the background in *foliage* and the sky in *temple*. Their complementary behaviors are properly utilized by the temporal adaptive aggregation of our model.

4.5.4 Analysis of Spatial Alignment Methods

We conduct experiments with the image alignment methods discussed in Section 4.4. We choose the algorithm of Liu [84] to calculate optical flow, considering the motion estimation accuracy and running time. Figure 4.5 shows a case where the conventional optical flow alignment fails and introduces obvious artifacts in LR frames while the rectified optical flow alignment succeeds.

As for the spatial alignment network (SAN), we use the result of the rectified optical flow alignment as the training target of the spatial transform parameter. We choose the input patch size as 60×60 pixels for the spatial alignment network to achieve the best motion modeling. We keep the center region to be 30×30 pixels in each patch and discard the remaining region after translation, such that SAN can handle horizontal and vertical transla-

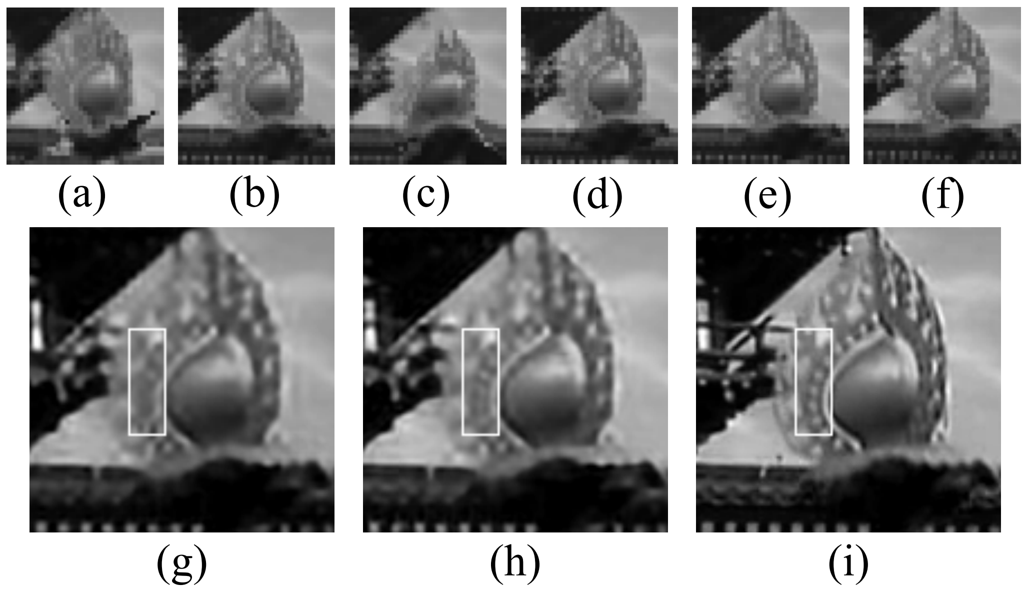


Figure 4.5: Comparison between the optical flow alignment and our proposed rectified optical flow alignment. (a)-(c): three consecutive LR patches warped by optical flow alignment. (d)-(f): the same three LR patches aligned by rectified optical flow alignment. (g): HR prediction from (a)-(c). (h): HR prediction from (d)-(f). (i): ground truth. Note that the rectified optical flow alignment recovers more details inside the white bounding box.

tions by up to 15 pixels in each direction with respect to the reference patch. In real application, the whole LR image is split into overlapped patches of 60×60 pixels such that the super-resolved patches can be tiled together to fully cover the HR image. The size of the overlap depends on the size of the cropped border of SAN and the following SR network. The whole LR image is padded with 0s to compensate for the cropped border of SAN, when we recover the boundary of the output image. In the joint training λ is set as -10^3 empirically. The visualization of the inputs and outputs of the spatial alignment network is shown in Figure 4.6. It is obvious that the output neighboring frames are aligned to the reference frame.

The PSNR comparisons between these alignment methods on six video sequences by 4x upscaling are shown in Table 4.2. Average PSNR of all the frames in each video is shown, and only B_3 is used in the SR network with the lightweight SR inference branch of three layers. Our proposed rectified optical flow alignment achieves the highest PSNR, demonstrating its superiority over the conventional optical flow alignment. The approach of spatial alignment

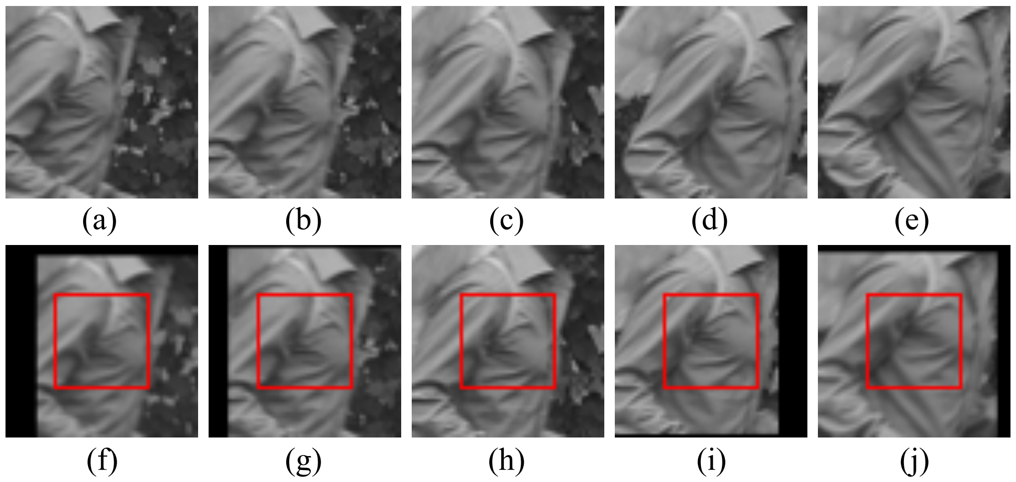


Figure 4.6: Visualization of the inputs and outputs of the spatial alignment network. (a)-(e): five consecutive LR frames as inputs. (f)-(j): their corresponding outputs. Note that (c) and (h) are the same reference frame without processing of this network. Only the region in the center red bounding box is kept for SR.

network clearly improves SR quality over its plain counterpart, which shows the effectiveness of its alignment.

4.5.5 Comparison with State-of-the-Art

Spatial Consistency

We use our best model $B_{1,2,3} + T$ with rectified optical flow alignment and experiment with two choices of SR inference branch described in Section 4.3.2: the lightweight version of three layers, denoted as *Proposed-S* and the much deeper network of 20 layers, denoted as *Proposed-L*. We compare them with several recent image and video SR methods: VSRnet [73], Bayesian method [68], Deep-DE [70], ESPCN [60] and VDSR [57] on the six test sequences. We use the model and code of VSRnet, Deep-DE and VDSR from their websites, respectively. The source code of Bayesian method is unavailable, so we adopt the reimplementation of Bayesian method in [69] and use five consecutive frames to predict the center frame. We implement ESPCN by ourselves since its source code is unavailable as well. We report the result on only the center frame of each sequence in that Deep-DE requires

Table 4.2: PSNR comparisons of various frame alignment methods: average PSNR of all the frames in each video sequence by 4x upscaling is displayed. Best results are shown in bold. From left to right, raw: raw LR frames; OF: conventional optical flow alignment; ROF: rectified optical flow alignment; SAN: spatial alignment network.

	raw	OF	ROF	SAN
calendar	21.07	21.28	**21.32**	21.27
city	25.92	**26.29**	26.25	26.23
foliage	24.49	24.62	**24.81**	24.78
penguin	36.37	36.32	**36.40**	36.29
temple	29.40	29.52	**29.72**	29.60
walk	27.82	27.79	**27.90**	27.83
average	27.51	27.64	**27.73**	27.67

Table 4.3: PSNR (SSIM) comparisons of various video SR methods: PSNR of only the center frame in each video sequence by 4x upscaling is displayed. Red indicates the best and blue indicates the second best performance.

	VSRnet [73]	Bayesian [68]	Deep-DE [70]	ESPCN [60]	VDSR [57]	Proposed-S	Proposed-L
calendar	20.99 (0.6182)	21.59 (0.7203)	21.40 (0.7096)	20.97 (0.6430)	21.50 (0.6739)	21.61 (0.6986)	22.73 (0.7466)
city	24.78 (0.5901)	26.23 (0.7147)	25.72 (0.6886)	25.60 (0.6179)	25.16 (0.6319)	26.29 (0.7052)	26.39 (0.7315)
foliage	23.87 (0.6079)	24.43 (0.7022)	24.92 (0.7074)	24.24 (0.6376)	24.41 (0.6436)	24.99 (0.7059)	25.50 (0.7201)
penguin	35.93 (0.9625)	32.65 (0.9577)	30.69 (0.9181)	36.50 (0.9668)	36.60 (0.9673)	36.68 (0.9682)	37.11 (0.9719)
temple	28.34 (0.8538)	29.18 (0.9043)	29.50 (0.9047)	29.17 (0.8779)	29.81 (0.8947)	30.65 (0.9069)	32.12 (0.9323)
walk	27.02 (0.8183)	26.39 (0.8327)	26.67 (0.7928)	27.74 (0.8407)	27.97 (0.8469)	28.06 (0.8485)	28.91 (0.8688)
average	26.82 (0.7418)	26.75 (0.8053)	26.48 (0.7869)	27.29 (0.7640)	27.58 (0.7764)	28.05 (0.8056)	28.79 (0.8285)

15 preceding and 15 succeeding frames to predict one center frame and there are only 31 frames in each sequence. We display several visual results in Figure 4.7. It can be seen that our method *Proposed-L* is able to recover more fine details and less artifacts. The PSNRs and SSIMs are shown in Table 4.3. Our method *Proposed-L* achieves both the highest PSNR and SSIM, which significantly outperforms all other methods over all the frames. Our method *Proposed-S* obtains the second best result in terms of average PSNR and SSIM, but its average PSNR is 0.74 dB lower than Proposed-L, which reveals the fact that the deeper architecture and the larger model size are highly beneficial to improve the performance of the temporal adaptive network.

For the application of HD video SR, we compare our two models with

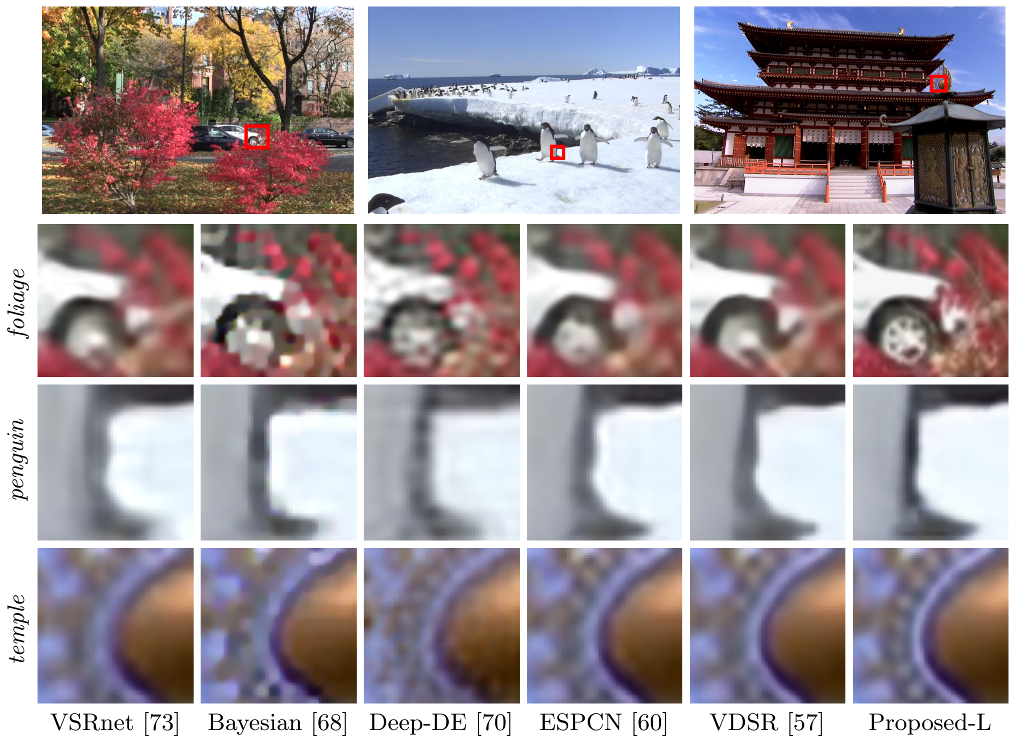


Figure 4.7: Visual comparisons of SR results by 4x upscaling among different methods. From top to bottom: zoom-in regions from *foliage*, *penguin* and *temple*.

VSRnet and ESPCN on Ultra Video Group Database. We do not include Bayesian method and Deep-DE for comparison, since both of them take multiple hours to predict one 4K HR frame (only the CPU version of code for Deep-DE is available). The PSNR and SSIM comparisons of these methods are in Table 4.4, where the average PSNR and SSIM of all the frames in every video are shown. The visual results are displayed in Figure 4.8. Our method *Proposed-L* obtains both the highest PSNR and SSIM consistently over all the video sequences, while our method *Proposed-S* achieves the second best. Visual results on Ultra Video Group Database are shown in Figure 4.8. It is noted that our method *Proposed-L* produces more visually pleasant results with sharper edges.

Temporal Coherence

We compare the temporal coherence of the SR results from our model and other competing methods. We adopt the MOVIE index [85], which is a metric

Table 4.4: PSNR (SSIM) comparisons of several video SR methods on Ultra Video Group Database by 4x upscaling. Red indicates the best and blue indicates the second best performance.

	VSRnet [73]	ESPCN [60]	VDSR [57]	Proposed-S	Proposed-L
Beauty	35.34 (0.7835)	35.56 (0.7933)	35.51 (0.7921)	35.59 (0.7939)	35.95 (0.8027)
Bosphorus	42.78 (0.9625)	42.78 (0.9620)	42.67 (0.9621)	43.07 (0.9658)	43.53 (0.9693)
HoneyBee	39.47 (0.9080)	39.60 (0.9104)	39.67 (0.9111)	39.69 (0.9124)	40.02 (0.9151)
Jockey	40.05 (0.9229)	40.45 (0.9264)	40.50 (0.9272)	40.62 (0.9291)	41.21 (0.9330)
ReadySteadyGo	39.51 (0.9472)	40.16 (0.9494)	40.24 (0.9516)	40.61 (0.9560)	41.17 (0.9594)
ShakeNDry	38.86 (0.9087)	39.29 (0.9147)	39.37 (0.9153)	39.38 (0.9155)	39.70 (0.9182)
YachtRide	37.39 (0.9441)	37.49 (0.9452)	37.53 (0.9473)	37.79 (0.9502)	38.03 (0.9511)
Average	39.06 (0.9110)	39.33 (0.9145)	39.36 (0.9152)	39.54 (0.9176)	39.94 0.9213

Table 4.5: MOIVE ($\times 10^{-3}$) index comparisons of SR results by 4x upscaling among various SR methods. Best results are shown in bold.

	VSRnet [73]	ESPCN [60]	VDSR [57]	Proposed-L
calendar	13.13	12.62	13.06	**7.77**
city	6.29	5.82	5.99	**3.92**
foliage	4.02	3.48	3.55	**2.88**
penguin	2.02	1.62	1.60	**1.59**
temple	6.20	4.62	4.33	**2.33**
walk	6.27	5.13	5.16	**4.19**
average	6.32	5.55	5.62	**3.78**

to measure video quality from human perception and temporal consistency. We evaluate the MOVIE indices of six video sequences over SR results of the methods: VSRnet, ESPCN, VDSR and Proposed-L, and display them in Table 4.5. Our method Proposed-L achieves significant reduction in the MOVIE index, which demonstrates the superiority of the temporal coherence of its super-resolved sequences.

Moreover, we select a line in a fixed position of a frame and stack every line in this fixed position from a number of consecutive frames to create its temporal profile. We display the results from various SR methods in Figure 4.9. It can be seen that our method *Proposed-L* generates smoother and more consistent temporal changes in these temporal profiles, which proves the enhancement of temporal coherence from our method.

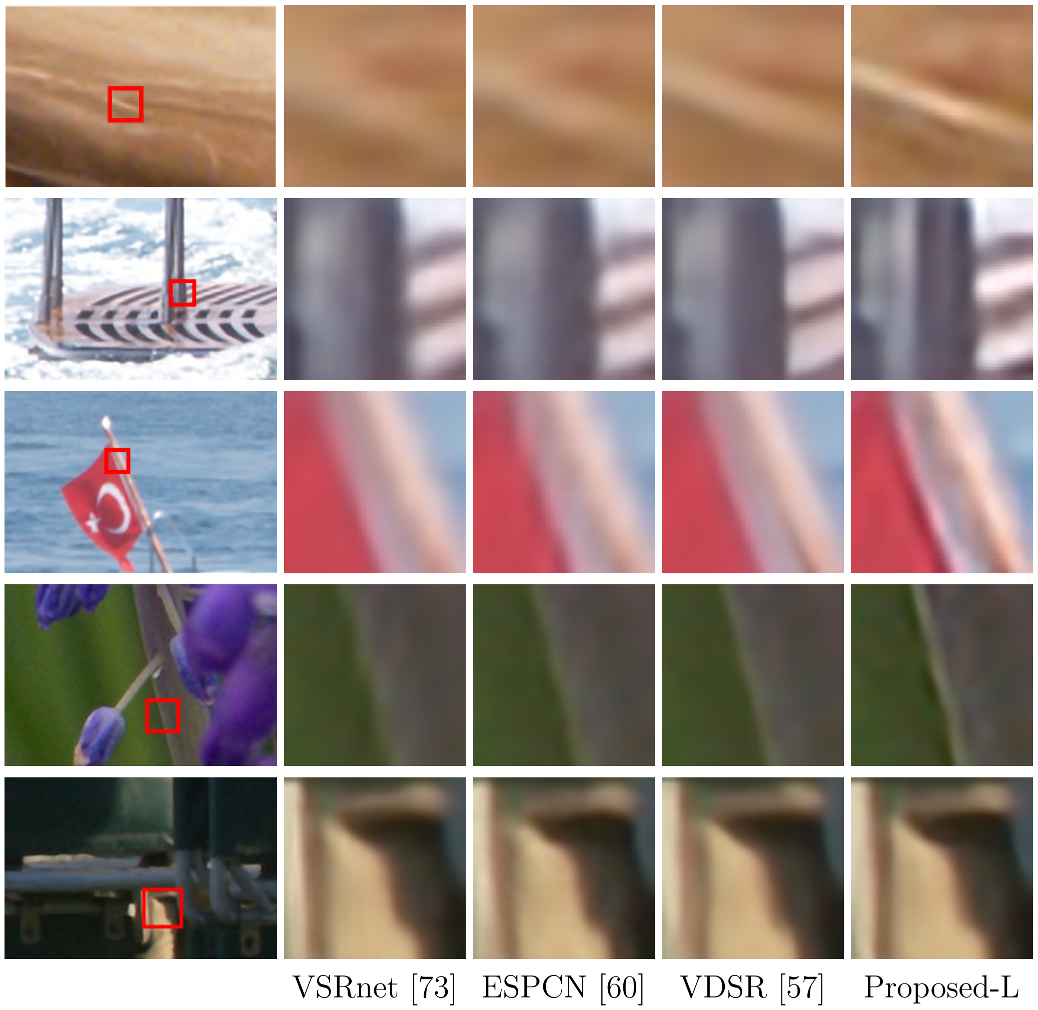


Figure 4.8: Visual comparisons of SR results on Ultra Video Group Database by 4x upscaling among different methods. From top to bottom: frames and zoom-in regions from *Beauty*, *Bosphorus*, *YachtRide*, *HoneyBee* and *ReadySteadyGo*.

4.5.6 Influence of Video SR on High-Level Vision Tasks

SR can be used as a pre-processing step to enhance the performance of high-level vision applications, such as semantic segmentation, face detection, emotion recognition and digit recognition [86, 87, 88, 89], especially when the input data is of low visual quality. Here we evaluate how various video SR algorithms could benefit the video face identification on YouTube Face (YTF) dataset [90]. We compare our method *Proposed-S* with VSRnet and ESPCN.

We form a YTF subset by choosing the 167 subject classes that contain more than three video sequences. For each class, we randomly select one

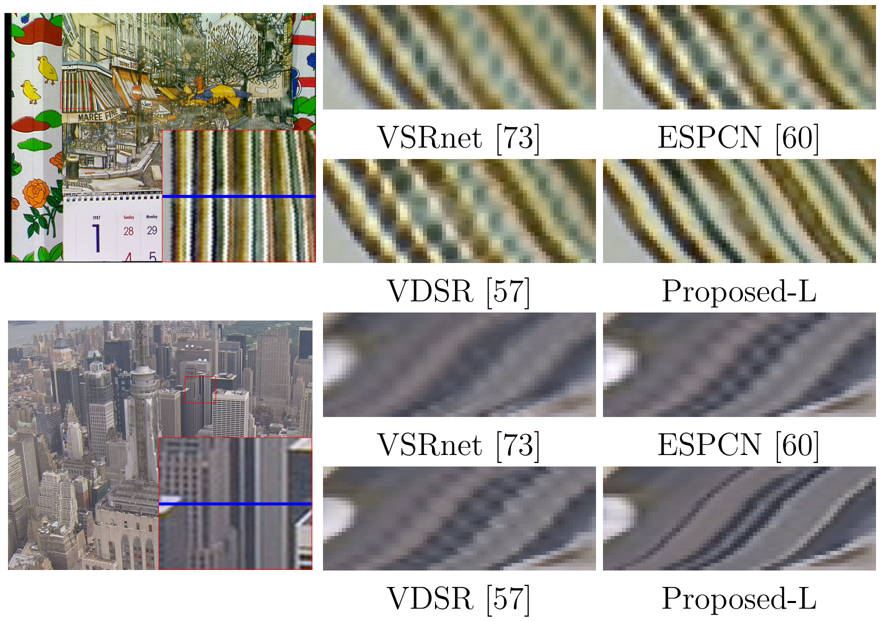


Figure 4.9: Visual comparison on the temporal profiles of the blue line in the zoom-in region from SR results by 4x upscaling over 27 consecutive frames among various methods. From top to bottom: *calendar* and *city*.

video for testing and the rest for training. Since the video duration varies a lot, we split original videos into clips of 50 frames and collect about 640 clips for testing.

The face regions are cropped and resized to 60×60 pixels, as the original resolution set, and then are downsampled by a factor of four to comprise the low-resolution set. We train a customized AlexNet [15] on the original resolution set as the classifier. Table 4.6 shows the architecture of our deep network used for this task. The input is one frame of facial region (60×60 pixels). We exploit four convolutional layers and append maximum pooling layers after the third and fourth convolutional layers. The last layer is a fully connected layer which has 167 nodes. We follow the convention of AlexNet to minimize the cross entropy loss during training. In addition, we train a network model directly on the low-resolution set as the baseline.

In testing, we feed into the network SR results from the low-resolution set by various algorithms. The prediction probability is aggregated over all the frames in each video. The top-1 and top-5 accuracy results of face identification are reported in Table 4.7. We include as the baseline the result from a model directly trained on the low-resolution set. Our method *Proposed-S* achieves both the highest top-1 and top-5 accuracy among all the SR meth-

Table 4.6: Network architecture for video face identification on YTF

Layer	Activation size
Input	$1 \times 60 \times 60$
$64 \times 9 \times 9$ Conv, stride 1, ReLU	$64 \times 52 \times 52$
$32 \times 5 \times 5$ Conv, stride 1, ReLU	$32 \times 48 \times 48$
$60 \times 4 \times 4$ Conv, stride 1, ReLU	$60 \times 45 \times 45$
Max pooling, kernel size 2, stride 2	$60 \times 23 \times 23$
$80 \times 3 \times 3$ Conv, stride 1, ReLU	$80 \times 21 \times 21$
Max pooling, kernel size 2, stride 2	$80 \times 11 \times 11$
Fc	167

Table 4.7: Face identification accuracy of various video SR methods on YouTube Face dataset downsampled by a factor of four. The baseline refers to the result from the model directly trained on LR frames. Best results are shown in bold.

	Top-1 accuracy	Top-5 accuracy
Baseline	0.442	0.709
VSRnet[73]	0.485	0.733
ESPCN[60]	0.493	0.734
Proposed-S	**0.511**	**0.762**

ods, showing that it is able not only to produce visually pleasant results, but also to recover more semantically faithful features that benefit high-level vision tasks.

4.5.7 Running Time Analysis

In testing, the running time is mainly composed of two parts: the frame alignment as pre-processing and the SR inference. For the first part, the spatial alignment network can align frames significantly faster than the optical flow based method. For 4x SR of 4K videos, it takes about 15 s to warp five consecutive frames of 540×960 pixels for the optical flow based method on an Intel i7 CPU, while the spatial alignment network needs only around 0.8 s on the same CPU, which reduces the time by one order of magnitude. For the second part, when we use the temporal adaptive network with the SR inference branch of three layers, B_1 takes about 0.3 s to generate a 4K HR frame. B_1, B_2 and B_3 differ only in the numbers of channels in the first layer, so their inference time varies a little. The inference time of the temporal modulation branch is comparable to that of the SR inference branch. All these branches enjoy the benefit of extracting features directly on LR frames and can be implemented in parallel for time efficiency.

4.6 Conclusion

In this chapter, we propose a temporal adaptive network and explore several methods of image alignment including a spatial alignment network, for learning the temporal dynamics to enhance video SR. Our proposed models with learned temporal dynamics are comprehensively evaluated on various video sequences and achieve state-of-the-art SR results. Both of the temporal adaptation and the spatial alignment modules show the increased robustness to complex motion and thus considerably improve SR performance over their plain counterparts.

Chapter 5

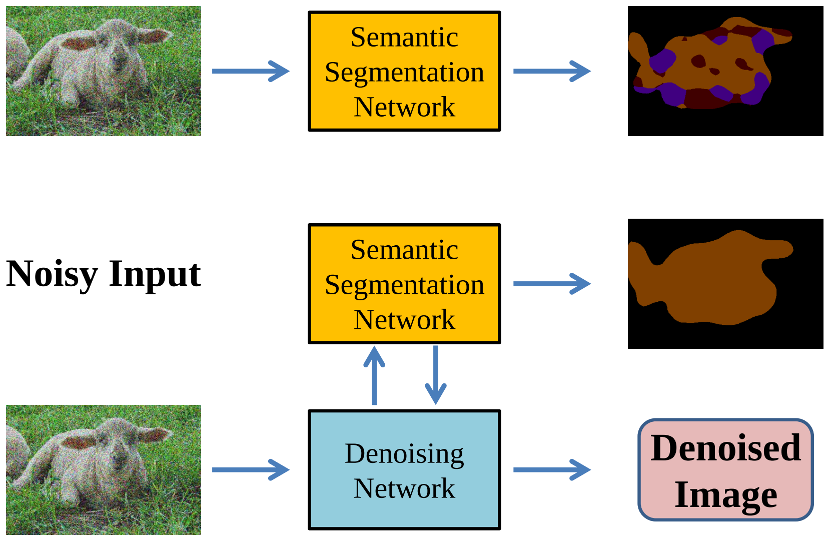
WHEN IMAGE DENOISING MEETS HIGH-LEVEL VISION TASKS: A DEEP LEARNING APPROACH

5.1 Introduction

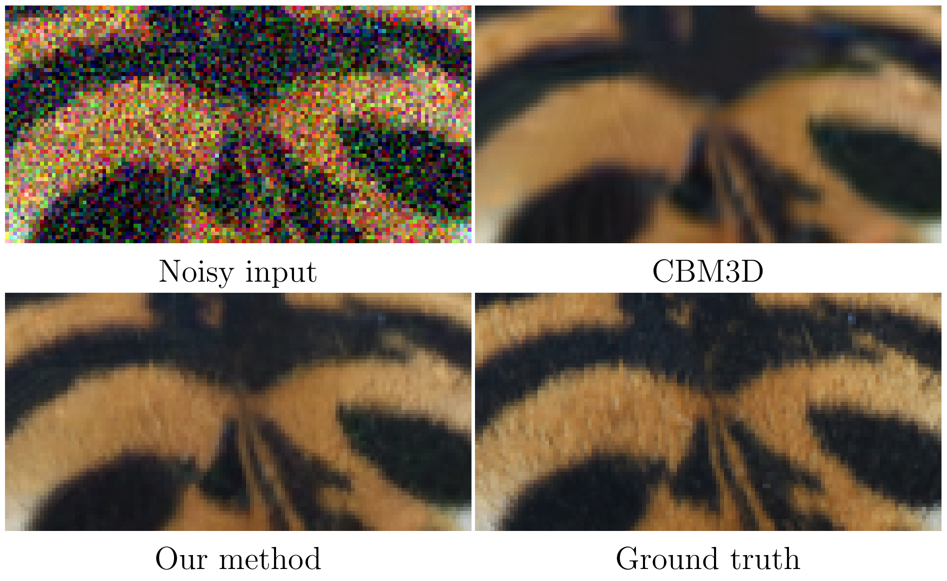
A common approach in computer vision is to separate low-level vision problems, such as image restoration and enhancement, from high-level vision problems, and solve them independently. In this chapter, we connect them by showing the mutual influence between the two, i.e., visual perception and semantics, and propose a new perspective for solving both the low-level and high-level computer vision problems in a single unified framework, as shown in Figure 5.1(a).

Image denoising, as one representative of low-level vision problems, is dedicated to recovering the underlying image signal from its noisy measurement. Classical image denoising methods take advantage of local or non-local structures presented in the image [91, 32, 92, 93, 94, 95]. More recently, a number of deep learning models have been developed for image denoising which demonstrated superior performance [96, 97, 98, 99, 100]. Inspired by U-Net [101], we propose a convolutional neural network for image denoising, which achieves the state-of-the-art performance.

While popular image denoising algorithms reconstruct images by minimizing the mean square error (MSE), important image details are usually lost which leads to image quality degradation. For example, over-smoothing artifacts in some texture-rich regions are commonly observed in the denoised output from conventional methods, as shown in Figure 5.1(b). To this end, we propose a cascade architecture connecting image denoising to a high-level vision network. We jointly minimize the image reconstruction loss and the high-level vision loss. With the guidance of image semantic information, the denoising network is able to further improve visual quality and generate more visually appealing outputs, which demonstrates the importance of semantic



(a)



(b)

Figure 5.1: (a) Upper: conventional semantic segmentation pipeline; lower: our proposed framework for joint image denoising and semantic segmentation. (b) Zoom-in regions of a noisy input, its denoised estimates using CBM3D and our proposed method, as well as its ground truth.

information for image denoising.

When high-level vision tasks are conducted on noisy data, an independent image restoration step is typically applied as preprocessing, which is suboptimal for the ultimate goal [87, 102, 103]. Recent research reveals that neural networks trained for image classification can be easily fooled by

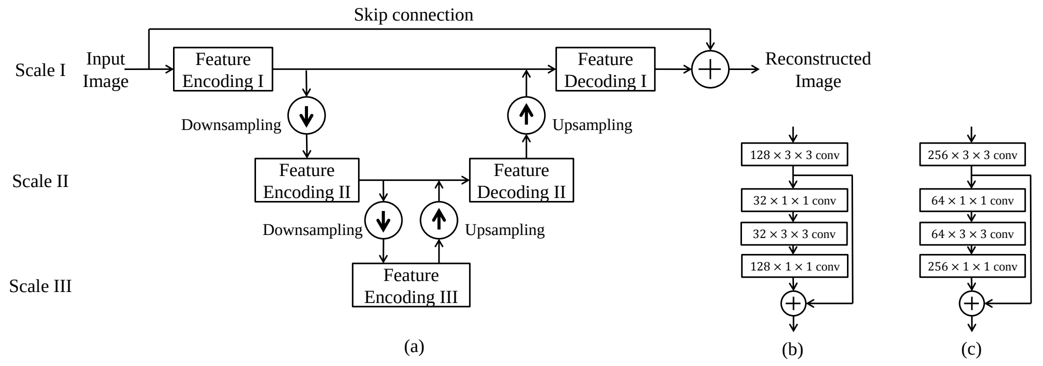


Figure 5.2: (a) Overview of our proposed denoising network. (b) Architecture of the feature encoding module. (c) Architecture of the feature decoding module.

small noise perturbation or other artificial patterns [104, 105]. Therefore, an application-driven denoiser should be capable of simultaneously removing noise and preserving semantic-aware details for the high-level vision tasks. Under the proposed architecture, we systematically investigate the mutual influence between the low-level and high-level vision networks. We show that the cascaded network trained with the joint loss not only boosts the denoising network performance via image semantic guidance, but also substantially improves the accuracy of high-level vision tasks. Moreover, our proposed training strategy makes the trained denoising network robust enough to different high-level vision tasks. In other words, our denoising module trained for one high-level vision task can be directly plugged into other high-level tasks without fine-tuning either module, which facilitates the training effort when applied to various high-level tasks [106]. The code is available online.[1]

5.2 Related Work

Denoising is the task of estimating the high-quality signal from its noisy measurements. Classical image denoising methods take advantage of local or non-local structures presented in the image explicitly. Natural images are well-known to be patch-wise sparse or compressible in transform domain, or over certain dictionary. Prior works [91, 107, 21] exploit such local structures and reduce noise by coefficient shrinkage for image restoration. The later

[1]`https://github.com/Ding-Liu/DeepDenoising`

approaches, including SSC [92], CSR [108], NCSR [93], GSR [109], WNNM [94], PCLR [110], PGPD [95], STROLLR [111], as well as BM3D [32] and its extension for color images CBM3D [112] – group similar patches within the image globally via block matching or clustering, and impose non-local structural priors on these groups, which usually lead to state-of-the-art image denoising performance.

More recently, the popular deep neural network techniques have been applied to low-level vision tasks. Specifically, a number of deep learning models have been developed for image denoising [96, 113, 97, 114, 98, 115], which can be classified into two categories: multilayer perception (MLP) based models and convolutional neural network (CNN) based models. Early MLP based models for image denoising include the stacked denoising autoencoder [96, 114]. Burger et al. [97] introduce a plain multilayer perception (MLP) and thoroughly compare its denoising performance with BM3D [32] in different experimental settings. CNNs are first utilized for image denoising by Jain and Seung [113]. Recent works [116, 99] attempt to unfold the iterative algorithms, and construct a cascaded convolutional filtering architecture for image denoising. Very deep networks are adopted with skip connections for image restoration [98, 100], and achieve superior performance over other recent methods. Dilated convolutions are exploited to learn the residual image for denoising in [117].

5.3 Method

We first introduce the denoising network utilized in our framework, and then explain the relationship between the image denoising module and the module for high-level vision tasks in detail.

5.3.1 Denoising Network

We propose a convolutional neural network for image denoising, which takes a noisy image as input and outputs the reconstructed image. This network conducts feature contraction and expansion through downsampling and upsampling operations, respectively. Each pair of downsampling and upsampling operations brings the feature representation into a new spatial scale, so

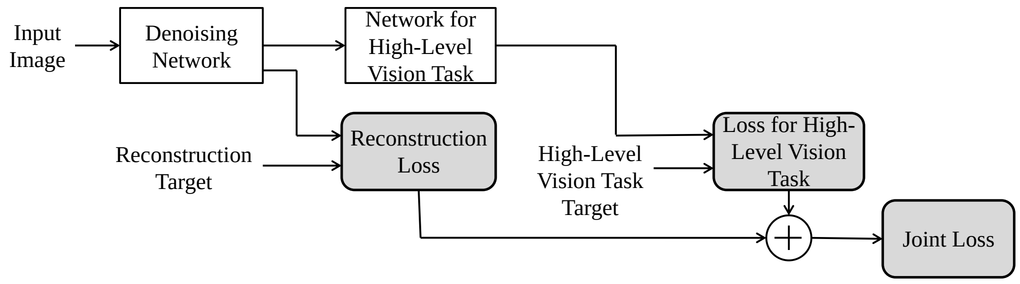

Input Image
Denoising Network
Network for High-Level Vision Task
Reconstruction Target
Reconstruction Loss
High-Level Vision Task Target
Loss for High-Level Vision Task
Joint Loss

Figure 5.3: Overview of our proposed cascaded network.

that the whole network can process information on different scales.

Specifically, on each scale, the input is encoded after downsampling the features from the previous scale. After feature encoding and decoding possibly with features on the next scale, the output is upsampled and fused with the feature on the previous scale. Such pairs of downsampling and upsampling steps can be nested to build deeper networks with more spatial scales of feature representation, which generally leads to better restoration performance. Considering the tradeoff between computation cost and restoration accuracy, we choose three scales for the denoising network in our experiments, while this framework can be easily extended for more scales.

These operations together are designed to learn the residual between the input and the target output and recover as many details as possible, so we use a long-distance skip connection to sum the output of these operations and the input image, in order to generate the reconstructed image. The overview is in Figure 5.2 (a). Each module in this network will be elaborated on as follows.

Feature Encoding: We design one feature encoding module on each scale, which is one convolutional layer plus one residual block as in [118]. The architecture is displayed in Figure 5.2 (b). Note that each convolutional layer is immediately followed by spatial batch normalization and a ReLU neuron. From top to down, the four convolutional layers have 128, 32, 32 and 128 kernels of size $3 \times 3, 1 \times 1, 3 \times 3$ and 1×1, respectively. The output of the first convolutional layer is passed through a skip connection for element-wise sum with the output of the last convolutional layer.

Feature Decoding: The feature decoding module is designed for fusing information from two adjacent scales. Two fusion schemes are tested: (1) concatenation of features on these two scales; (2) element-wise sum of them.

Both schemes obtain similar denoising performance. Thus we choose the first scheme to accommodate feature representations of different channel numbers from two scales. We use a structure similar to that of the feature encoding module except that the numbers of kernels in the four convolutional layers are 256, 64, 64 and 256. Its architecture is in Figure 5.2(c).

Feature Downsampling & Upsampling: Downsampling operations are adopted multiple times to progressively increase the receptive field of the following convolution kernels and to reduce the computation cost by decreasing the feature map size. The larger receptive field enables the kernels to incorporate larger spatial context for denoising. We use two as both the downsampling factor and the upsampling factor, and try two schemes for downsampling in the experiments: (1) max pooling with stride of two; (2) conducting convolutions with stride of two. Both schemes achieve similar denoising performance in practice, so we use the second scheme in the rest of the experiments for computation efficiency. Upsampling operations are implemented by deconvolution with 4×4 kernels, which aim to expand the feature map to the same spatial size as the previous scale.

Since all the operations in our proposed denoising network are spatially invariant, it has the merit of handling input images of arbitrary size.

5.3.2 When Image Denoising Meets High-Level Vision Tasks

We propose a robust deep architecture processing a noisy image input, via cascading a network for denoising and the other for high-level vision task, aiming to simultaneously:

1. reconstruct visually pleasing results guided by the high-level vision information, as the output of the denoising network;

2. attain sufficiently good accuracy across various high-level vision tasks, when trained for only one high-level vision task.

The overview of the proposed cascaded network is displayed in Figure 5.3. Specifically, given a noisy input image, the denoising network is first applied, and the denoised result is then fed into the following network for high-level vision task, which generates the high-level vision task output.

Training Strategy: First we initialize the network for high-level vision task from a network that is well-trained in the noiseless setting. We train the cascade of two networks in an end-to-end manner while fixing the weights in the network for high-level vision task. Only the weights in the denoising network are updated by the error back-propagated from the following network for high-level vision task, which is similar to minimizing the perceptual loss for image super-resolution [119]. The reason for adopting such a training strategy is to make the trained denoising network robust enough without losing the generality for various high-level vision tasks. More specifically, our denoising module trained for one high-level vision task can be directly plugged into other high-level tasks without fine-tuning either the denoiser or the high-level network. Our approach not only facilitates the training effort when applying the denoiser to different high-level tasks while keeping the high-level vision network performing consistently for noisy and noise-free images, but also enables the denoising network to produce high-quality perceptual and semantically faithful results.

Loss: The reconstruction loss of the denoising network is the mean squared error (MSE) between the denoising network output and the noiseless image. The losses of the classification network and the segmentation network both are the cross-entropy loss between the predicted label and the ground truth label. The joint loss is defined as the weighted sum of the reconstruction loss and the loss for high-level vision task, which can be represented as

$$L(F(x), y) = L_D(F_D(x), \tilde{x}) + \lambda L_H(F_H(F_D(x)), y), \tag{5.1}$$

where x is the noisy input image, $\tilde{x}$ is the noiseless image and y is the ground truth label of high-level vision task. F_D, F_H and F denote the denoising network, the network of high-level vision task and the whole cascaded network, respectively. L_D, L_H represent the losses of the denoising network and the high-level vision task network, respectively, while L is the joint loss, as illustrated in Figure 5.3. λ is the weight for balancing the losses L_D and L_H.

Table 5.1: Color image denoising results (PSNR) of different methods on Kodak dataset. The best result is shown in bold.

	$\sigma = 25$					$\sigma = 35$				$\sigma = 50$				
Image	**CBM3D**	**TNRD**	**MCWNNM**	**DnCNN**	**Proposed**	**CBM3D**	**MCWNNM**	**DnCNN**	**Proposed**	**CBM3D**	**TNRD**	**MCWNNM**	**DnCNN**	**Proposed**
01	29.13	27.21	28.66	29.75	**29.76**	27.31	26.93	28.10	**28.11**	25.86	24.46	25.28	26.52	**26.55**
02	32.44	31.44	31.92	32.97	**33.00**	31.07	30.62	31.65	**31.75**	29.84	29.12	29.27	30.44	**30.54**
03	34.54	32.73	34.05	34.97	**35.12**	32.62	32.27	33.37	**33.58**	31.34	29.95	30.52	31.76	**31.99**
04	32.67	31.16	32.42	32.94	**33.01**	31.02	30.92	31.51	**31.59**	29.92	28.65	29.37	30.12	**30.22**
05	29.73	27.81	29.37	30.53	**30.55**	27.61	27.53	28.66	**28.72**	25.92	24.37	25.60	26.77	**26.87**
06	30.59	28.52	30.18	31.05	**31.08**	28.78	28.44	29.37	**29.45**	27.34	25.62	26.70	27.74	**27.85**
07	33.66	31.90	33.36	34.42	**34.47**	31.64	31.53	32.60	**32.70**	29.99	28.24	29.51	30.67	**30.82**
08	29.88	27.38	29.39	30.30	**30.37**	27.82	27.67	28.53	**28.64**	26.23	23.93	25.86	26.65	**26.84**
09	34.06	32.21	33.42	34.59	**34.63**	32.28	31.76	33.06	**33.11**	30.86	28.78	30.00	31.42	**31.53**
10	33.82	31.91	33.23	34.33	**34.38**	31.97	31.51	32.74	**32.83**	30.48	28.78	29.63	31.03	**31.17**
11	31.25	29.51	30.62	31.82	**31.84**	29.53	29.04	30.23	**30.29**	28.00	26.75	27.41	28.67	**28.76**
12	33.76	32.17	33.02	34.12	**34.18**	32.24	31.52	32.73	**32.83**	30.98	29.70	30.00	31.32	**31.47**
13	27.64	25.52	27.19	**28.26**	28.24	25.70	25.40	26.46	**26.47**	24.03	22.54	23.70	24.73	**24.76**
14	30.03	28.50	29.67	30.79	**30.80**	28.24	28.05	29.17	**29.20**	26.74	25.67	26.43	27.57	**27.63**
15	33.08	31.62	32.69	33.32	**33.35**	31.47	31.15	31.89	**31.96**	30.32	29.07	29.59	30.50	**30.59**
16	32.33	30.36	31.79	32.69	**32.74**	30.64	30.15	31.16	**31.23**	29.36	27.82	28.53	29.68	**29.78**
17	32.93	31.20	32.39	**33.53**	33.50	30.64	30.75	31.96	**31.98**	29.36	28.07	28.98	30.33	**30.40**
18	29.83	28.00	29.46	30.40	**30.46**	28.00	27.70	28.72	**28.79**	26.41	25.06	25.94	27.03	**27.14**
19	31.78	30.01	31.29	**32.23**	32.30	30.19	29.86	30.80	**30.88**	29.06	27.30	28.44	29.34	**29.49**
20	33.45	32.00	32.78	34.15	**34.29**	31.84	31.32	32.73	**32.91**	30.51	29.24	29.79	31.28	**31.53**
21	30.99	29.09	30.55	31.61	**31.63**	29.17	28.86	29.94	**29.98**	27.61	26.09	27.13	28.27	**28.34**
22	30.93	29.60	30.48	**31.41**	31.38	29.36	28.93	29.94	**29.95**	28.09	27.14	27.47	28.54	**28.58**
23	34.79	33.68	34.45	35.36	**35.40**	33.09	32.79	33.86	**33.89**	31.75	30.53	30.96	32.18	**32.30**
24	30.09	28.17	29.93	**30.79**	30.77	28.19	28.17	28.98	**29.03**	26.62	24.92	26.37	27.18	**27.30**
Average	31.81	30.08	31.35	32.35	**32.39**	30.04	29.70	30.76	**30.83**	28.62	27.17	28.02	29.16	**29.27**

5.4 Experiments

5.4.1 Image Denoising

Our proposed denoising network takes RGB images as input, and outputs the reconstructed images directly. We add independent and identically distributed Gaussian noise with zero mean to the original image as the noisy input image during training. We use the same training set as in [120]. The loss of training is equivalent to Eq. (5.1) as $\lambda = 0$. We use SGD with a batch size of 32, and the input patches are 48×48 pixels. The initial learning rate is set as 10^{-4} and is divided by 10 after every 500,000 iterations. The training is terminated after 1,500,000 iterations. We train a different denoising network for each noise level in our experiment.

We compare our denoising network with several state-of-the-art color image denoising approaches on various noise levels: $\sigma = 25, 35$ and 50. We evaluate their denoising performance over the widely used Kodak dataset,[2] which consists of 24 color images. Table 5.1 shows the peak signal-to-noise ratio (PSNR) results for CBM3D [112], TNRD [99], MCWNNM [121], DnCNN [100], and our proposed method. We do not list other methods [97, 107, 94, 117] whose average performance is wore than DnCNN. The implementation codes used are from the authors websites and the default

[2]`http://r0k.us/graphics/kodak/`

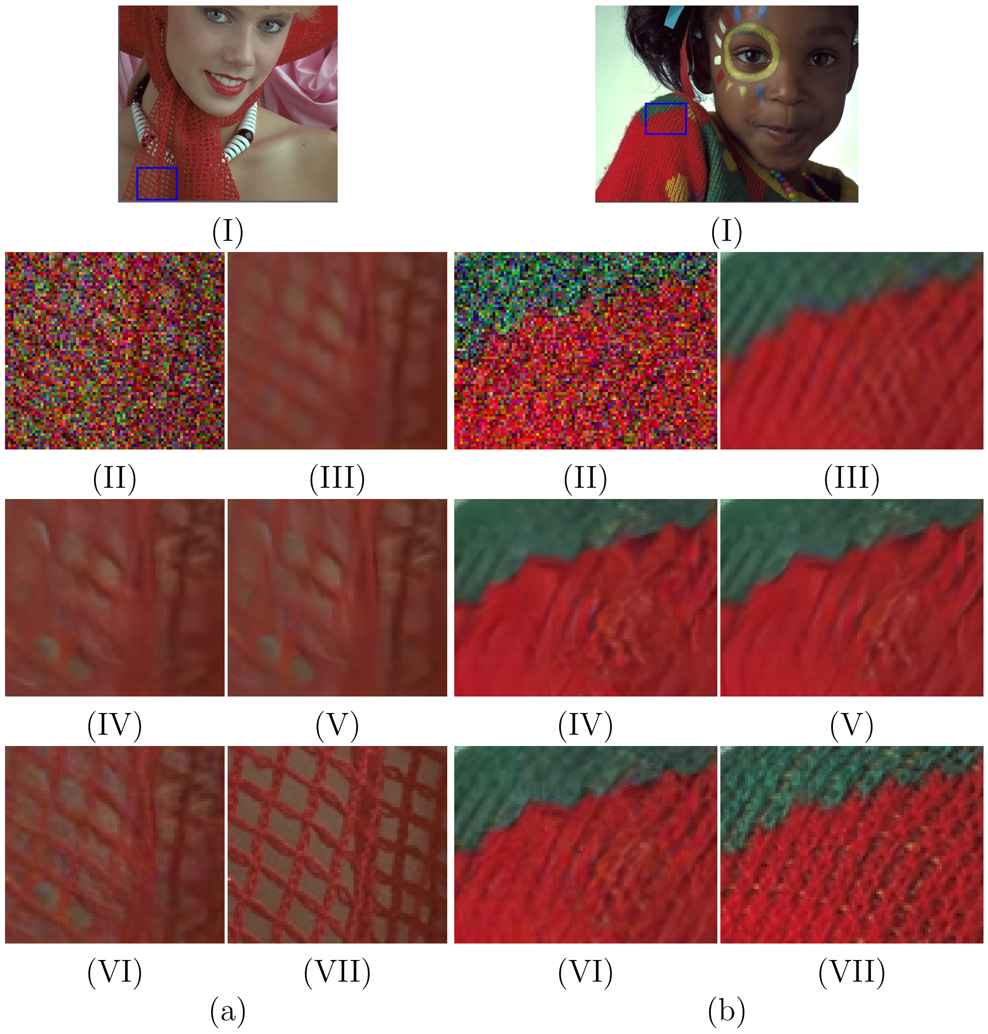


Figure 5.4: Two image denoising examples from Kodak dataset with noise level of 50 are displayed in (a) and (b). We show (I) the ground truth image and the zoom-in regions of: (II) the noisy image; (III) the denoised image by CBM3D; (IV) the denoised image by DnCNN; the denoising result of our proposed model (V) without the guidance of high-level vision information; (VI) with the guidance of high-level vision information and (VII) the ground truth.

parameter settings are adopted in our experiments.[3]

It is clear that our proposed method outperforms all the competing approaches quantitatively across different noise levels. It achieves the highest PSNR in almost every image of the Kodak dataset.

[3]For TNRD, we denoise each color channel using the grayscale image denoising implementation which is from the authors' website. TNRD for $\sigma = 35$ is not publicly available, so we do not include this case here.

5.4.2 When Image Denoising Meets High-Level Vision Tasks

We choose two high-level vision tasks as representatives in our study: image classification and semantic segmentation, which have been dominated by deep network based models. We utilize two popular VGG-based deep networks in our system for each task, respectively. *VGG-16* in [122] is employed for image classification; we select *DeepLab-LargeFOV* in [120] for semantic segmentation. We follow the preprocessing protocols (e.g. crop size, mean removal of each color channel) in [122] and [120] accordingly while training and deploying them in our experiments.

As for the cascaded network for image classification and the corresponding experiments, we train our model on ILSVRC2012 training set, and evaluate the classification accuracy on ILSVRC2012 validation set. λ is empirically set as 0.25. As for the cascaded network for image semantic segmentation and its corresponding experiments, we train our model on the augmented training set of Pascal VOC 2012 as in [120], and test on its validation set. λ is empirically set as 0.5.

High-Level Vision Information Guided Image Denoising

The typical metric used for image denoising is PSNR, which has been shown to sometimes correlate poorly with human assessment of visual quality [123]. Since PSNR depends on the reconstruction error between the denoised output and the reference image, a model trained by minimizing MSE on the image domain should always outperform a model trained by minimizing our proposed joint loss (with the guidance of high-level vision semantics) in the metric of PSNR. Therefore, we emphasize that the goal of our following experiments is not to pursue the highest PSNR, but to demonstrate the qualitative difference between the model trained with our proposed joint loss and the model trained with MSE on the image domain.

Figure 5.4 displays two image denoising examples from the Kodak dataset with noise level of 50. A visual comparison is illustrated for a zoom-in region: (II) and (III) are the denoising results using CBM3D [112], and DnCNN [100], respectively; (IV) is the proposed denoiser trained separately without the guidance of high-level vision information; (V) is the denoising result using the proposed denoising network trained jointly with a segmentation network.

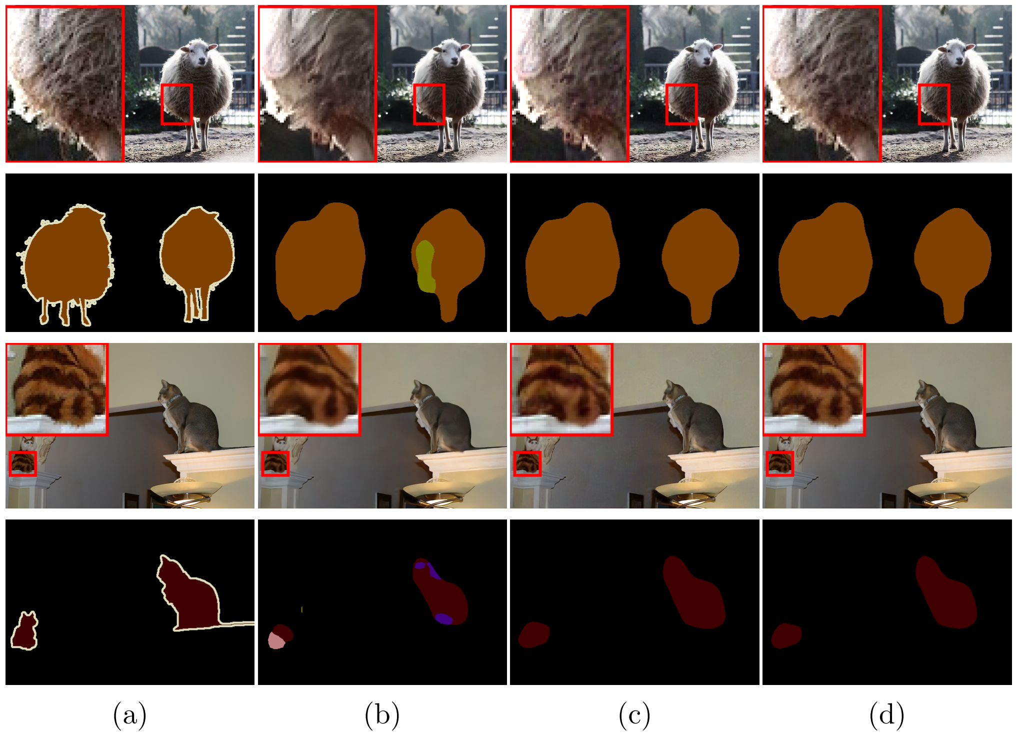


Figure 5.5: Two semantic segmentation examples from Pascal VOC 2012 validation set. From left to right: (a) the ground truth image, the denoised image using (b) the separately trained denoiser, (c) the denoiser trained with the reconstruction and segmentation joint loss, and (d) the denoiser trained with the classification network and evaluated for semantic segmentation. Their corresponding segmentation label maps are shown below. The zoom-in region which generates inaccurate segmentation in (b) is displayed in the red box.

We can find that the results using CBM3D, DnCNN and our separately trained denoiser generate oversmoothing regions, while the jointly trained denoising network is able to reconstruct the denoised image which preserves more details and textures with better visual quality.

Generality of the Denoiser for High-Level Vision Tasks

We now investigate how the image denoising can enhance the high-level vision applications, including image classification and semantic segmentation, over the ILSVRC2012 and Pascal VOC 2012 datasets, respectively. The noisy images ($\sigma = 15, 30, 45, 60$) are denoised and then fed into the VGG-based networks for high-level vision tasks. To evaluate how different denoising schemes

Table 5.2: Classification accuracy after denoising noisy image input, averaged over ILSVRC2012 validation dataset. Red is the best and blue is the second best results.

		VGG	CBM3D + VGG	Separate + VGG	Joint Training	Joint Training (Cross-Task)
σ=15	Top-1	62.4	68.2	68.3	69.9	69.8
	Top-5	84.2	88.8	88.7	89.5	89.4
σ=30	Top-1	44.4	62.3	62.7	67.0	66.4
	Top-5	68.9	84.8	84.9	87.6	87.2
σ=45	Top-1	24.3	55.2	54.6	63.0	62.0
	Top-5	46.1	79.4	78.8	84.6	84.0
σ=60	Top-1	11.4	50.0	50.1	59.2	57.0
	Top-5	26.3	74.2	74.5	81.8	80.2

contribute to the performance of high-level vision tasks, we experiment with the following cases:

- Noisy images are directly fed into the high-level vision network, termed as *VGG*. This approach serves as the baseline.
- Noisy images are first denoised by CBM3D, and then fed into the high-level vision network, termed as *CBM3D+VGG*.
- Noisy images are denoised via the separately trained denoising network, and then fed into the high-level vision network, termed as *Separate+VGG*.
- Our proposed approach: noisy images are processed by the cascade of these two networks, which is trained using the joint loss, termed as *Joint Training*.
- A denoising network is trained with the classification network in our proposed approach, but then is connected to the segmentation network and evaluated for the task of semantic segmentation, or vice versa. This is to validate the generality of our denoiser for various high-level tasks, termed as *Joint Training (Cross-Task)*.

Note that the weights in the high-level vision network are initialized from a well-trained network under the noiseless setting and not updated during training in our experiments.

Table 5.3: Segmentation results (mIoU) after denoising noisy image input, averaged over Pascal VOC 2012 validation dataset. Red is the best and blue is the second best results.

	VGG	CBM3D + VGG	Separate + VGG	Joint Training	Joint Training (Cross-Task)
σ=15	56.78	59.58	58.70	60.46	60.41
σ=30	43.43	55.29	54.13	57.86	56.29
σ=45	27.99	50.69	49.51	54.83	54.01
σ=60	14.94	46.56	46.59	52.02	51.82

Table 5.2 and Table 5.3 list the performance of high-level vision tasks, i.e., top-1 and top-5 accuracy for classification and mean intersection-over-union (IoU) without conditional random field (CRF) postprocessing for semantic segmentation. We notice that the baseline VGG approach obtains much lower accuracy than all the other cases, which shows the necessity of image denoising as a preprocessing step for high-level vision tasks on noisy data. When we only apply denoising without considering high-level semantics (e.g., in CBM3D+VGG and Separate+VGG), it also fails to achieve high accuracy due to the artifacts introduced by the denoisers. The proposed Joint Training approach achieves sufficiently high accuracy across various noise levels.

As for the case of Joint Training (Cross-Task), first we train the denoising network jointly with the segmentation network and then connect this denoiser to the classification network. As shown in Table 5.2, its accuracy remarkably outperforms the cascade of a separately trained denoising network and a classification network (i.e., Separate+VGG), and is comparable to our proposed model dedicatedly trained for classification (Joint Training). In addition, we use the denoising network jointly trained with the classification network to connect the segmentation network. Its mean IoU is much better than Separate+VGG in Table 5.3. These two experiments show that the high-level semantics of different tasks are universal in terms of low-level vision tasks, which is in line with intuition, and the denoiser trained in our method has the generality for various high-level tasks.

Figure 5.5 displays two visual examples of how the data-driven denoising can enhance the semantic segmentation performance. It is observed that the segmentation result of the denoised image from the separately trained denoising network has lower accuracy compared to those using the joint loss and the joint loss (cross-task), while the zoom-in region of its denoised image for inaccurate segmentation in Figure 5.5 (b) contains oversmoothing artifacts. In contrast, both the Joint Training and Joint Training (Cross-Task) approaches achieve finer segmentation result and produce more visually pleasing denoised outputs simultaneously.

5.5 Conclusion

Exploring the connection between low-level vision and high-level semantic tasks is of great practical value in various applications of computer vision. In this chapter, we tackle this challenge in a simple yet efficient way by allowing the high-level semantic information to flow back to the low-level vision part, which achieves superior performance in both image denoising and various high-level vision tasks. In our method, the denoiser trained for one high-level task has the generality to other high-level vision tasks. Overall, it provides a feasible and robust solution in a deep learning fashion to real world problems. For future work, we will explore the performance of the denoising network when dealing with other types of noise.

Chapter 6

NON-LOCAL RECURRENT NETWORK FOR IMAGE RESTORATION

6.1 Introduction

Image restoration is an ill-posed inverse problem that aims at estimating the underlying image from its degraded measurements. Depending on the type of degradation, image restoration can be categorized into different sub-problems, e.g., image denoising and image super-resolution. The key to successful restoration typically relies on the design of an effective regularizer based on image priors. Both local and non-local image priors have been extensively exploited in the past. Considering image denoising as an example, local image properties such as Gaussian filtering and total variation based methods [124] are widely used in early studies. Later on, the notion of self-similarity in natural images draws more attention and it has been exploited by non-local-based methods, e.g., non-local means [30], collaborative filtering [32], joint sparsity [92], and low-rank modeling [94]. These non-local methods are shown to be effective in capturing the correlation among non-local patches to improve the restoration quality.

While non-local self-similarity has been extensively studied in the literature, approaches for capturing this intrinsic property with deep networks are little explored. Recent convolutional neural networks (CNNs) for image restoration [18, 57, 98, 100] achieve impressive performance over conventional approaches but do not explicitly use self-similarity properties in images. To rectify this weakness, a few studies [125, 126] apply block matching to patches before feeding them into CNNs. Nevertheless, the block matching step is isolated and thus not jointly trained with image restoration networks.

In this chapter, we present the first attempt to incorporate non-local operations in CNN for image restoration, and propose a non-local recurrent network (NLRN) as an efficient yet effective network with non-local module.

First, we design a non-local module to produce reliable feature correlation for self-similarity measurement given severely degraded images, which can be flexibly integrated into existing deep networks while embracing the benefit of end-to-end learning. For high parameter efficiency without compromising restoration quality, we deploy a recurrent neural network (RNN) framework similar to [59, 63, 127] such that operations with shared weights are applied recursively. Second, we carefully study the behavior of non-local operation in deep feature space and find that limiting the neighborhood of correlation computation improves its robustness to degraded images. The confined neighborhood helps concentrate the computation on relevant features in the spatial vicinity and disregard noisy features, which is in line with conventional image restoration approaches [32, 94]. In addition, we allow message passing of non-local operations between adjacent recurrent states of RNN. Such inter-state flow of feature correlation facilitates more robust correlation estimation. By combining the non-local operation with typical convolutions, our NLRN can effectively capture and employ both local and non-local image properties for image restoration.

It is noteworthy that recent work has adopted similar ideas on video classification [2]. However, our method significantly differs from it in the following aspects. For each location, we measure the feature correlation of each location only in its neighborhood, rather than throughout the whole image as in [2]. In our experiments, we show that deep features useful for computing non-local priors are more likely to reside in neighboring regions. A larger neighborhood (the whole image as one extreme) can lead to inaccurate correlation estimation over degraded measurements. In addition, our method fully exploits the advantage of RNN architecture: the correlation information is propagated among adjacent recurrent states to increase the robustness of correlation estimation to degradations of various degrees. Moreover, our non-local module is flexible to handle inputs of various sizes, while the module in [2] handles inputs of fixed sizes only.

We introduce NLRN by first relating our proposed model to other classic and existing non-local image restoration approaches in a unified framework. We thoroughly analyze the non-local module and recurrent architecture in our NLRN via extensive ablation studies. We provide a comprehensive comparison with recent competitors, in which our NLRN achieves state-of-the-art performance in image denoising and super-resolution over several benchmark datasets, demonstrating the superiority of the non-local operation with recurrent architecture for image restoration [128].

6.2 Related Work

Image self-similarity as an important image characteristic has been used in a number of non-local-based image restoration approaches. The early works include bilateral filtering [129] and non-local means [30] for image denoising. Recent approaches exploit image self-similarity by imposing sparsity [92, 21]. Alternatively, similar image patches are modeled with low-rankness [94], or by collaborative Wiener filtering [32]. Neighborhood embedding is a common approach for image super-resolution [51, 27], in which each image patch is approximated by multiple similar patches in a manifold. Self-example based image super-resolution approaches [10, 11] exploit the local self-similarity assumption, and extract LR-HR exemplar pairs merely from the low-resolution image across different scales to predict the high-resolution image.

Deep neural networks have been prevalent for image restoration. The pioneering works include a multilayer perceptron for image denoising [97] and a three-layer CNN for image super-resolution [18]. Deconvolution is adopted to save computation cost and accelerate inference speed [60, 61]. Very deep CNNs are designed to boost SR accuracy in [57, 62, 130, 131]. Dense connections among various residual blocks are included in [132]. Similarly CNN based methods are developed for image denoising in [98, 100, 117]. Block matching as a preprocessing step is cascaded with CNNs for image denoising [125, 126]. Besides CNNs, RNNs have also been applied for image restoration while enjoying the high parameter efficiency [59, 63, 127, 79]. Deep neural networks are designed to exploit both the spatial relation within a single frame and the temporal relation among consecutive frames for video SR in [133].

In addition to image restoration, feature correlations are widely exploited along with neural networks in many other areas, including graphical models [134, 135, 136], relational reasoning [137], machine translation [138, 139] and so on. We do not elaborate on them here due to the limitation of space.

6.3 Non-Local Operations for Image Restoration

In this section, we first present a unified framework of non-local operations used for image restoration methods, e.g., collaborative filtering [32], non-local means [30], and low-rank modeling [94], and we discuss the relations between them. We then present the proposed non-local operation module.

6.3.1 A General Framework

In general, a non-local operation takes a multi-channel input $\boldsymbol{X} \in \mathbb{R}^{N\times m}$ as the image feature, and generates output feature $\boldsymbol{Z} \in \mathbb{R}^{N\times k}$. Here N and m denote the number of image pixels and data channels, respectively. We propose a general framework with the following formulation:

$$\boldsymbol{Z} = \mathrm{diag}\{\delta(\boldsymbol{X})\}^{-1}\,\Phi(\boldsymbol{X})\,\boldsymbol{G}(\boldsymbol{X})\,. \tag{6.1}$$

Here, $\Phi(\boldsymbol{X}) \in \mathbb{R}^{N\times N}$ is the non-local correlation matrix, and $\boldsymbol{G}(\boldsymbol{X}) \in \mathbb{R}^{N\times k}$ is the multi-channel non-local transform. Each row vector $\boldsymbol{X}_i$ denotes the local features in location i. $\Phi(\boldsymbol{X})_i^j$ represents the relationship between the $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$, and each row vector $\boldsymbol{G}(\boldsymbol{X})_j$ is the embedding of $\boldsymbol{X}_j$.[1] The diagonal matrix $\mathrm{diag}\{\delta(\boldsymbol{X})\} \in \mathbb{R}^{N\times N}$ normalizes the output at each i-th pixel with normalization factor $\delta_i(\boldsymbol{X})$.

6.3.2 Classic Methods

The proposed framework works with various classic non-local methods for image restoration, including methods based on low-rankness [94], collaborative filtering [32], joint sparsity [92], as well as non-local mean filtering [30].

Block matching (BM) is a commonly used approach for exploiting non-local image structures in conventional methods [94, 32, 92]. A $q \times q$ spatial neighborhood is set to be centered at each location i, and $\boldsymbol{X}_i$ reduces to the image patch centered at i. BM selects the K_i most similar patches ($K_i \ll q^2$) from this neighborhood, which are used jointly to restore $\boldsymbol{X}_i$. Under the

[1]In our analysis, if $\boldsymbol{A}$ is a matrix, $\boldsymbol{A}_i$, $\boldsymbol{A}^j$, and $\boldsymbol{A}_i^j$ denote its i-th row, j-th column, and the element at the i-th row and j-th column, respectively.

proposed non-local framework, these methods can be represented as

$$\boldsymbol{Z}_i = \frac{1}{\delta_i(\boldsymbol{X})} \sum\nolimits_{j \in \mathbb{C}_i} \Phi(\boldsymbol{X})_i^j \, \boldsymbol{G}(\boldsymbol{X})_j \,, \ \ \forall i \,. \tag{6.2}$$

Here $\delta_i(\boldsymbol{X}) = \sum_{j \in \mathbb{C}_i} \Phi(\boldsymbol{X})_i^j$ and $\mathbb{C}_i$ denotes the set of indices of the K_i selected patches. Thus, each row $\Phi(\boldsymbol{X})_i$ has only K_i non-zero entries. The embedding $\boldsymbol{G}(\boldsymbol{X})$ and the non-zero elements vary for non-local methods based on different models. For example, in WNNM [94], $\sum_{j \in \mathbb{C}_i} \Phi(\boldsymbol{X})_i^j \, \boldsymbol{G}(\boldsymbol{X})_j$ corresponds to the projection of $\boldsymbol{X}_i$ onto the group-specific subspace as a function of the selected patches. Specifically, the subspace for calculating $\boldsymbol{Z}_i$ is spanned by the eigenvectors $\boldsymbol{U}_i$ of $\boldsymbol{X}_{\mathbb{C}_i}^T \boldsymbol{X}_{\mathbb{C}_i}$. Thus $\boldsymbol{Z}_i = \boldsymbol{X}_{\mathbb{C}_i} \boldsymbol{U}_i \text{diag}\{\sigma\} \boldsymbol{U}_i^T$, where $\text{diag}\{\sigma\}$ is obtained by applying the shrinkage function associated with the weighted nuclear norm [94] to the eigenvalues of $\boldsymbol{X}_{\mathbb{C}_i}^T \boldsymbol{X}_{\mathbb{C}_i}$. We show the generalization about more classic non-local image restoration methods in Section 6.3.3.

Except for the *hard* block matching, other methods, e.g., the non-local means algorithm [30], apply *soft* block matching by calculating the correlation between the reference patch and each patch in the neighborhood. Each element $\Phi(\boldsymbol{X})_i^j$ is determined only by each $\{\boldsymbol{X}_i, \boldsymbol{X}_j\}$ pair, so $\Phi(\boldsymbol{X})_i^j = \phi(\boldsymbol{X}_i, \boldsymbol{X}_j)$, where $\phi(\,\cdot\,)$ is determined by the distance metric. In [30], weighted Euclidean distance with Gaussian kernel is applied as the metric, such that $\phi(\boldsymbol{X}_i, \boldsymbol{X}_j) = \exp\{-\left\|\boldsymbol{X}_i - \boldsymbol{X}_j\right\|_{2,a}^2 / h^2\}$. Besides, identity mapping is directly used as the embedding in [30], i.e., $\boldsymbol{G}(\boldsymbol{X})_j = \boldsymbol{X}_j$. In this case, the non-local framework in (6.1) reduces to

$$\boldsymbol{Z}_i = \frac{1}{\delta_i(\boldsymbol{X})} \sum\nolimits_{j \in \mathbb{S}_i} \exp\{-\frac{\left\|\boldsymbol{X}_i - \boldsymbol{X}_j\right\|_{2,a}^2}{h^2}\} \boldsymbol{X}_j \,, \ \ \forall i, \tag{6.3}$$

where $\delta_i(\boldsymbol{X}) = \sum_{j \in \mathbb{S}_i} \exp\{-\left\|\boldsymbol{X}_i - \boldsymbol{X}_j\right\|_{2,a}^2 / h^2\}$ and $\mathbb{S}_i$ is the set of indices in the neighborhood of $\boldsymbol{X}_i$. Note that both a and h are constants, denoting the standard deviation of Gaussian kernel, and the degree of filtering, respectively [30]. It is noteworthy that the cardinality of $\mathbb{S}_i$ for soft BM is much larger than that of $\mathbb{C}_i$ for hard BM, which gives more flexibility of using feature correlations between neighboring locations.

The conventional non-local methods suffer from the drawback that parameters are either fixed [30] or obtained by suboptimal approaches [32, 92, 94], e.g., the parameters of WNNM are learned based on the low-rankness assumption, which is suboptimal as the ultimate objective is to minimize the image reconstruction error.

6.3.3 Extension of the General Framework to Other Classic Non-Local Methods

Besides the extension to WNNM and non-local means, which are discussed in Section 6.3.1, we show the proposed non-local framework (6.1) can be extended to collaborative filtering methods, e.g., BM3D algorithm [32], as well as joint sparsity based methods, e.g., LSSC algorithm [92]. We follow the same notations in Section 6.3.1. Both BM3D and LSSC apply block matching (BM) first before processing, and form N groups of similar patches into data matrices. The index set of the matched patches for the i-th reference patch is denoted as $\mathbb{C}_i$. The group of matched patches for the i-th reference patch is denoted as $\boldsymbol{X}_{\mathbb{C}_i}$.

Similar to WNNM [94], BM3D [32] also applies BM first to group similar patches based on their Euclidean distances. The matched patches are then processed via Wiener filtering [32], and the denoised results of the i-th group of patches are

$$\boldsymbol{Z}_{\mathbb{C}_i} = \tau^{-1}(\mathrm{diag}(\omega)\tau(\boldsymbol{X}_{\mathbb{C}_i})). \tag{6.4}$$

Here $\tau(\cdot)$ and $\tau^{-1}(\cdot)$ denote the forward and backward Wiener filtering applied to the groups of matched patches, respectively. The diagonal matrix $\mathrm{diag}(\omega)$ is formed by the empirical Wiener coefficients ω. BM3D applies data pre-cleaning, using discrete cosine transform (DCT), to estimate the original patch, and calculate the estimate of ω [32]. Since calculating $\boldsymbol{Z}_{\mathbb{C}_i}$ in (6.4) involves only linear filtering, it can also be generalized using the proposed non-local framework as (6.2). Unlike the extension to WNNM, here $\sum_{j\in\mathbb{C}_i} \Phi(\boldsymbol{X})_i^j \boldsymbol{G}(\boldsymbol{X})_j$ corresponds to the denoised results via Wiener filtering as shown in (6.4), of the i-th group of matched patches.

Different from BM3D and WNNM, LSSC learns a common dictionary $\boldsymbol{D}$ for all image patches and imposes joint sparsity [92] on each data matrix of matched patches $\boldsymbol{X}_{\mathbb{C}_i}$, so that the correlation of the matched patches are exploited by enforcing the same support of their sparse codes. Thus, the joint sparse coding in LSSC [92] becomes

$$\hat{\boldsymbol{A}}_i = \text{argmin}_{\boldsymbol{A}_i} \left\| \boldsymbol{A}_i \right\|_{0,\infty} \quad s.t. \quad \left\| \boldsymbol{X}_{\mathbb{C}_i}^T - \boldsymbol{D}\boldsymbol{A}_i \right\|_F^2 \leq \epsilon \left| \mathbb{C}_i \right| , \ \ \forall i, \tag{6.5}$$

where the $(0, \infty)$ "norm" $\|\cdot\|_{0,\infty}$ counts the number of non-zero columns of each sparse code matrix $\boldsymbol{A}_i$ [92], and $|\mathbb{C}_i|$ is the cardinality of $\mathbb{C}_i$. The coefficient ϵ is a constant, which is used to upper bound the sparse modeling errors. In general, the solution to (6.5) is NP-hard. To simplify the discussion, we assume the dictionary to be unitary (which reduces the sparse coding problem to the transform-model sparse coding [21]), i.e., $\boldsymbol{D}^T\boldsymbol{D} = \boldsymbol{I}$ and $\boldsymbol{D} \in \mathbb{R}^{k\times k}$. Thus there exists a corresponding shrinkage function $\eta(\cdot)$ for imposing joint sparsity on the sparse codes [92, 116], such that the denoised estimates of the i-th patch group can be obtained as $\boldsymbol{Z}_{\mathbb{C}_i} = \hat{\boldsymbol{A}}_i^T \boldsymbol{D}^T = \eta(\, \boldsymbol{X}_{\mathbb{C}_i} \boldsymbol{D}\,)\, \boldsymbol{D}^T$. Though joint sparse coding projects all data onto a union of subspaces [92, 140, 21] which is a non-linear operation in general, each data matrix $\boldsymbol{X}_{\mathbb{C}_i}$ is projected onto one particular subspace spanned by the selected atoms corresponding to the non-zero columns in $\hat{\boldsymbol{A}}_i$, which is locally linear. For the i-th group of patches, such a subspace projection corresponds to $\sum_{j\in\mathbb{C}_i} \Phi(\boldsymbol{X})_i^j\, \boldsymbol{G}(\boldsymbol{X})_j$ in the proposed general framework.

6.3.4 The Proposed Non-Local Module

Based on the general non-local framework in (6.1), we propose another soft block matching approach and apply the Euclidean distance with linearly embedded Gaussian kernel [2] as the distance metric. The linear embeddings are defined as follows:

$$\Phi(\boldsymbol{X})_i^j = \phi(\boldsymbol{X}_i, \boldsymbol{X}_j) = \exp\{\theta(\boldsymbol{X}_i)\psi(\boldsymbol{X}_j)^T\} \,, \ \ \forall i, j \,, \tag{6.6}$$

$$\theta(\boldsymbol{X}_i) = \boldsymbol{X}_i\boldsymbol{W}_\theta, \ \ \psi(\boldsymbol{X}_i) = \boldsymbol{X}_i\boldsymbol{W}_\psi, \ \ \boldsymbol{G}(\boldsymbol{X})_i = \boldsymbol{X}_i\boldsymbol{W}_g \,, \ \ \forall i \,. \tag{6.7}$$

The embedding transforms $\boldsymbol{W}_\theta$, $\boldsymbol{W}_\phi$, and $\boldsymbol{W}_g$ are all learnable and have the shape of $m \times l$, $m \times l$, $m \times m$, respectively. Thus, the proposed non-local operation can be written as

$$\boldsymbol{Z}_i = \frac{1}{\delta_i(\boldsymbol{X})} \sum\nolimits_{j\in\mathbb{S}_i} \exp\left\{\boldsymbol{X}_i\boldsymbol{W}_\theta\boldsymbol{W}_\psi^T\boldsymbol{X}_j^T\right\} \boldsymbol{X}_i\boldsymbol{W}_g \,, \ \ \forall i \,, \tag{6.8}$$

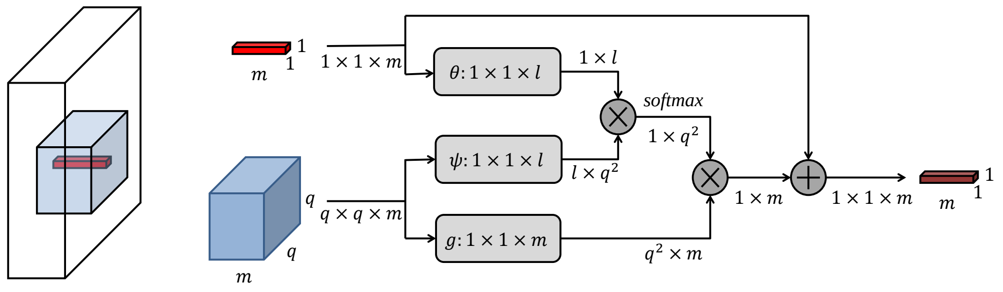


Figure 6.1: An illustration of our non-local module working on a single location. The white tensor denotes the deep feature representation of an entire image. The red fiber is the features of this location and the blue tensor denotes the features in its neighborhood. θ, ψ and g are implemented by 1×1 convolution followed by reshaping operations.

where $\delta_i(\boldsymbol{X}) = \sum_{j \in \mathbb{S}_i} \phi(\boldsymbol{X}_i, \boldsymbol{X}_j)$. Similar to [30], to obtain $\boldsymbol{Z}_i$, we evaluate the correlation between $\boldsymbol{X}_i$ and each $\boldsymbol{X}_j$ in the neighborhood $\mathbb{S}_i$. More choices of $\phi(\boldsymbol{X}_i, \boldsymbol{X}_j)$ are discussed in Section 6.5.

The proposed non-local operation can be implemented by common differentiable operations, and thus can be jointly learned when incorporated into a neural network. We wrap it as a non-local module by adding a skip connection, as shown in Figure 6.1, since the skip connection enables us to insert a non-local module into any pre-trained model, while maintaining its initial behavior by initializing $\boldsymbol{W}_g$ as zero. Such a module introduces only a limited number of parameters since θ, ψ and g are 1×1 convolutions and $m = 128, l = 64$ in practice. The output of this module on each location only depends on its $q \times q$ neighborhood, so this operation can work on inputs of various sizes.

6.3.5 Relation to Other Methods

Recent works have combined non-local BM and neural networks for image restoration [126, 125, 2]. Lefkimmiatis [125] proposed to first apply BM to noisy image patches. The hard BM results are used to group patch features, and a CNN conducts a trainable collaborative filtering over the matched patches. Qiao et al. [126] combined similar non-local BM with TNRD networks [99] for image denoising. However, as conventional methods [32, 92, 94], these works [125, 126] conduct hard BM directly over degraded input patches, which may be inaccurate over severely degraded images. In contrast, our proposed non-local operation as soft BM is applied on learned deep feature representations that are more robust to degradation. Furthermore, the matching results in [125] are isolated from the neural network, similar to the conventional approaches, whereas the proposed non-local module is trained jointly with the entire network in an end-to-end manner.

Wang et al. [2] used similar approaches to add non-local operations into neural networks for high-level vision tasks. However, unlike our approach, Wang et al. [2] calculated feature correlations throughout the whole image. which is equivalent to enlarging the neighborhood to the entire image in our approach. We empirically show that increasing the neighborhood size does not always improve image restoration performance, due to the inaccuracy of correlation estimation over degraded input images. Hence it is imperative to choose a neighborhood of a proper size to achieve best performance for image restoration. In addition, the non-local operation in [2] can only handle input images of fixed size, while our module in (6.8) is flexible for various image sizes. Finally, our non-local module, when incorporated into an RNN framework, allows the flow of correlation information between adjacent states to enhance robustness against inaccurate correlation estimation. This is a new unique formulation to deal with degraded images. More details are provided next.

6.4 Non-Local Recurrent Network

In this section, we describe the RNN architecture that incorporates the non-local module to form our NLRN. We adopt the common formulation of an RNN, which consists of a set of states, namely, input state, output state and recurrent state, as well as transition functions among the states. The input, output, and recurrent states are represented as $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{s}$ respectively. At each time step t, an RNN receives an input $\boldsymbol{x}^t$, and the recurrent state and the output state of the RNN are updated recursively as follows:

$$\boldsymbol{s}^t = f_{\text{input}}(\boldsymbol{x}^t) + f_{\text{recurrent}}(\boldsymbol{s}^{t-1}), \quad \boldsymbol{y}^t = f_{\text{output}}(\boldsymbol{s}^t), \tag{6.9}$$

where f_{input}, f_{output}, and $f_{\text{recurrent}}$ are reused at every time step. In our NLRN, we set the following:

- $\boldsymbol{s}^0$ is a function of the input image $\boldsymbol{I}$.
- $\boldsymbol{x}^t = 0$, $\forall t \in \{1, \ldots, T\}$, and $f_{\text{input}}(0) = 0$.
- The output state $\boldsymbol{y}^t$ is calculated only at the time T as the final output.

We add an identity path from the very first state which helps gradient back-propagation during training [63], and a residual path of the deep feature correlation between each location and its neighborhood from the previous state. Hence, $\boldsymbol{s}^t = \{\boldsymbol{s}^t_{\text{feat}}, \boldsymbol{s}^t_{\text{corr}}\}$, and $\boldsymbol{s}^t = f_{\text{recurrent}}(\boldsymbol{s}^{t-1}, \boldsymbol{s}^0)$, $\forall t \in \{1, \ldots, T\}$, where $\boldsymbol{s}^t_{\text{feat}}$ denotes the feature map in time t and $\boldsymbol{s}^t_{\text{corr}}$ is the collection of deep feature correlation. For the transition function $f_{\text{recurrent}}$, a non-local module is first adopted and is followed by two convolutional layers, before the feature $\boldsymbol{s}^0$ is added from the identity path. The weights in the non-local module are shared across recurrent states just as convolutional layers, so our NLRN still keeps high parameter efficiency as a whole. An illustration is displayed in Figure 6.2.

It is noteworthy that inside the non-local module, the feature correlation for location i from the previous state, $\boldsymbol{s}^{t-1}_{\text{corr},i}$, is added to the estimated feature correlation in the current state before the *softmax* normalization, which enables the propagation of correlation information between adjacent states for more robust correlation estimation. The details can be found in Figure 6.3. The initial state $\boldsymbol{s}^0$ is set as the feature after a convolutional layer on the input image. f_{output} is represented by another single convolutional layer. All layers have 128 filters with 3×3 kernel size except for the non-local module. Batch normalization and ReLU activation function are performed ahead of each convolutional layer following [141]. We adopt residual learning and the output of NLRN is the residual image $\hat{I} = f_{\text{output}}(\boldsymbol{s}^T)$ when NLRN is unfolded T times. During training, the objective is to minimize the mean square error $\mathcal{L}(\hat{I}, \tilde{I}) = \frac{1}{2}||\hat{I} + I - \tilde{I}||^2$, where $\tilde{I}$ denotes the ground truth image.

Relation to Other RNN Methods

Although RNNs have been adopted for image restoration before, our NLRN is the first to incorporate non-local operations into an RNN framework with

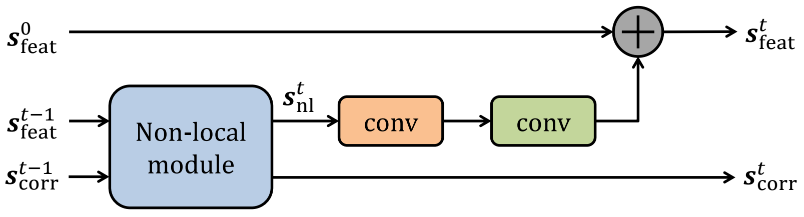


Figure 6.2: An illustration of the transition function $f_{\text{recurrent}}$ in the proposed NLRN.

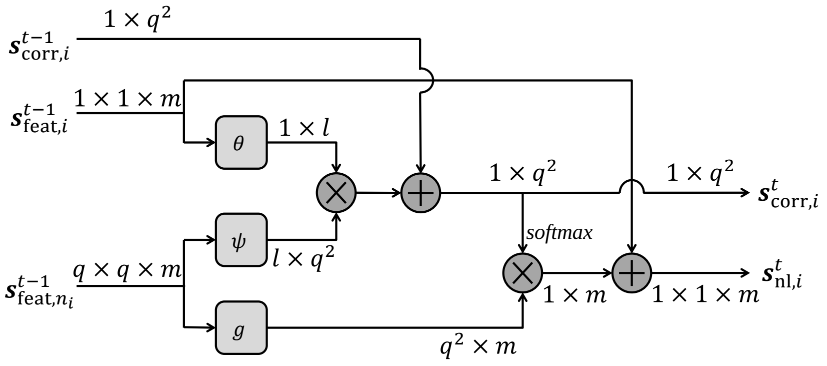


Figure 6.3: The operations for a single location i in the non-local module used in NLRN.

correlation propagation. DRCN [59] recursively applies a single convolutional layer to the input feature map multiple times without the identity path from the first state. DRRN [63] applies both the identity path and the residual path in each state, but without non-local operations, and thus there is no correlation information flow across adjacent states. MemNet [127] builds dense connections among several types of memory blocks, and weights are shared in the same type of memory blocks but are different across various types. Compared with MemNet, our NLRN has an efficient yet effective RNN structure with shallower effective depth and fewer parameters, but obtains better restoration performance, which is shown in Section 6.5 in detail.

6.5 Experiments

6.5.1 Implementation Details

Dataset

For image denoising, we adopt two different settings to fairly and comprehensively compare with recent deep learning based methods [98, 125, 100, 127]: (1) As in [99, 100, 125], we choose as the training set the combination of 200 images from the *train* set and 200 images from the *test* set in the Berkeley Segmentation Dataset (BSD) [40], and test on two popular benchmarks: Set12 and Set68 with $\sigma = 15, 25, 50$ following [100]. (2) As in [98, 127], we use as the training set the combination of 200 images from the *train* set and 100 images from the *val* set in BSD, and test on Set14 and the BSD *test* set of 200 images with $\sigma = 30, 50, 70$ following [98, 127]. In addition, we evaluate our NLRN on the Urban100 dataset [45], which contains abundant structural patterns and textures, to further demonstrate the capability of using image self-similarity of our NLRN. The training set and test set are strictly disjoint and all the images are converted to gray-scale in each experiment setup. For image super-resolution, we follow [57, 63, 127] and use a training set of 291 images where 91 images are proposed in [7] and other 200 are from the BSD *train* set. We adopt four benchmark sets: Set5 [38], Set14 [39], BSD100 [40] and Urban100 [45] for testing with three upscaling factors $\times 2$, $\times 3$ and $\times 4$. The low-resolution images are synthesized by bicubic downsampling.

Training Settings

We randomly sample patches whose size equals the neighborhood of non-local operation from images during training. We use flipping, rotation and scaling for augmenting training data. For image denoising, we add independent and identically distributed Gaussian noise with zero mean to the original image as the noisy input during training. We train a different model for each noise level. For image super-resolution, only the luminance channel of images is super-resolved, and the other two color channels are upscaled by bicubic interpolation, following [57, 59, 63]. Moreover, the training images for all three upscaling factors: $\times 2$, $\times 3$ and $\times 4$ are upscaled by bicubic interpolation

Table 6.1: Image denoising comparison of our proposed model with various distance metrics on Set12 with noise level of 25.

Distance metric	$\phi(\boldsymbol{X}_i, \boldsymbol{X}_j)$	PSNR
Euclidean distance	$\exp\{-\left\|\boldsymbol{X}_i - \boldsymbol{X}_j\right\|_2^2 / h^2\}$	30.74
Dot product	$\boldsymbol{X}_i \boldsymbol{X}_j^T$	30.68
Embedded dot product	$\theta(\boldsymbol{X}_i)\psi(\boldsymbol{X}_j)^T$	30.75
Gaussian	$\exp\{\boldsymbol{X}_i \boldsymbol{X}_j^T\}$	30.69
Symmetric embedded Gaussian	$\exp\{\theta(\boldsymbol{X}_i)\theta(\boldsymbol{X}_j)^T\}$	30.76
Embedded Gaussian	$\exp\{\theta(\boldsymbol{X}_i)\psi(\boldsymbol{X}_j)^T\}$	30.80

into the desired spatial size and are combined into one training set. We use this set to train one single model for all these three upscaling factors as in [57, 63, 127].

We use Adam optimizer to minimize the loss function. We set the initial learning rate as 1e-3 and reduce it by half five times during training. We use Xavier initialization for the weights. We clip the gradient at the norm of 0.5 to prevent the gradient explosion which is shown to empirically accelerate training convergence, and we adopt 16 as the minibatch size during training. Training a model takes about three days with a Titan Xp GPU. For non-local module, we use circular padding for the neighborhood outside input patches. For convolution, we pad the boundaries of feature maps with zeros to preserve the spatial size of feature maps.

6.5.2 Model Analysis

In this section, we analyze our model in the following aspects. First, we conduct the ablation study of using different distance metrics in the non-local module. Table 6.1 compares instantiations including Euclidean distance, dot product, embedded dot product, Gaussian, symmetric embedded Gaussian and embedded Gaussian when used in NLRN of 12 unfolded steps. Embedded Gaussian achieves the best performance and is adopted in the following experiments.

We compare the NLRN with its variants in terms of PSNR in Table 6.2. We have a few observations. First, the same model with untied weights performs worse than its weight-sharing counter-part. We speculate that the model with untied weights is prone to model over-fitting and suffers much

Table 6.2: Image denoising comparison of our NLRN with its variants on Set12 with noise level of 25.

Model	PSNR
NLRN w/o parameter sharing	30.65
RNN with same parameter no.	30.61
Non-local module in every other state	30.76
Non-local module in every 3 states	30.72
NLRN w/o propagating correlations	30.78
NLRN	30.80

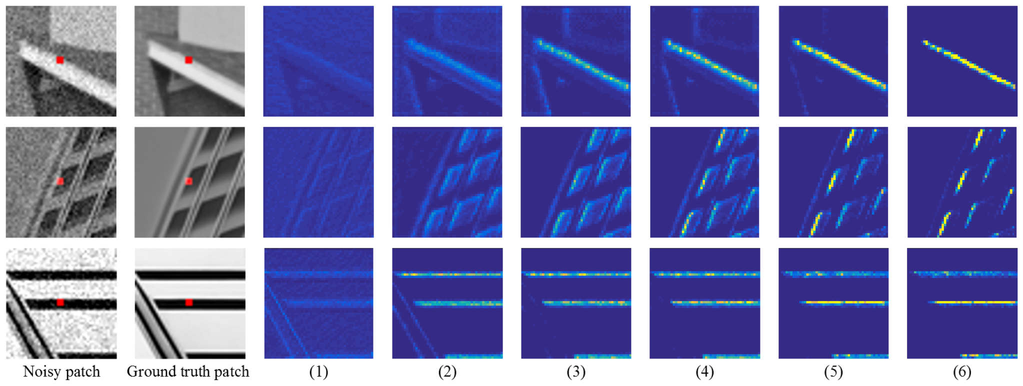


Figure 6.4: Examples of correlation maps of non-local operations for image denoising. Noisy patch/ground truth patch: the neighborhood of the red center pixel used in non-local operations. (1)-(6): the correlation map for recurrent state 1-6 from NLRN with unrolling length of 6.

slower training convergence, both of which undermine its performance. To investigate the function of non-local modules, we implement a baseline RNN with the same parameter number of NLRN, and find it is worse than NLRN by about 0.2 dB, showing the advantage of using non-local image properties for image restoration. Besides, we implement NLRNs where non-local module is used in every other state or every three states, and observe that if the frequency of using non-local modules in NLRN is reduced, the performance decreases accordingly. We show the benefit of propagating correlation information among adjacent states by comparing with the counter-part in terms of restoration accuracy. To further analyze the non-local module, we visualize the feature correlation maps for non-local operations in Figure 6.4. It can be seen that as the number of recurrent states increases, the locations with similar features progressively show higher correlations in the map, which demonstrates the effectiveness of the non-local module for exploiting

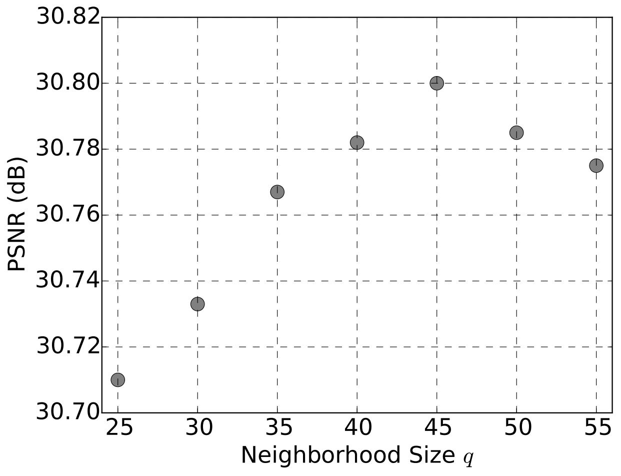


Figure 6.5: Neighborhood size vs. image denoising performance of our proposed model on Set12 with noise level of 25.

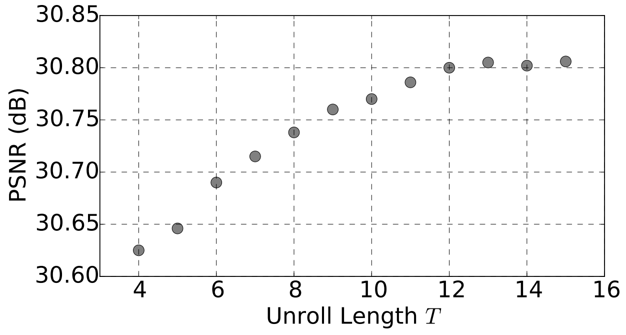


Figure 6.6: Unrolling length vs. image denoising performance of our proposed model on Set12 with noise level of 25.

image self-similarity.

Figure 6.5 investigates the influence of the neighborhood size in the non-local module on image denoising results. The performance peaks at $q = 45$. This shows that limiting the neighborhood helps concentrate the correlation calculation on relevant features in the spatial vicinity and enhance correlation estimation. Therefore, it is necessary to choose a proper neighborhood size (rather than the whole image) for image restoration. We select $q = 45$ for the rest of this section unless stated otherwise.

The unrolling length T determines the maximum effective depth (i.e., maximum number of convolutional layers) of NLRN. The influence of the unrolling length on image denoising results is shown in Figure 6.6. The performance increases as the unrolling length rises, but gets saturated after

Table 6.3: Benchmark image denoising results. Training and testing protocols are followed as in [100]. Average PSNR/SSIM for various noise levels on Set12, BSD68 and Urban100. The best performance is in bold.

Dataset	Noise	BM3D	WNNM	TNRD	NLNet	DnCNN	NLRN
Set12	15	32.37/0.8952	32.70/0.8982	32.50/0.8958	-/-	32.86/0.9031	**33.16/0.9070**
	25	39.97/0.8504	30.28/0.8557	30.06/0.8512	-/-	30.44/0.8622	**30.80/0.8689**
	50	26.72/0.7676	27.05/0.7775	26.81/0.7680	-/-	27.18/0.7829	**27.64/0.7980**
BSD68	15	31.07/0.8717	31.37/0.8766	31.42/0.8769	31.52/-	31.73/0.8907	**31.88/0.8932**
	25	28.57/0.8013	28.83/0.8087	28.92/0.8093	29.03/-	29.23/0.8278	**29.41/0.8331**
	50	25.62/0.6864	25.87/0.6982	25.97/0.6994	26.07/-	26.23/0.7189	**26.47/0.7298**
Urban100	15	32.35/0.9220	32.97/0.9271	31.86/0.9031	-/-	32.68/0.9255	**33.45/0.9354**
	25	29.70/0.8777	30.39/0.8885	29.25/0.8473	-/-	29.97/0.8797	**30.94/0.9018**
	50	25.95/0.7791	26.83/0.8047	25.88/0.7563	-/-	26.28/0.7874	**27.49/0.8279**

$T = 12$. Given the tradeoff between restoration accuracy and inference time, we adopt $T = 12$ for NLRN in all the experiments.

6.5.3 Comparisons with State-of-the-Art Methods

We compare our proposed model with a number of recent competitors for image denoising and image super-resolution, respectively. PSNR and SSIM [44] are adopted for measuring quantitative restoration performance.

Image Denoising

For a fair comparison with other methods based on deep networks, we train our model under two settings: (1) We use the training data as in TNRD [99], DnCNN [100] and NLNet [125], and the result is shown in Table 6.3. We cite the result of NLNet in the original paper [125], since no public code or model is available. (2) We use the training data as in RED [98] and MemNet [127], and the result is shown in Table 6.4. We note that RED uses multi-view testing [55] to boost the restoration accuracy, i.e., RED processes each test image as well as its rotated and flipped versions, and all the outputs are then averaged to form the final denoised image. Accordingly, we perform the same procedure for NLRN and find its performance, termed as *NLRN-MV*, is consistently improved. In addition, we include recent non-deep-learning based methods: BM3D [32] and WNNM [94] in our comparison. We do not list other methods [107, 97, 95, 110, 117] whose average performances are worse than DnCNN or MemNet. Our NLRN significantly outperforms all

Table 6.4: Benchmark image denoising results. Training and testing protocols are followed as in [127]. Average PSNR/SSIM for various noise levels on 14 images, BSD200 and Urban100. Red is the best and blue is the second best performance.

Dataset	Noise	BM3D	WNNM	RED	MemNet	NLRN	NLRN-MV
14 images	30	28.49/0.8204	28.74/0.8273	29.17/0.8423	29.22/0.8444	29.37/0.8460	29.41/0.8472
	50	26.08/0.7427	26.32/0.7517	26.81/0.7733	26.91/0.7775	27.00/0.7777	27.05/0.7791
	70	24.65/0.6882	24.80/0.6975	25.31/0.7206	25.43/0.7260	25.49/0.7255	25.54/0.7273
BSD200	30	27.31/0.7755	27.48/0.7807	27.95/0.8056	28.04/0.8053	28.15/0.8423	28.20/0.8436
	50	25.06/0.6831	25.26/0.6928	25.75/0.7167	25.86/0.7202	25.93/0.7214	25.97/0.8429
	70	23.82/0.6240	23.95/0.6346	24.37/0.6551	24.53/0.6608	24.58/0.6614	24.62/0.6634
Urban100	30	28.75/0.8567	29.47/0.8697	29.12/0.8674	29.10/0.8631	29.94/0.8830	29.99/0.8842
	50	25.95/0.7791	26.83/0.8047	26.44/0.7977	26.65/0.8030	27.38/0.8241	27.43/0.8256
	70	24.27/0.7163	25.11/0.7501	24.75/0.7415	25.01/0.7496	25.66/0.7707	25.71/0.7724

Table 6.5: Image denoising comparison of our proposed model with state-of-the-art network models on Set12 with noise level of 50. Model complexities are also compared.

	DnCNN	RED	MemNet	NLRN		
Max effective depth	17	30	80	38		
Parameter sharing	No	No	Yes	Yes		
Parameter no.	554k	4,131k	667k	330k		
Multi-view testing	No	Yes	No	No	No	Yes
Training images	400	300	300	400	300	300
PSNR	27.18	27.33	27.38	27.64	27.60	27.66

the competitors on Urban100 and yields the best results across almost all the noise levels and datasets.

To further show the advantage of the network design of NLRN, we compare different versions of NLRN with several state-of-the-art network models, i.e., DnCNN, RED and MemNet in Table 6.5. NLRN uses the fewest parameters but outperforms all the competitors. Specifically, NLRN benefits from inherent parameter sharing and uses only less than 1/10 parameters of RED. Compared with the RNN competitor, MemNet, NLRN uses only half of parameters and much shallower depth to obtain better performance, which shows the superiority of our non-local recurrent architecture.

Figure 6.7 shows the visual comparison of our NLRN and several competing image denoising methods: BM3D [32], WNNM [94], and MemNet [127]. Our method can recover more details from the noisy measurement.

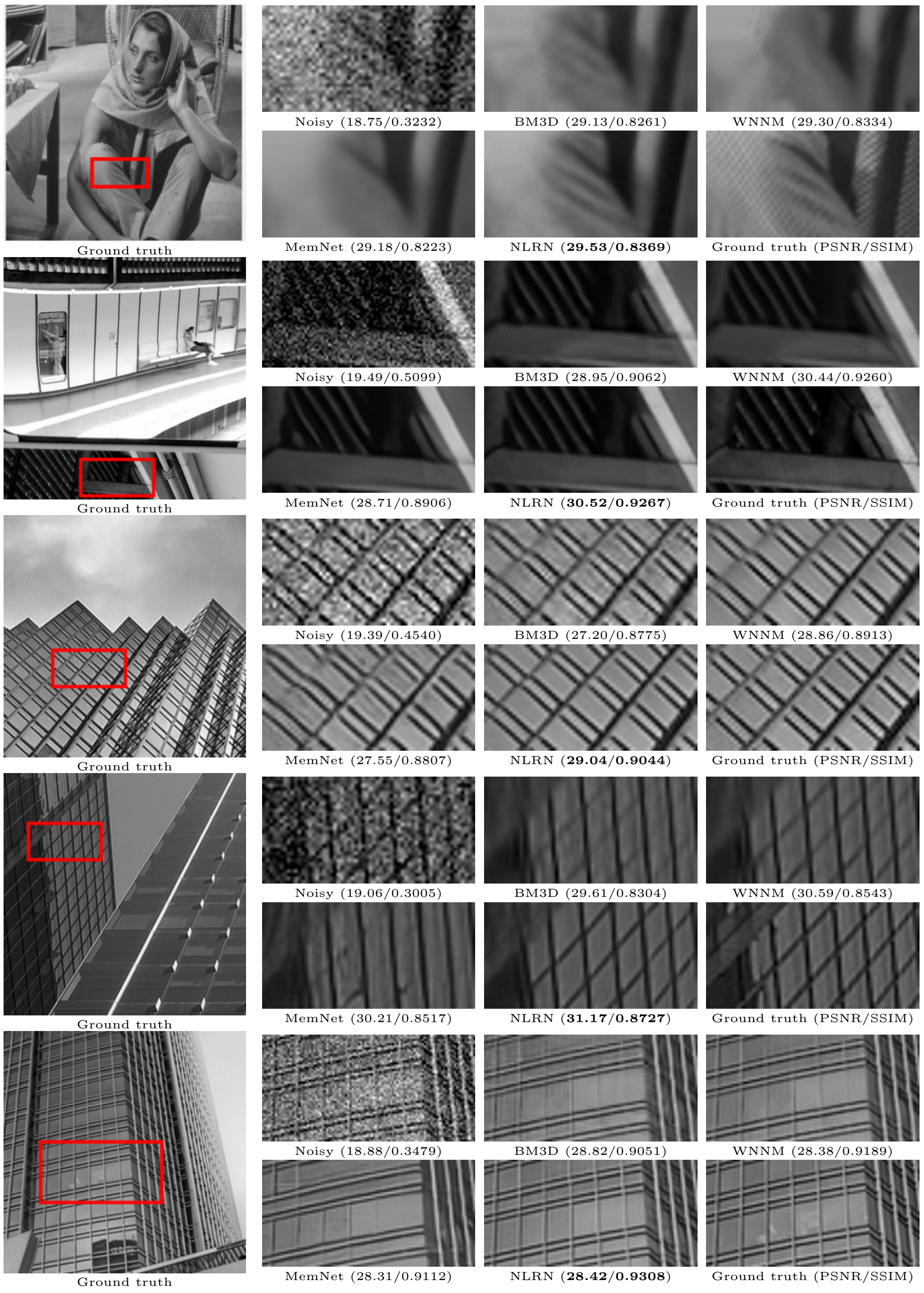


Figure 6.7: Qualitative comparison of image denoising results with noise level of 30. The zoom-in region in the red bounding box is shown on the right. From top to bottom: 1) the image *barbara*. 2) image *004* in Urban100. 3) image *019* in Urban100. 4) image *033* in Urban100. 5) image *046* in Urban100.

Table 6.6: Benchmark SISR results. Average PSNR/SSIM for scale factor ×2, ×3 and ×4 on datasets Set5, Set14, BSD100 and Urban100. The best performance is in bold.

Dataset	Scale	SRCNN	VDSR	DRCN	LapSRN	DRRN	MemNet	NLRN
Set5	×2	36.66/0.9542	37.53/0.9587	37.63/0.9588	37.52/0.959	37.74/0.9591	37.78/0.9597	**38.00/0.9603**
	×3	32.75/0.9090	33.66/0.9213	33.82/0.9226	33.82/0.923	34.03/0.9244	34.09/0.9248	**34.27/0.9266**
	×4	30.48/0.8628	31.35/0.8838	31.53/0.8854	31.54/0.885	31.68/0.8888	31.74/0.8893	**31.92/0.8916**
Set14	×2	32.45/0.9067	33.03/0.9124	33.04/0.9118	33.08/0.913	33.23/0.9136	33.28/0.9142	**33.46/0.9159**
	×3	29.30/0.8215	29.77/0.8314	29.76/0.8311	29.79/0.832	29.96/0.8349	30.00/0.8350	**30.16/0.8374**
	×4	27.50/0.7513	28.01/0.7674	28.02/0.7670	28.19/0.772	28.21/0.7721	28.26/0.7723	**28.36/0.7745**
BSD100	×2	31.36/0.8879	31.90/0.8960	31.85/0.8942	31.80/0.895	32.05/0.8973	32.08/0.8978	**32.19/0.8992**
	×3	28.41/0.7863	28.82/0.7976	28.80/0.7963	28.82/0.797	28.95/0.8004	28.96/0.8001	**29.06/0.8026**
	×4	26.90/0.7101	27.29/0.7251	27.23/0.7233	27.32/0.728	27.38/0.7284	27.40/0.7281	**27.48/0.7306**
Urban100	×2	29.50/0.8946	30.76/0.9140	30.75/0.9133	30.41/0.910	31.23/0.9188	31.31/0.9195	**31.81/0.9249**
	×3	26.24/0.7989	27.14/0.8279	27.15/0.8276	27.07/0.827	27.53/0.8378	27.56/0.8376	**27.93/0.8453**
	×4	24.52/0.7221	25.18/0.7524	25.14/0.7510	25.21/0.756	25.44/0.7638	25.50/0.7630	**25.79/0.7729**

Image Super-Resolution

We compare our model with several recent SISR approaches, including SRCNN [18], VDSR [57], DRCN [59], LapSRN [62], DRRN [63] and MemNet [127] in Table 6.6. We crop pixels near image borders before calculating PSNR and SSIM as in [18, 52, 57, 59]. We do not list other methods [45, 52, 55, 60] since their performances are worse than that of DRRN or MemNet. Besides, we do not include SRDenseNet [132] and EDSR [130] in the comparison because the number of parameters in these two network models is over two orders of magnitude larger than that of our NLRN and their training datasets are significantly larger than ours. It can be seen that NLRN yields the best result across all the upscaling factors and datasets. Figure 6.8 shows the visual comparison of our NLRN and several recent image SR methods: DRCN [59], LapSRN [62], DRRN [63] and MemNet [127]. Our method is able to reconstruct sharper edges and produce fewer artifacts especially in the regions of repetitive patterns.

6.6 Conclusion

We have presented a new and effective recurrent network that incorporates non-local operations for image restoration. The proposed non-local module can be trained end-to-end with the recurrent network. We have studied the importance of computing reliable feature correlations within a confined neighborhood against the whole image, and we have shown the benefits of passing feature correlation messages between adjacent recurrent stages. Comprehensive evaluations over benchmarks for image denoising and super-resolution demonstrate the superiority of NLRN over existing methods. For future work, we will explore the connection between image restoration and improving the image aesthetic quality using deep learning [142].

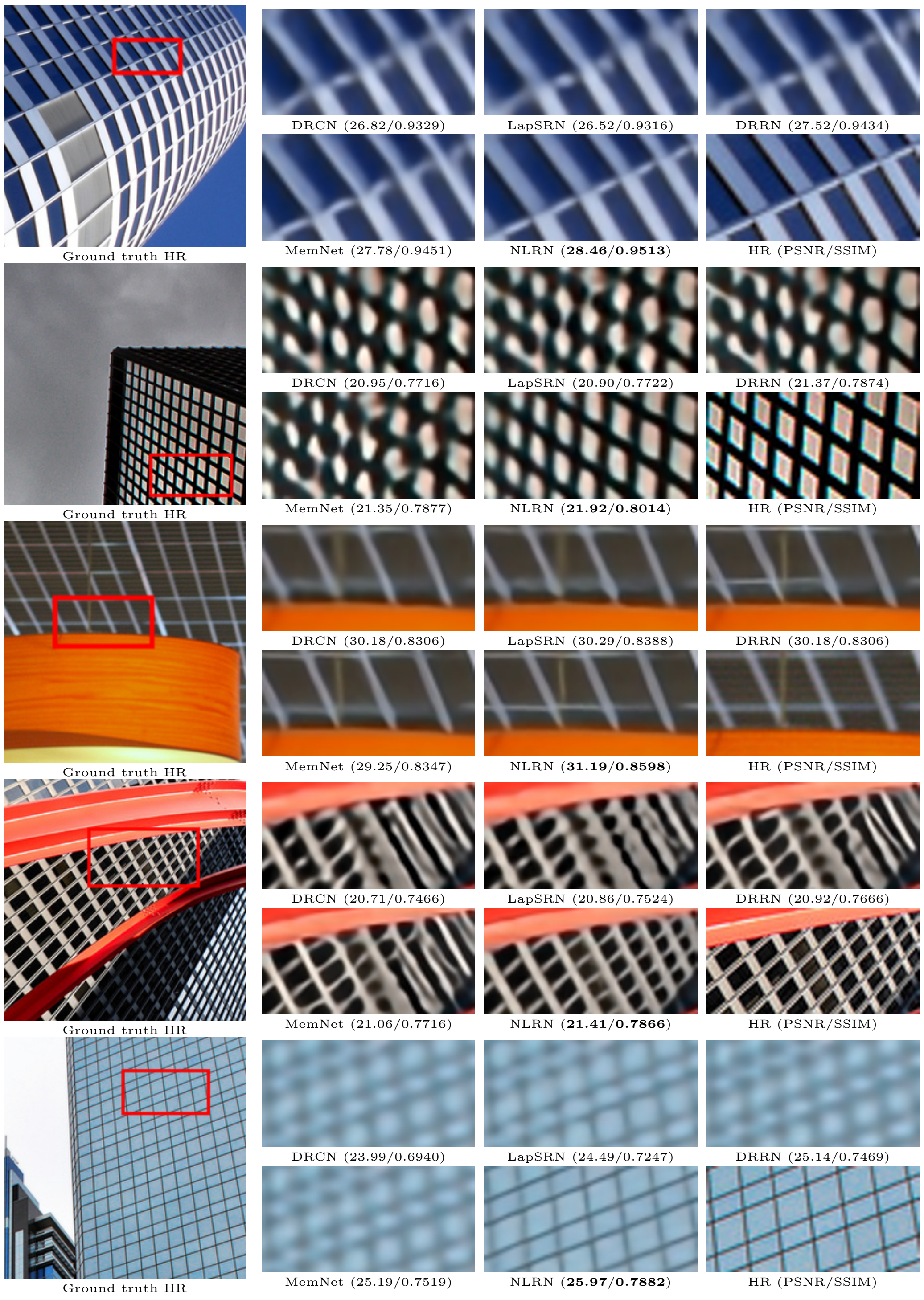


Figure 6.8: Qualitative comparison of image super-resolution results with $\times 4$ upscaling. The zoom-in region in the red bounding box is shown on the right. From top to bottom: 1) image *005* in Urban100. 2) image *019* in Urban100. 3) image *044* in Urban100. 4) image *062* in Urban100. 5) image *099* in Urban100.

Chapter 7

CONCLUSION

In this dissertation, we have studied the use of deep learning for image and video super-resolution and image denoising. The main contributions are summarized as follows.

First, we propose a model for image SR by combining the strengths of sparse coding and deep network, and improve SR accuracy over existing deep and shallow SR models both quantitatively and qualitatively. Besides producing better SR results, the domain knowledge of sparse representation also benefits training speed and model compactness. Moreover, we design a unified framework to jointly learn a mixture of deep networks for single image SR, each of which serves as a SR inference module to handle a certain class of image signals. An adaptive weight module is designed to predict pixel-level aggregation weights of the HR estimates. Extensive experiments show that our proposed model is able to achieve outstanding SR performance along with more flexibility of design.

Second, we propose a temporal adaptive network and explore several methods of image alignment, including a spatial alignment network, for learning the temporal dynamics to enhance video SR. Both the temporal adaptation and the enhanced spatial alignment increase the robustness to complex motion which benefits video SR.

Third, we explore the connection between image denoising and high-level semantic tasks, which is of great practical value in various applications of computer vision. We tackle this challenge in a simple yet efficient way by allowing the high-level semantic information flowing back to the low-level vision network, which achieves superior performance in both image denoising and high-level vision tasks. In our training strategy, the denoiser trained for one high-level task has the generality to be applied to other high-level vision tasks. Overall, it provides a feasible and robust solution in a deep learning fashion to real world problems.

Last, we present an effective recurrent network that incorporates non-local operations for image restoration. The proposed non-local module can be trained end-to-end with the recurrent network. We study the importance of computing reliable feature correlations with a proper neighborhood size, and show the benefits of passing feature correlation messages between adjacent recurrent stages.

REFERENCES

[1] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015, pp. 1–9.

[2] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," *arXiv preprint arXiv:1711.07971*, 2017.

[3] S. Baker and T. Kanade, "Limits on super-resolution and how to break them," *IEEE TPAMI*, vol. 24, no. 9, pp. 1167–1183, 2002.

[4] Z. Lin and H.-Y. Shum, "Fundamental limits of reconstruction-based superresolution algorithms under local translation," *IEEE TPAMI*, vol. 26, no. 1, pp. 83–97, 2004.

[5] R. Fattal, "Image upsampling via imposed edge statistics," in *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3. ACM, 2007, p. 95.

[6] H. A. Aly and E. Dubois, "Image up-sampling using total-variation regularization with a new observation model," *IEEE TIP*, vol. 14, no. 10, pp. 1647–1659, 2005.

[7] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE TIP*, 2010.

[8] J. Yang, Z. Wang, Z. Lin, S. Cohen, and T. Huang, "Coupled dictionary training for image super-resolution," *IEEE TIP*, vol. 21, no. 8, pp. 3467–3478, 2012.

[9] X. Gao, K. Zhang, D. Tao, and X. Li, "Image super-resolution with sparse neighbor embedding," *IEEE TIP*, vol. 21, no. 7, pp. 3194–3205, 2012.

[10] D. Glasner, S. Bagon, and M. Irani, "Super-resolution from a single image," in *ICCV*, 2009.

[11] G. Freedman and R. Fattal, "Image and video upscaling from local self-examples," *ACM Transactions on Graphics (TOG)*, 2011.

[12] K. I. Kim and Y. Kwon, "Single-image super-resolution using sparse regression and natural image prior," *IEEE TPAMI*, vol. 32, no. 6, pp. 1127–1133, 2010.

[13] C. Deng, J. Xu, K. Zhang, D. Tao, X. Gao, and X. Li, "Similarity constraints-based structured output regression machine: An approach to image super-resolution," *IEEE Transactions on Neural Networks and Learning Systems*, 2015.

[14] C.-Y. Yang, C. Ma, and M.-H. Yang, "Single-image super-resolution: a benchmark," in *ECCV*, 2014, pp. 372–386.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.

[16] Z. Cui, H. Chang, S. Shan, B. Zhong, and X. Chen, "Deep network cascade for image super-resolution," in *ECCV*. Springer, 2014, pp. 49–64.

[17] Z. Wang, Y. Yang, Z. Wang, S. Chang, W. Han, J. Yang, and T. Huang, "Self-tuned deep super resolution," in *CVPR Workshops*, 2015, pp. 1–8.

[18] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *ECCV*, 2014.

[19] C. Osendorfer, H. Soyer, and P. van der Smagt, "Image super-resolution with fast approximate convolutional sparse coding," in *Neural Information Processing*. Springer, 2014, pp. 250–257.

[20] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *ICML*, 2010, pp. 399–406.

[21] B. Wen, S. Ravishankar, and Y. Bresler, "Structured overcomplete sparsifying transform learning with convergence guarantees and applications," *IJCV*, 2015.

[22] K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "Fast inference in sparse coding algorithms with applications to object recognition," *arXiv preprint arXiv:1010.3467*, 2010.

[23] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on Pure and Applied Mathematics*, vol. 57, no. 11, pp. 1413–1457, 2004.

[24] C. J. Rozell, D. H. Johnson, R. G. Baraniuk, and B. A. Olshausen, "Sparse coding via thresholding and local competition in neural circuits," *Neural Computation*, vol. 20, no. 10, pp. 2526–2563, 2008.

[25] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[26] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *ICML*, 2010, pp. 807–814.

[27] R. Timofte, V. De, and L. Van Gool, "Anchored neighborhood regression for fast example-based super-resolution," in *ICCV*, 2013.

[28] Q. V. Le, "Building high-level features using large scale unsupervised learning," in *ICASSP*, 2013, pp. 8595–8598.

[29] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *CVPR*, 2014, pp. 1717–1724.

[30] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *CVPR*, 2005.

[31] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE TIP*, vol. 54, no. 11, pp. 4311–4322, 2006.

[32] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE TIP*, 2007.

[33] B. Wen, S. Ravishankar, and Y. Bresler, "Video denoising by online 3d sparsifying transform learning," in *ICIP*, 2015.

[34] S. Ravishankar, B. Wen, and Y. Bresler, "Online sparsifying transform learningpart i: Algorithms," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 4, pp. 625–636, 2015.

[35] Z. Wang, Y. Yang, Z. Wang, S. Chang, J. Yang, and T. S. Huang, "Learning super-resolution jointly from external and internal examples," *IEEE TIP*, vol. 24, no. 11, pp. 4359–4371, 2015.

[36] J. Yang, Z. Lin, and S. Cohen, "Fast image super-resolution based on in-place example regression," in *CVPR*, 2013, pp. 1059–1066.

[37] R. A. Bradley and M. E. Terry, "Rank analysis of incomplete block designs: I. the method of paired comparisons," *Biometrika*, pp. 324–345, 1952.

[38] M. Bevilacqua, A. Roumy, C. Guillemot, and M.-L. A. Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," in *BMVC*. BMVA Press, 2012.

[39] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *International Conference on Curves and Surfaces*, 2010.

[40] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *ICCV*, 2001.

[41] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *TPAMI*, vol. 38, no. 2, pp. 295–307, 2016.

[42] R. Timofte, V. De Smet, and L. Van Gool, "A+: Adjusted anchored neighborhood regression for fast super-resolution," in *ACCV*. Springer, 2014, pp. 111–126.

[43] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

[44] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE TIP*, 2004.

[45] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *CVPR*, 2015.

[46] W. Dong, L. Zhang, and G. Shi, "Centralized sparse representation for image restoration," in *ICCV*, 2011, pp. 1259–1266.

[47] K. Zhang, X. Gao, D. Tao, and X. Li, "Single image super-resolution with non-local means and steering kernel regression," *IEEE TIP*, vol. 21, no. 11, pp. 4544–4556, 2012.

[48] X. Lu, H. Yuan, P. Yan, Y. Yuan, and X. Li, "Geometry constrained sparse coding for single image super-resolution," in *CVPR*, 2012, pp. 1648–1655.

[49] S. C. Park, M. K. Park, and M. G. Kang, "Super-resolution image reconstruction: a technical overview," *IEEE Signal Processing Magazine*, vol. 20, no. 3, pp. 21–36, 2003.

[50] B. S. Morse and D. Schwartzwald, "Image magnification using level-set reconstruction," in *CVPR 2001*, vol. 1. IEEE, 2001, pp. I–333.

[51] H. Chang, D.-Y. Yeung, and Y. Xiong, "Super-resolution through neighbor embedding," in *CVPR*, 2004.

[52] S. Schulter, C. Leistner, and H. Bischof, "Fast and accurate image upscaling with super-resolution forests," in *CVPR*, 2015.

[53] D. Dai, R. Timofte, and L. Van Gool, "Jointly optimized regressors for image super-resolution," in *Eurographics*, vol. 7, 2015, p. 8.

[54] R. Timofte, R. Rasmus, and L. Van Gool, "Seven ways to improve example-based single image super resolution," in *CVPR*. IEEE, 2016.

[55] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang, "Deep networks for image super-resolution with sparse prior," in *ICCV*, 2015.

[56] D. Liu, Z. Wang, B. Wen, J. Yang, W. Han, and T. S. Huang, "Robust single image super-resolution via deep networks with sparse prior," *IEEE TIP*, vol. 25, no. 7, pp. 3194–3207, 2016.

[57] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *CVPR*, 2016.

[58] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM International Conference on Multimedia*. ACM, 2014, pp. 675–678.

[59] J. Kim, J. Kwon Lee, and K. Mu Lee, "Deeply-recursive convolutional network for image super-resolution," in *CVPR*, 2016.

[60] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *CVPR*, 2016.

[61] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *ECCV*, 2016.

[62] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep Laplacian pyramid networks for fast and accurate super-resolution," in *CVPR*, 2017.

[63] Y. Tai, J. Yang, and X. Liu, "Image super-resolution via deep recursive residual network," in *CVPR*, 2017.

[64] M. Dorr, T. Martinetz, K. R. Gegenfurtner, and E. Barth, "Variability of eye movements when viewing dynamic natural scenes," *Journal of vision*, vol. 10, no. 10, pp. 28–28, 2010.

[65] S. Farsiu, M. D. Robinson, M. Elad, and P. Milanfar, “Fast and robust multiframe super resolution,” *TIP*, vol. 13, no. 10, pp. 1327–1344, 2004.

[66] H. Takeda, P. Milanfar, M. Protter, and M. Elad, “Super-resolution without explicit subpixel motion estimation,” *TIP*, vol. 18, no. 9, pp. 1958–1975, 2009.

[67] S. P. Belekos, N. P. Galatsanos, and A. K. Katsaggelos, “Maximum a posteriori video super-resolution using a new multichannel image prior,” *TIP*, vol. 19, no. 6, pp. 1451–1464, 2010.

[68] C. Liu and D. Sun, “On Bayesian adaptive video super resolution,” *TPAMI*, vol. 36, no. 2, pp. 346–360, 2014.

[69] Z. Ma, R. Liao, X. Tao, L. Xu, J. Jia, and E. Wu, “Handling motion blur in multi-frame super-resolution,” in *CVPR*, 2015, pp. 5224–5232.

[70] R. Liao, X. Tao, R. Li, Z. Ma, and J. Jia, “Video super-resolution via deep draft-ensemble learning,” in *CVPR*, 2015, pp. 531–539.

[71] Q. Dai, S. Yoo, A. Kappeler, and A. K. Katsaggelos, “Sparse representation-based multiple frame video super-resolution,” *TIP*, vol. 26, no. 2, pp. 765–781, 2017.

[72] Y. Huang, W. Wang, and L. Wang, “Bidirectional recurrent convolutional networks for multi-frame super-resolution,” in *NIPS*, 2015, pp. 235–243.

[73] A. Kappeler, S. Yoo, Q. Dai, and A. K. Katsaggelos, “Video super-resolution with convolutional neural networks,” *IEEE Transactions on Computational Imaging*, vol. 2, no. 2, pp. 109–122, 2016.

[74] M. Jaderberg, K. Simonyan, A. Zisserman et al., “Spatial transformer networks,” in *NIPS*, 2015, pp. 2017–2025.

[75] D. Liu, Z. Wang, Y. Fan, X. Liu, Z. Wang, S. Chang, X. Wang, and T. S. Huang, “Learning temporal dynamics for video super-resolution: A deep learning approach,” *IEEE TIP*, vol. 27, no. 7, pp. 3432–3445, 2018.

[76] D. Liu, Z. Wang, N. Nasrabadi, and T. Huang, “Learning a mixture of deep networks for single image super-resolution,” in *ACCV*. Springer, 2016, pp. 145–156.

[77] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang et al., “Photo-realistic single image super-resolution using a generative adversarial network,” in *CVPR*, 2017.

[78] H. Yu, D. Liu, H. Shi, H. Yu, Z. Wang, X. Wang, B. Cross, M. Bramler, and T. S. Huang, "Computed tomography super-resolution using convolutional neural networks," in *ICIP*. IEEE, 2017, pp. 3944–3948.

[79] W. Han, S. Chang, D. Liu, M. Yu, M. Witbrock, and T. S. Huang, "Image super-resolution via dual-state recurrent networks," in *CVPR*, June 2018.

[80] K. Seshadrinathan, R. Soundararajan, A. C. Bovik, and L. K. Cormack, "Study of subjective and objective quality assessment of video," *TIP*, vol. 19, no. 6, pp. 1427–1441, 2010.

[81] J. Y. Lin, R. Song, C.-H. Wu, T. Liu, H. Wang, and C.-C. J. Kuo, "Mcl-v: A streaming video quality assessment database," *Journal of Visual Communication and Image Representation*, vol. 30, pp. 1–9, 2015.

[82] C. Keimel, J. Habigt, T. Habigt, M. Rothbucher, and K. Diepold, "Visual quality of current coding technologies at high definition iptv bitrates," in *Multimedia Signal Processing (MMSP), 2010 IEEE International Workshop on*. IEEE, 2010, pp. 390–393.

[83] http://ultravideo.cs.tut.fi/.

[84] C. Liu, "Beyond pixels: exploring new representations and applications for motion analysis," Ph.D. dissertation, Citeseer, 2009.

[85] K. Seshadrinathan and A. C. Bovik, "Motion tuned spatio-temporal quality assessment of natural videos," *TIP*, vol. 19, no. 2, pp. 335–350, 2010.

[86] D. Dai, Y. Wang, Y. Chen, and L. Van Gool, "Is image super-resolution helpful for other vision tasks?" in *WACV*. IEEE, 2016, pp. 1–9.

[87] Z. Wang, S. Chang, Y. Yang, D. Liu, and T. S. Huang, "Studying very low resolution recognition using deep networks," in *CVPR*. IEEE, 2016, pp. 4792–4800.

[88] B. Cheng, Z. Wang, Z. Zhang, Z. Li, D. Liu, J. Yang, S. Huang, and T. S. Huang, "Robust emotion recognition from low quality and low bit rate video: A deep learning approach," in *Affective Computing and Intelligent Interaction (ACII), 2017 Seventh International Conference on*. IEEE, 2017, pp. 65–70.

[89] Y. Zhou, D. Liu, and T. Huang, "Survey of face detection on low-quality images," in *Automatic Face & Gesture Recognition (FG 2018)*. IEEE, 2018, pp. 769–773.

[90] L. Wolf, T. Hassner, and I. Maoz, “Face recognition in unconstrained videos with matched background similarity,” in *CVPR*. IEEE, 2011, pp. 529–534.

[91] M. Aharon, M. Elad, and A. Bruckstein, “K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation,” *TSP*, vol. 54, no. 11, pp. 4311–4322, 2006.

[92] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, “Non-local sparse models for image restoration,” in *ICCV*, 2009.

[93] W. Dong, L. Zhang, G. Shi, and X. Li, “Nonlocally centralized sparse representation for image restoration,” *TIP*, vol. 22, no. 4, pp. 1620–1630, 2013.

[94] S. Gu, L. Zhang, W. Zuo, and X. Feng, “Weighted nuclear norm minimization with application to image denoising,” in *CVPR*, 2014, pp. 2862–2869.

[95] J. Xu, L. Zhang, W. Zuo, D. Zhang, and X. Feng, “Patch group based nonlocal self-similarity prior learning for image denoising,” in *ICCV*, 2015.

[96] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, “Extracting and composing robust features with denoising autoencoders,” in *ICML*. ACM, 2008, pp. 1096–1103.

[97] H. C. Burger, C. J. Schuler, and S. Harmeling, “Image denoising: Can plain neural networks compete with BM3D?” in *CVPR*. IEEE, 2012, pp. 2392–2399.

[98] X. Mao, C. Shen, and Y.-B. Yang, “Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections,” in *NIPS*, 2016.

[99] Y. Chen and T. Pock, “Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration,” *IEEE TPAMI*, 2017.

[100] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, “Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising,” *IEEE TIP*, 2017.

[101] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *MICCAI*. Springer, 2015, pp. 234–241.

[102] J. Wu, R. Timofte, Z. Huang, and L. Van Gool, "On the relation between color image denoising and classification," *arXiv preprint arXiv:1704.01372*, 2017.

[103] D. Liu, B. Cheng, Z. Wang, H. Zhang, and T. S. Huang, "Enhance visual recognition under adverse conditions via deep networks," *arXiv preprint arXiv:1712.07732*, 2017.

[104] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[105] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *CVPR*, 2015, pp. 427–436.

[106] D. Liu, B. Wen, X. Liu, Z. Wang, and T. S. Huang, "When image denoising meets high-level vision tasks: A deep learning approach," in *IJCAI*, 2018.

[107] D. Zoran and Y. Weiss, "From learning models of natural image patches to whole image restoration," in *ICCV*, 2011.

[108] W. Dong, X. Li, L. Zhang, and G. Shi, "Sparsity-based image denoising via dictionary learning and structural clustering," in *CVPR*, 2011.

[109] J. Zhang, D. Zhao, and W. Gao, "Group-based sparse representation for image restoration," *TIP*, vol. 23, no. 8, pp. 3336–3351, 2014.

[110] F. Chen, L. Zhang, and H. Yu, "External patch prior guided internal clustering for image denoising," in *ICCV*, 2015.

[111] B. Wen, Y. Li, and Y. Bresler, "When sparsity meets low-rankness: Transform learning with non-local low-rank constraint for image restoration," in *ICASSP*, 2017.

[112] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Color image denoising via sparse 3d collaborative filtering with grouping constraint in luminance-chrominance space," in *ICIP*, vol. 1. IEEE, 2007, pp. I–313.

[113] V. Jain and S. Seung, "Natural image denoising with convolutional networks," in *NIPS*, 2009, pp. 769–776.

[114] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *NIPS*, 2012, pp. 341–349.

[115] Y. Kim, H. Jung, D. Min, and K. Sohn, "Deeply aggregated alternating minimization for image restoration," in *CVPR*, July 2017.

[116] U. Schmidt and S. Roth, "Shrinkage fields for effective image restoration," in *CVPR*, 2014.

[117] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep CNN denoiser prior for image restoration," in *CVPR*, 2017.

[118] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[119] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *ECCV*. Springer, 2016, pp. 694–711.

[120] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected CRFs," *arXiv preprint arXiv:1412.7062*, 2014.

[121] J. Xu, L. Zhang, D. Zhang, and X. Feng, "Multi-channel weighted nuclear norm minimization for real color image denoising," in *ICCV*, 2017.

[122] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[123] Q. Huynh-Thu and M. Ghanbari, "Scope of validity of PSNR in image/video quality assessment," *Electronics Letters*, vol. 44, no. 13, pp. 800–801, 2008.

[124] L. I. Rudin and S. Osher, "Total variation based image restoration with free local constraints," in *ICIP*, 1994.

[125] S. Lefkimmiatis, "Non-local color image denoising with convolutional neural networks," in *CVPR*, 2017.

[126] P. Qiao, Y. Dou, W. Feng, R. Li, and Y. Chen, "Learning non-local image diffusion for image denoising," in *ACM on Multimedia Conference*, 2017.

[127] Y. Tai, J. Yang, X. Liu, and C. Xu, "Memnet: A persistent memory network for image restoration," in *ICCV*, 2017.

[128] D. Liu, B. Wen, Y. Fan, C. C. Loy, and T. S. Huang, "Non-local recurrent network for image restoration," *arXiv preprint arXiv:1806.02919*, 2018.

[129] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *ICCV*, 1998.

[130] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *CVPR Workshops*, 2017.

[131] Y. Fan, H. Shi, J. Yu, D. Liu, W. Han, H. Yu, Z. Wang, X. Wang, and T. S. Huang, "Balanced two-stage residual networks for image super-resolution," in *CVPR Workshops (CVPRW)*. IEEE, 2017, pp. 1157–1164.

[132] T. Tong, G. Li, X. Liu, and Q. Gao, "Image super-resolution using dense skip connections," in *ICCV*, 2017.

[133] D. Liu, Z. Wang, Y. Fan, X. Liu, Z. Wang, S. Chang, and T. S. Huang, "Robust video super-resolution with learned temporal dynamics," in *ICCV*, 2017.

[134] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *ICCV*, 2015.

[135] S. Chandra, N. Usunier, and I. Kokkinos, "Dense and low-rank Gaussian CRFs using deep embeddings," in *ICCV*, 2017.

[136] A. W. Harley, K. G. Derpanis, and I. Kokkinos, "Segmentation-aware convolutional networks using local attention masks," in *ICCV*, 2017.

[137] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap, "A simple neural network module for relational reasoning," in *NIPS*, 2017.

[138] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *ICML*, 2017.

[139] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017.

[140] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE TIP*, 2006.

[141] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *ECCV*, 2016.

[142] Z. Wang, D. Liu, S. Chang, F. Dolcos, D. Beck, and T. Huang, "Image aesthetics assessment using deep Chatterjee's machine," in *IJCNN*. IEEE, 2017, pp. 941–948.